Fejlesztői dokumentáció

Adatbetöltő modul

Felhasznált library-k, programok 5

Szerver oldal 5

Kliens oldal 5

Szerver oldali programok 5

Frontend 5

HTML 5

404.html 5

CSS 6

default.css 6

Dictionaries 11

hungary.dic 11

PHP 13

Index.php 13

Template 17

resources/templates 17

footer.tpl 17

header.tpl 18

admin.tpl 18

admin/batchUpload.tpl 18

admin/singleUpload.tpl 19

JS 19

model.js 19

script.js 20

uploadvm.js 20

utils.js 24

Kommunikációs réteg 27

Restful 27

dispatch.php 27

app/routes/uploadServices.php 27

.htaccess 31

Backend 32

Config 32

config.php 32

library/etraffic/utils 32

utils.php 32

Repository 35

etraffic/repository/repository.php 35

Entity 56

etraffic/orm/generated-classes/Base 56

Stg10FonelNagyobbSzervezetek.php 56

Stg10FonelNagyobbSzervezetekQuery.php 84

StgBelfoldiOsszesJovedelem.php 94

StgBelfoldiOsszesJovedelemQuery.php 120

StgCegautok.php 131

StgCegautokQuery.php 158

StgEgeszsegugyiSzocialisEllatottakElvandoroltak.php 178

StgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery.php 218

StgEgyebIntezmenyiEllatottsagSulya.php 241

StgEgyebIntezmenyiEllatottsagSulyaQuery.php 286

StgElelmiszeruzletekEsAruhazak.php 305

StgElelmiszeruzletekEsAruhazakQuery.php 333

StgErintettNepessegSzama.php 343

StgErintettNepessegSzamaQuery.php 387

StgErtekesitesNettoArbevetele.php 413

StgErtekesitesNettoArbeveteleQuery.php 439

StgFoglalkoztatottakSzama.php 448

StgFoglalkoztatottakSzamaQuery.php 476

StgJovedelemEsAllaskeresokSzama.php 486

StgJovedelemEsAllaskeresokSzamaQuery.php 515

StgJovedelmiHelyzet.php 527

StgJovedelmiHelyzetQuery.php 554

StgKeresztmetszetiForgalom.php 564

StgKeresztmetszetiForgalomQuery.php 646

StgKeresztmetszetiForgalomV1.php 697

StgKeresztmetszetiForgalomV1Query.php 734

StgKollegium.php 751

StgKollegiumQuery.php 783

StgKulturaEsSport.php 796

StgKulturaEsSportQuery.php 835

StgKulturalisFunkciokAranya.php 850

StgKulturalisFunkciokAranyaQuery.php 879

StgLakonepesseg.php 891

StgLakonepessegQuery.php 916

StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok.php 925

StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery.php 954

StgMukodoVallalkozasokSzama.php 965

StgMukodoVallalkozasokSzamaAKiemeltIparagakban.php 992

StgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery.php 1026

StgMukodoVallalkozasokSzamaQuery.php 1043

StgMunkakepesKoruNepesseg.php 1052

StgMunkakepesKoruNepessegQuery.php 1078

StgNemzetkoziTranzitforgalom.php 1086

StgNemzetkoziTranzitforgalomQuery.php 1112

StgNodeAttribs.php 1121

StgNodeAttribsQuery.php 1190

StgNodes.php 1233

StgNodesQuery.php 1260

StgOktatasiFerohelyekSzama.php 1269

StgOktatasiFerohelyekSzamaQuery.php 1303

StgSzabadIskolavalasztasValtozo.php 1318

StgSzabadIskolavalasztasValtozoQuery.php 1356

StgSzallashely.php 1376

StgSzallashelyQuery.php 1404

StgSzemelygepkocsik.php 1414

StgSzemelygepkocsikQuery.php 1424

StgTanulasiCeluUtazasokAltalanosJellemzese.php 1432

StgTanulasiCeluUtazasokAltalanosJellemzeseQuery.php 1464

StgTanulokEsHallgatokSzama.php 1478

StgTanulokEsHallgatokSzamaQuery.php 1516

StgTelepules.php 1536

StgTelepulesKoordinata.php 1573

StgTelepulesKoordinataQuery.php 1601

StgTelepulesQuery.php 1613

StgTomegkozlekedes.php 1629

StgTomegkozlekedesQuery.php 1657

StgUgyintezesAltalanosJellemzese.php 1666

StgUgyintezesAltalanosJellemzeseQuery.php 1711

StgUzletekVendeglatohelyekGyogyszertarak.php 1737

StgUzletekVendeglatohelyekGyogyszertarakQuery.php 1770

StgVandorlasiEgyenlegAranya.php 1785

StgVandorlasiEgyenlegAranyaQuery.php 1816

Table maping 1829

Stg10FonelNagyobbSzervezetekTableMap.php 1829

StgBelfoldiOsszesJovedelemTableMap.php 1839

StgCegautokTableMap.php 1849

StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap.php 1860

StgEgyebIntezmenyiEllatottsagSulyaTableMap.php 1875

StgElelmiszeruzletekEsAruhazakTableMap.php 1889

StgErintettNepessegSzamaTableMap.php 1900

StgErtekesitesNettoArbeveteleTableMap.php 1914

StgFoglalkoztatottakSzamaTableMap.php 1924

StgJovedelemEsAllaskeresokSzamaTableMap.php 1935

StgJovedelmiHelyzetTableMap.php 1946

StgKeresztmetszetiForgalomTableMap.php 1956

StgKeresztmetszetiForgalomV1TableMap.php 1976

StgKollegiumTableMap.php 1988

StgKulturaEsSportTableMap.php 2000

StgKulturalisFunkciokAranyaTableMap.php 2011

StgLakonepessegTableMap.php 2022

StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap.php 2032

StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap.php 2043

StgMukodoVallalkozasokSzamaTableMap.php 2057

StgMunkakepesKoruNepessegTableMap.php 2067

StgNemzetkoziTranzitforgalomTableMap.php 2077

StgNodeAttribsTableMap.php 2087

StgNodesTableMap.php 2104

StgOktatasiFerohelyekSzamaTableMap.php 2113

StgSzabadIskolavalasztasValtozoTableMap.php 2126

StgSzallashelyTableMap.php 2140

StgSzemelygepkocsikTableMap.php 2150

StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap.php 2160

StgTanulokEsHallgatokSzamaTableMap.php 2172

StgTelepulesKoordinataTableMap.php 2187

StgTelepulesTableMap.php 2197

StgTomegkozlekedesTableMap.php 2209

StgUgyintezesAltalanosJellemzeseTableMap.php 2219

StgUzletekVendeglatohelyekGyogyszertarakTableMap.php 2234

StgVandorlasiEgyenlegAranyaTableMap.php 2247

SQL 2258

stg\_10\_fonel\_nagyobb\_szervezetek 2258

stg\_elelmiszeruzletek\_es\_aruhazak 2258

stg\_jovedelmi\_helyzet 2259

stg\_kulturalis\_funkciok\_aranya 2259

stg\_munkakepes\_koru\_nepesseg 2259

stg\_szabad\_iskolavalasztas\_valtozo 2259

stg\_telepules 2260

stg\_vandorlasi\_egyenleg\_aranya 2260

stg\_belfoldi\_osszes\_jovedelem 2261

stg\_erintett\_nepesseg\_szama 2261

stg\_keresztmetszeti\_forgalom 2262

stg\_lakonepesseg 2263

stg\_nemzetkozi\_tranzitforgalom 2263

stg\_szallashely 2264

stg\_telepules\_koordinata 2264

stg\_cegautok 2264

stg\_ertekesites\_netto\_arbevetele 2264

stg\_keresztmetszeti\_forgalom\_v1 2265

stg\_keresztmetszeti\_forgalom\_v1 2265

stg\_node\_attribs 2266

stg\_szemelygepkocsik 2267

stg\_tomegkozlekedes 2267

stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak 2268

stg\_foglalkoztatottak\_szama 2268

stg\_kollegium 2268

stg\_mukodo\_vallalkozasok\_szama 2269

stg\_nodes 2269

stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese 2269

stg\_ugyintezes\_altalanos\_jellemzese 2270

stg\_egyeb\_intezmenyi\_ellatottsag\_sulya 2270

stg\_jovedelem\_es\_allaskeresok\_szama` 2271

stg\_kultura\_es\_sport 2271

stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban 2271

stg\_oktatasi\_ferohelyek\_szama 2272

stg\_tanulok\_es\_hallgatok\_szama 2272

stg\_uzletek\_vendeglatohelyek\_gyogyszertarak 2273

# Felhasznált library-k, programok

## Szerver oldal

* PHP 5 <http://php.net/>
* Propel orm <http://propelorm.org/>
* PHPExcel <https://phpexcel.codeplex.com/>
* Slim <http://www.slimframework.com/>
* Smarty <http://www.smarty.net/>

## Kliens oldal

* JQuery <https://jquery.com/>
* Knockoutjs <http://knockoutjs.com/>
* Jqueryui <https://jqueryui.com/>
* Knockoutjs-jqueryui <http://gvas.github.io/knockout-jqueryui/>

## Szerver oldali programok

* Apache 2.4 <http://httpd.apache.org/>
* MySQL 5.6 <https://www.mysql.com/>

# Frontend

A frontend állományai a public\_html könyvtárban helyezkendek el

## HTML

### 404.html

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Page Not Found :(</title>

<style>

::-moz-selection { background: #fe57a1; color: #fff; text-shadow: none; }

::selection { background: #fe57a1; color: #fff; text-shadow: none; }

html { padding: 30px 10px; font-size: 20px; line-height: 1.4; color: #737373; background: #f0f0f0; -webkit-text-size-adjust: 100%; -ms-text-size-adjust: 100%; }

html, input { font-family: "Helvetica Neue", Helvetica, Arial, sans-serif; }

body { max-width: 500px; \_width: 500px; padding: 30px 20px 50px; border: 1px solid #b3b3b3; border-radius: 4px; margin: 0 auto; box-shadow: 0 1px 10px #a7a7a7, inset 0 1px 0 #fff; background: #fcfcfc; }

h1 { margin: 0 10px; font-size: 50px; text-align: center; }

h1 span { color: #bbb; }

h3 { margin: 1.5em 0 0.5em; }

p { margin: 1em 0; }

ul { padding: 0 0 0 40px; margin: 1em 0; }

.container { max-width: 380px; \_width: 380px; margin: 0 auto; }

/\* google search \*/

#goog-fixurl ul { list-style: none; padding: 0; margin: 0; }

#goog-fixurl form { margin: 0; }

#goog-wm-qt, #goog-wm-sb { border: 1px solid #bbb; font-size: 16px; line-height: normal; vertical-align: top; color: #444; border-radius: 2px; }

#goog-wm-qt { width: 220px; height: 20px; padding: 5px; margin: 5px 10px 0 0; box-shadow: inset 0 1px 1px #ccc; }

#goog-wm-sb { display: inline-block; height: 32px; padding: 0 10px; margin: 5px 0 0; white-space: nowrap; cursor: pointer; background-color: #f5f5f5; background-image: -webkit-linear-gradient(rgba(255,255,255,0), #f1f1f1); background-image: -moz-linear-gradient(rgba(255,255,255,0), #f1f1f1); background-image: -ms-linear-gradient(rgba(255,255,255,0), #f1f1f1); background-image: -o-linear-gradient(rgba(255,255,255,0), #f1f1f1); -webkit-appearance: none; -moz-appearance: none; appearance: none; \*overflow: visible; \*display: inline; \*zoom: 1; }

#goog-wm-sb:hover, #goog-wm-sb:focus { border-color: #aaa; box-shadow: 0 1px 1px rgba(0, 0, 0, 0.1); background-color: #f8f8f8; }

#goog-wm-qt:focus, #goog-wm-sb:focus { border-color: #105cb6; outline: 0; color: #222; }

input::-moz-focus-inner { padding: 0; border: 0; }

</style>

</head>

<body>

<div class="container">

<h1>Not found <span>:(</span></h1>

<p>Sorry, but the page you were trying to view does not exist.</p>

<p>It looks like this was the result of either:</p>

<ul>

<li>a mistyped address</li>

<li>an out-of-date link</li>

</ul>

<script>

var GOOG\_FIXURL\_LANG = (navigator.language || '').slice(0,2),GOOG\_FIXURL\_SITE = location.host;

</script>

<script src="http://linkhelp.clients.google.com/tbproxy/lh/wm/fixurl.js"></script>

</div>

## CSS

### default.css

body{

background:#eff3f6;

}

header {

padding: 2px;

margin-bottom: 2px;

}

header img.logo {

border-radius: 35px;

-webkit-border-radius: 35px;

-moz-border-radius: 35px;

margin-left: 10px;

margin-top: 3px;

margin-bottom: 3px;

}

.box

{

background:#fefefe;

border: 1px solid #C3D4DB;

border-top:1px;

-webkit-border-radius:5px;

-moz-border-radius:5px;

border-radius:5px;

-moz-box-shadow:rgba(0,0,0,0.15) 0 0 1px;

-webkit-box-shadow:rgba(0,0,0,0.15) 0 0 1px;

box-shadow:rgba(0,0,0,0.15) 0 0 1px;

color:#444;

font:normal 12px/14px Arial, Helvetica, Sans-serif;

margin:0 auto 30px;

overflow:hidden;

}

.box.login

{

height:260px;

width:332px;

position:absolute;

left:50%;

top:50%;

margin:-130px 0 0 -166px;

}

.box.passwdreminder {

height: 260px;

left: 50%;

margin: -130px 0 0 -166px;

position: absolute;

top: 50%;

width: 332px;

}

.box.passwdreminder p {

margin-bottom: 23px;

margin-left: 10px;

margin-right: 10px;

text-align: center;

font-weight: bold;

padding-top: 15px;

}

.box.resetpassword {

height: 360px;

left: 50%;

margin: -200px 0 0 -166px;

position: absolute;

top: 50%;

width: 332px;

}

.box.resetpassword p {

margin-bottom: 57px;

margin-left: 10px;

margin-right: 10px;

text-align: center;

font-weight: bold;

padding-top: 15px;

}

.boxBody

{

background:#fefefe;

border-top:1px solid #dde0e8;

border-bottom:1px solid #dde0e8;

padding:10px 20px;

}

.box footer

{

background:#eff4f6;

border-top:1px solid #fff;

padding:22px 26px;

overflow:hidden;

height:32px;

}

.box label

{

display:block;

font:14px/22px Arial, Helvetica, Sans-serif;

margin:10px 0 0 6px;

}

.box footer label{

float:left;

margin:4px 0 0;

}

.box footer input[type=checkbox]{

vertical-align:sub;

\*vertical-align:middle;

margin-right:10px;

}

.box input[type=text],

.box input[type=password],

.txtField,

.cjComboBox

{

border:6px solid #F7F9FA;

-webkit-border-radius:3px;

-moz-border-radius:3px;

border-radius:3px;

-moz-box-shadow:2px 3px 3px rgba(0, 0, 0, 0.06) inset, 0 0 1px #95a2a7 inset;

-webkit-box-shadow:2px 3px 3px rgba(0, 0, 0, 0.06) inset, 0 0 1px #95a2a7 inset;

box-shadow:2px 3px 3px rgba(0, 0, 0, 0.06) inset, 0 0 1px #95a2a7 inset;

margin:3px 0 4px;

padding:8px 6px;

width:270px;

display:block;

}

.box input[type=text]:focus,

.box input[type=password]:focus,

.txtField:focus,

.cjComboBox:focus

{

border:6px solid #f0f7fc;

-moz-box-shadow:2px 3px 3px rgba(0, 0, 0, 0.04) inset, 0 0 1px #0d6db6 inset;

-webkit-box-shadow:2px 3px 3px rgba(0, 0, 0, 0.04) inset, 0 0 1px #0d6db6 inset;

box-shadow:2px 3px 3px rgba(0, 0, 0, 0.04) inset, 0 0 1px #0d6db6 inset;

color:#333;

}

.cjComboBox

{

width:294px;

}

.cjComboBox.small

{

padding:3px 2px 3px 6px;

width:100px;

border-width:3px !important;

}

.txtField.small

{

padding:3px 6px;

width:200px;

border-width:3px !important;

}

.rLink{padding:0 6px 0 0; font-size:11px; float:right;}

.box a{color:#999;}

.box a:hover, .box a:focus{text-decoration:underline;}

.box a:active{color:#f84747;}

.btnLogin

{

-moz-border-radius:2px;

-webkit-border-radius:2px;

border-radius:15px;

background:#a1d8f0;

background:-moz-linear-gradient(top, #badff3, #7acbed);

background:-webkit-gradient(linear, left top, left bottom, from(#badff3), to(#7acbed));

-ms-filter: "progid:DXImageTransform.Microsoft.gradient(startColorStr='#badff3', EndColorStr='#7acbed')";

border:1px solid #7db0cc !important;

cursor: pointer;

padding:11px 16px;

font:bold 11px/14px Verdana, Tahomma, Geneva;

text-shadow:rgba(0,0,0,0.2) 0 1px 0px;

color:#fff;

-moz-box-shadow:inset rgba(255,255,255,0.6) 0 1px 1px, rgba(0,0,0,0.1) 0 1px 1px;

-webkit-box-shadow:inset rgba(255,255,255,0.6) 0 1px 1px, rgba(0,0,0,0.1) 0 1px 1px;

box-shadow:inset rgba(255,255,255,0.6) 0 1px 1px, rgba(0,0,0,0.1) 0 1px 1px;

margin-left:12px;

float:right;

padding:7px 21px;

}

.btnLogin:hover,

.btnLogin:focus,

.btnLogin:active{

background:#a1d8f0;

background:-moz-linear-gradient(top, #7acbed, #badff3);

background:-webkit-gradient(linear, left top, left bottom, from(#7acbed), to(#badff3));

-ms-filter: "progid:DXImageTransform.Microsoft.gradient(startColorStr='#7acbed', EndColorStr='#badff3')";

}

.btnLogin:active

{

text-shadow:rgba(0,0,0,0.3) 0 -1px 0px;

}

footer#main{

position:fixed;

left:0;

bottom:10px;

text-align:center;

font:normal 11px/16px Arial, Helvetica, sans-serif;

width:100%;

}

#headerText {

height: 70px;

position:relative;

}

#headerText div {

position:absolute;

top:45%;

}

#logout {

right: 0;

color: lightblue;

cursor: pointer;

}

body, ul, li {

font-size:14px;

font-family:Arial, Helvetica, sans-serif;

line-height:21px;

text-align:left;

}

## Dictionaries

### hungary.dic

[header]

domain = "etraffic.hu"

[administrator]

home = "Home"

selectUploadedFile = "Válassza ki a feltöltendő file típusát!"

stg\_10\_fonel\_nagyobb\_szervezetek = "10 főnél nagyobb szervezetek"

stg\_belfoldi\_osszes\_jovedelem = " belföldi összes jövedelem"

stg\_cegautok = "cégautók"

stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak = "EÜ szociális ellátottak és elvándoroltak"

stg\_egyeb\_intezmenyi\_ellatottsag\_sulya = "Egyéb intézményi ellátottság"

stg\_elelmiszeruzletek\_es\_aruhazak = "Elélmiszerüzletek és áruházak"

stg\_erintett\_nepesseg\_szama = "Érintett népesség száma"

stg\_ertekesites\_netto\_arbevetele = "Értékesítés nettó árbevétele"

stg\_foglalkoztatottak\_szama = "Foglalkoztatottak száma"

stg\_jovedelem\_es\_allaskeresok\_szama = "Jövedelem és álláskeresők száma"

stg\_jovedelmi\_helyzet = "Jövedelmi helyzet"

stg\_keresztmetszeti\_forgalom = "Keresztmetszetti forgalom"

stg\_keresztmetszeti\_forgalom\_v1 = "Keresztmetszetti forgalom v1"

stg\_kollegium = "Kollégium"

stg\_kultura\_es\_sport = "Kultúra és sport"

stg\_kulturalis\_funkciok\_aranya = "Kultúrásli funkciók aránya"

stg\_lakonepesseg = "Lakónépesség"

stg\_mas\_telepulesrol\_bejaro\_altalanos\_es\_kozepiskolasok = "Más településről bejáró ált. iskolások száma"

stg\_mukodo\_vallalkozasok\_szama = "Működő vállalkozások száma"

stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban = "Működő vállalkozások száma kiemelt iparágakban"

stg\_munkakepes\_koru\_nepesseg = "Munkaképes népesség"

stg\_nemzetkozi\_tranzitforgalom = "Nemzetközi tranzitforgalom"

stg\_node\_attribs = "Node attribs"

stg\_nodes = "Nodes"

stg\_oktatasi\_ferohelyek\_szama = "Oktatási férőhelyek száma"

stg\_szabad\_iskolavalasztas\_valtozo = "Szabad iskolaválasztás változó"

stg\_szallashely = "Szálláshely"

stg\_szemelygepkocsik = "Személygépkocsik"

stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese = "Tanulási célú utazások jellemzés"

stg\_tanulok\_es\_hallgatok\_szama = "Tanulók és hallgatók száma"

stg\_telepules = "Település"

stg\_telepules\_koordinata = "Település koordináta"

stg\_tomegkozlekedes = "Tömegközlekedés"

stg\_ugyintezes\_altalanos\_jellemzese = "Ügyintézés általános jellemzése"

stg\_uzletek\_vendeglatohelyek\_gyogyszertarak = "Üzletek, vendéglátóhelyek, gyógyszertárak"

stg\_vandorlasi\_egyenleg\_aranya = "Vándorlási egyenleg aránya"

batchUpload = "Kötegelt feltöltés"

singleUpload = "Egyedi feltöltés"

upload = "Feltöltés"

[footer]

example = “példa”

## PHP

### Index.php

<?php

/\*

TODO: write header information here

\*/

/\*

Requires

\*/

require\_once("../resources/config.php");

if (function\_exists('mb\_internal\_charset')) {

mb\_internal\_charset('UTF-8');

}

define('SMARTY\_RESOURCE\_CHAR\_SET', 'UTF-8');

require\_once(LIBRARY\_PATH.'/Smarty-3.1.12/libs/Smarty.class.php');

/\*

INIT

\*/

$smarty = new Smarty;

//$smarty->force\_compile = true;

$smarty->debugging = false;

$smarty->caching = false;

$smarty->cache\_lifetime = 120;

$dictionary = "hungary";

$smarty->configLoad(DICTIONARIES\_PATH."/$dictionary.dic");

// setup the autoloading

require\_once(LIBRARY\_PATH.'/propel/vendor/autoload.php');

// setup Propel

require\_once(LIBRARY\_PATH.'/etraffic/orm/generated-conf/config.php');

require\_once(LIBRARY\_PATH.'/etraffic/repository/repository.php');

session\_start();

/\*

unset($\_SESSION['SUBSCRIBER']);

unset($\_SESSION['USER']);

unset($\_SESSION['ADMIN']);

\*/

//$\_SESSION['ADMIN'] = new Admin();

//$\_SESSION['SUBSCRIBER'] = new Subscriber();

//$\_SESSION['USER'] = Repository::getInstance()->getUserById(3);

/\*

unset($\_SESSION['SUBSCRIBER']);

unset($\_SESSION['USER']);

unset($\_SESSION['ADMIN']);

$\_SESSION['ADMIN'] = new Admin();

$\_SESSION['SUBSCRIBER'] = new Subscriber();

$\_SESSION['USER'] = Repository::getInstance()->getUserById(4);

\*/

//require\_once('init.php');

?>

<!doctype html>

<!-- paulirish.com/2008/conditional-stylesheets-vs-css-hacks-answer-neither/ -->

<!--[if lt IE 7]> <html class="no-js lt-ie9 lt-ie8 lt-ie7" lang="en"> <![endif]-->

<!--[if IE 7]> <html class="no-js lt-ie9 lt-ie8" lang="en"> <![endif]-->

<!--[if IE 8]> <html class="no-js lt-ie9" lang="en"> <![endif]-->

<!-- Consider adding a manifest.appcache: h5bp.com/d/Offline -->

<!--[if gt IE 8]><!--> <html class="no-js" lang="en"> <!--<![endif]-->

<head>

<meta charset="utf-8">

<!-- Use the .htaccess and remove these lines to avoid edge case issues.

More info: h5bp.com/i/378 -->

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<title>eTraffic</title>

<meta name="description" content="">

<!-- Mobile viewport optimized: h5bp.com/viewport -->

<meta name="viewport" content="width=device-width">

<!-- Place favicon.ico and apple-touch-icon.png in the root directory: mathiasbynens.be/notes/touch-icons -->

<!--

<link type="text/css" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-ui.css" rel="Stylesheet" />

<link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.css">

-->

<link type="text/css" href="js/libs/jquery/jquery-ui-1.10.4.css" rel="Stylesheet" />

<link rel="stylesheet" href="css/style.css">

<link rel="stylesheet" href="css/960.css">

<link rel="stylesheet" href="css/default.css">

<link rel="stylesheet" href="css/style\_leaflet.css">

<link rel="stylesheet" href="css/leaflet.css">

<!-- More ideas for your <head> here: h5bp.com/d/head-Tips -->

<!-- All JavaScript at the bottom, except this Modernizr build.

Modernizr enables HTML5 elements & feature detects for optimal performance.

Create your own custom Modernizr build: www.modernizr.com/download/ -->

<script src="js/libs/modernizr-2.5.3.min.js"></script>

</head>

<body>

<!-- Prompt IE 6 users to install Chrome Frame. Remove this if you support IE 6.

chromium.org/developers/how-tos/chrome-frame-getting-started -->

<!--[if lt IE 7]><p class=chromeframe>Your browser is <em>ancient!</em> <a href="http://browsehappy.com/">Upgrade to a different browser</a> or <a href="http://www.google.com/chromeframe/?redirect=true">install Google Chrome Frame</a> to experience this site.</p><![endif]-->

<header>

<?php

$smarty->configLoad(DICTIONARIES\_PATH."/$dictionary.dic", "header");

$loggedin = isset($\_SESSION['USER']) || isset($\_SESSION['SUBSCRIBER']) || isset($\_SESSION['ADMIN']) ; $smarty->assign('loggedin', $loggedin);

$smarty->display(TEMPLATES\_PATH.'/header.tpl');

?>

</header>

<div role="main">

<div id="uploadView" class="container\_16">

<?php

$smarty->configLoad(DICTIONARIES\_PATH."/$dictionary.dic", "administrator");

$smarty->display(TEMPLATES\_PATH.'/admin.tpl');

?>

</div>

</div>

<footer class="container\_16">

<?php

$smarty->configLoad(DICTIONARIES\_PATH."/$dictionary.dic", "footer");

$smarty->display(TEMPLATES\_PATH.'/footer.tpl');

?>

</footer>

<!-- JavaScript at the bottom for fast page loading -->

<script src="js/libs/jquery/jquery-1.11.1.js"></script>

<script src="js/libs/jquery/jquery-ui-1.10.4.js"></script>

<script src="js/libs/knockout-3.0.0.js"></script>

<script src="js/libs/knockout-jqueryui.js"></script>

<script src="js/libs/jquery/jquery.validate.min.js"></script>

<script src="js/libs/highcharts/highcharts.js"></script>

<script src="js/libs/highcharts/modules/exporting.js"></script>

<script src="js/libs/leaflet.js"></script>

<script src="js/libs/Leaflet.Editable.js"></script>

<script src="js/libs/L.Control.MousePosition.js"></script>

<script src="js/libs/proj4.js"></script>

<!-- scripts concatenated and minified via build script -->

<script src="js/plugins.js"></script>

<script src="js/utils.js"></script>

<script src="js/constant.js"></script>

<script src="js/model.js"></script>

<script src="js/uploadvm.js"></script>

<!--

<script src="js/viewmodel.js"></script>

-->

<?php

echo "<script src='js/jsrepository.js'></script>";

?>

<script src="js/script.js"></script>

<!-- end scripts -->

<!-- Asynchronous Google Analytics snippet. Change UA-XXXXX-X to be your site's ID.

mathiasbynens.be/notes/async-analytics-snippet -->

<script>

/\*

var \_gaq=[['\_setAccount','UA-XXXXX-X'],['\_trackPageview']];

(function(d,t){var g=d.createElement(t),s=d.getElementsByTagName(t)[0];

g.src=('https:'==location.protocol?'//ssl':'//www')+'.google-analytics.com/ga.js';

s.parentNode.insertBefore(g,s)}(document,'script'));

\*/

</script>

</body>

</html>

## Template

### resources/templates

### footer.tpl

<div class="container\_16">

footer

</div>

### header.tpl

<div class="container\_16 ui-widget-header">

<div class='grid\_4 alpha'>

<img class='logo' src='img/logo.png' alt='logo' />

</div>

<div id="headerText" class='grid\_12 omega'>

<div id="sitename">eTraffic</div>

{if $loggedin}

<div id="logout">logout</div>

{/if}

</div>

</div>

### admin.tpl

<div id="tabs">

<ul>

<li><a href="#tabs-1">{#singleUpload#}</a></li>

<li><a href="#tabs-2">{#batchUpload#}</a></li>

</ul>

<div id="tabs-1">

{include file="file:{$smarty.const.TEMPLATES\_PATH}/admin/singleUpload.tpl" scope=parent}

</div>

<div id="tabs-2">

{include file="file:{$smarty.const.TEMPLATES\_PATH}/admin/batchUpload.tpl" scope=parent}

</div>

</div>

### admin/batchUpload.tpl

<div class="ui-widget">

<div data-bind='foreach: { data: fileTypes }'>

<div>

<label class="grid\_8" data-bind="text: displayName"></label>

<input type="file" data-bind="attr: { id: id, name: name } "/>

</div>

</div>

</div>

### admin/singleUpload.tpl

<div>

<label>{#selectUploadedFile#}</label>

<select data-bind="options: fileTypes, optionsText: 'displayName', value: selectedFileType" ></select>

</div>

<div>

<input type="file" id="singleupload" name="singleupload" >

</div>

<div>

<input type="submit" data-bind="click: uploadSingle"/>

</div>

## JS

### model.js

function FileType(data) {

"use strict";

if (typeof data === 'undefined') {

var data = null;

}

this.id = data !== null && data.id ? ko.observable(data.id) : ko.observable(null); //;

this.name = data !== null && data.name ? ko.observable(data.name) : ko.observable(null); //;

this.displayName = data !== null && data.displayName ? ko.observable(data.displayName) : ko.observable(null); //;

}

### script.js

/\* Author:

\*/

$(document).ready(function() {

var viewModel = new UploadViewModel();

ko.applyBindings(viewModel, document.getElementById("uploadView"));

$( "#tabs" ).tabs();

});

### uploadvm.js

function UploadViewModel(prepository, pparent) {

var self = this;

this.fileTypes = ko.observableArray();

this.selectedFileType = ko.observable();

this.name = ko.observable();

this.fileData = ko.observable();

this.uploadSingle = function() {

if (document.getElementById('singleupload').files.length == 0) {

alert('Kérem válasszon ki egy feltöltendő MS EXCEL file-t!');

return;

}

var form\_data = new FormData();

form\_data.append('singleupload', document.getElementById('singleupload').files[0]);

form\_data.append('fileType', ko.toJSON(self.selectedFileType()));

$.ajax({

url: 'restful/DS/singleUpload', // point to server-side PHP script

dataType: 'text', // what to expect back from the PHP script, if anything

cache: false,

contentType: false,

processData: false,

data: form\_data,

type: 'POST',

success: function(php\_script\_response){

alert(php\_script\_response); // display response from the PHP script, if any

},

error: function(xhr, status, error){

alert(xhr.responseText);

}

});

}

this.batchUpload = function() {

var form\_data = new FormData();

var error = false;

ko.utils.arrayForEach(self.fileTypes(), function(item) {

if (document.getElementById(item.id()).files[0] == undefined) {

error = true;

}

form\_data.append('files['+item.id()+']', document.getElementById(item.id()).files[0]);

// console.log(pitem);

});

if (error) {

alert('Kérem minden állományt töltsön fel!');

return;

}

$.ajax({

url: 'restful/DS/batchUpload', // point to server-side PHP script

dataType: 'text', // what to expect back from the PHP script, if anything

cache: false,

contentType: false,

processData: false,

data: form\_data,

type: 'POST',

success: function(php\_script\_response){

alert(php\_script\_response); // display response from the PHP script, if any

},

error: function(xhr, status, error){

alert(xhr.responseText);

}

});

}

this.init = function() {

this.fileTypes.push(new FileType({id: 1, name: 'stg\_10\_fonel\_nagyobb\_szervezetek', displayName: "10 főnél nagyobb szervezetek"}));

this.fileTypes.push(new FileType({id: 2, name: 'stg\_belfoldi\_osszes\_jovedelem', displayName: "belföldi összes jövedelem"}));

this.fileTypes.push(new FileType({id: 3, name: 'stg\_cegautok', displayName: "cégautók"}));

this.fileTypes.push(new FileType({id: 4, name: 'stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak', displayName: "EÜ szociális ellátottak és elvándoroltak"}));

this.fileTypes.push(new FileType({id: 5, name: 'stg\_egyeb\_intezmenyi\_ellatottsag\_sulya', displayName: "Egyéb intézményi ellátottság"}));

this.fileTypes.push(new FileType({id: 6, name: 'stg\_elelmiszeruzletek\_es\_aruhazak', displayName: "Elélmiszerüzletek és áruházak"}));

this.fileTypes.push(new FileType({id: 7, name: 'stg\_erintett\_nepesseg\_szama', displayName: "Érintett népesség száma"}));

this.fileTypes.push(new FileType({id: 8, name: 'stg\_ertekesites\_netto\_arbevetele', displayName: "Értékesítés nettó árbevétele"}));

this.fileTypes.push(new FileType({id: 9, name: 'stg\_foglalkoztatottak\_szama', displayName: "Foglalkoztatottak száma"}));

this.fileTypes.push(new FileType({id: 10, name: 'stg\_jovedelem\_es\_allaskeresok\_szama', displayName: "Jövedelem és álláskeresők száma"}));

this.fileTypes.push(new FileType({id: 11, name: 'stg\_jovedelmi\_helyzet', displayName: "Jövedelmi helyzet"}));

this.fileTypes.push(new FileType({id: 12, name: 'stg\_keresztmetszeti\_forgalom', displayName: "Keresztmetszetti forgalom"}));

this.fileTypes.push(new FileType({id: 13, name: 'stg\_keresztmetszeti\_forgalom\_v1', displayName: "Keresztmetszetti forgalom v1"}));

this.fileTypes.push(new FileType({id: 14, name: 'stg\_kollegium', displayName: "Kollégium"}));

this.fileTypes.push(new FileType({id: 15, name: 'stg\_kultura\_es\_sport', displayName: "Kultúra és sport"}));

this.fileTypes.push(new FileType({id: 16, name: 'stg\_kulturalis\_funkciok\_aranya', displayName: "Kultúrásli funkciók aránya"}));

this.fileTypes.push(new FileType({id: 17, name: 'stg\_lakonepesseg', displayName: "Lakónépesség"}));

this.fileTypes.push(new FileType({id: 18, name: 'stg\_mas\_telepulesrol\_bejaro\_altalanos\_es\_kozepiskolasok', displayName: "Más településről bejáró ált. iskolások száma"}));

this.fileTypes.push(new FileType({id: 19, name: 'stg\_mukodo\_vallalkozasok\_szama', displayName: "Működő vállalkozások száma"}));

this.fileTypes.push(new FileType({id: 20, name: 'stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban', displayName: "Működő vállalkozások száma kiemelt iparágakban"}));

this.fileTypes.push(new FileType({id: 21, name: 'stg\_munkakepes\_koru\_nepesseg', displayName: "Munkaképes népesség"}));

this.fileTypes.push(new FileType({id: 22, name: 'stg\_nemzetkozi\_tranzitforgalom', displayName: "Nemzetközi tranzitforgalom"}));

this.fileTypes.push(new FileType({id: 23, name: 'stg\_node\_attribs', displayName: "Node attribs"}));

this.fileTypes.push(new FileType({id: 24, name: 'stg\_nodes', displayName: "Nodes"}));

this.fileTypes.push(new FileType({id: 25, name: 'stg\_oktatasi\_ferohelyek\_szama', displayName: "Oktatási férőhelyek száma"}));

this.fileTypes.push(new FileType({id: 26, name: 'stg\_szabad\_iskolavalasztas\_valtozo', displayName: "Szabad iskolaválasztás változó"}));

this.fileTypes.push(new FileType({id: 27, name: 'stg\_szallashely', displayName: "Szálláshely"}));

this.fileTypes.push(new FileType({id: 28, name: 'stg\_szemelygepkocsik', displayName: "Személygépkocsik"}));

this.fileTypes.push(new FileType({id: 29, name: 'stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese', displayName: "Tanulási célú utazások jellemzés"}));

this.fileTypes.push(new FileType({id: 30, name: 'stg\_tanulok\_es\_hallgatok\_szama', displayName: "Tanulók és hallgatók száma"}));

this.fileTypes.push(new FileType({id: 31, name: 'stg\_telepules', displayName: "Település"}));

this.fileTypes.push(new FileType({id: 32, name: 'stg\_telepules\_koordinata', displayName: "Település koordináta"}));

this.fileTypes.push(new FileType({id: 33, name: 'stg\_tomegkozlekedes', displayName: "Tömegközlekedés"}));

this.fileTypes.push(new FileType({id: 34, name: 'stg\_ugyintezes\_altalanos\_jellemzese', displayName: "Ügyintézés általános jellemzése"}));

this.fileTypes.push(new FileType({id: 35, name: 'stg\_uzletek\_vendeglatohelyek\_gyogyszertarak', displayName: "Üzletek, vendéglátóhelyek, gyógyszertárak"}));

this.fileTypes.push(new FileType({id: 36, name: 'stg\_vandorlasi\_egyenleg\_aranya', displayName: "Vándorlási egyenleg aránya"}));

}

this.init();

}

### utils.js

function keysToLowerCase (obj) {

var key, keys = Object.keys(obj);

var n = keys.length;

var newobj={}

while (n--) {

key = keys[n];

newobj[key.toLowerCase()] = obj[key];

}

return newobj;

}

function clone(obj) {

// Handle the 3 simple types, and null or undefined

if (null == obj || "object" != typeof obj) return obj;

// Handle Date

if (obj instanceof Date) {

var copy = new Date();

copy.setTime(obj.getTime());

return copy;

}

// Handle Array

if (obj instanceof Array) {

var copy = [];

for (var i = 0, len = obj.length; i < len; i++) {

copy[i] = clone(obj[i]);

}

return copy;

}

// Handle Object

if (obj instanceof Object) {

var copy = {};

for (var attr in obj) {

if (obj.hasOwnProperty(attr)) copy[attr] = clone(obj[attr]);

}

return copy;

}

throw new Error("Unable to copy obj! Its type isn't supported.");

}

function print(obj) {

// Handle the 3 simple types, and null or undefined

if (null == obj || "object" != typeof obj) console.log(obj);

// Handle Date

if (obj instanceof Date) {

console.log(obj);

}

// Handle Array

if (obj instanceof Array) {

var copy = [];

for (var i = 0, len = obj.length; i < len; i++) {

console.log(obj[i]);

}

}

// Handle Object

if (obj instanceof Object) {

var copy = {};

for (var attr in obj) {

console.log(obj[attr]);

}

}

}

function loadHighcharts () {

/\*

var chart = $('#highchartsContainer').highcharts({

title: {

text: 'Makrogazdasági mutatók',

x: -20 //center

},

subtitle: {

text: 'eTraffic.hu',

x: -20

}

xAxis: {

categories: ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',

'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

},

yAxis: {

plotLines: [{

value: 0,

width: 1,

color: '#808080'

}]

},

legend: {

layout: 'vertical',

align: 'right',

verticalAlign: 'middle',

borderWidth: 0

},

series: [{

name: 'Tokyo',

data: [7.0, 6.9, 9.5, 14.5, 18.2, 21.5, 25.2, 26.5, 23.3, 18.3, 13.9, 9.6]

}, {

name: 'New York',

data: [-0.2, 0.8, 5.7, 11.3, 17.0, 22.0, 24.8, 24.1, 20.1, 14.1, 8.6, 2.5]

}, {

name: 'Berlin',

data: [-0.9, 0.6, 3.5, 8.4, 13.5, 17.0, 18.6, 17.9, 14.3, 9.0, 3.9, 1.0]

}, {

name: 'London',

data: [3.9, 4.2, 5.7, 8.5, 11.9, 15.2, 17.0, 16.6, 14.2, 10.3, 6.6, 4.8]

}]

});

\*/

/\*

setInterval(function(){

var chart = $('#highchartsContainer').highcharts();

chart.addSeries({

data: [194.1, 95.6, 54.4, 29.9, 71.5, 106.4, 129.2, 144.0, 176.0, 135.6, 148.5, 216.4]

});

}, 3000);

\*/

}

# Kommunikációs réteg

## Restful

public\_html/restful

### dispatch.php

<?php

require\_once("../../resources/config.php");

require\_once(LIBRARY\_PATH.'/Slim/Slim.php');

// setup the autoloading

require\_once(LIBRARY\_PATH.'/propel/vendor/autoload.php');

// setup Propel

require\_once(LIBRARY\_PATH.'/etraffic/orm/generated-conf/config.php');

\Slim\Slim::registerAutoloader();

session\_start();

require\_once(LIBRARY\_PATH.'/etraffic/repository/repository.php');

$app = new \Slim\Slim();

require 'app/routes/uploadServices.php';

$app->run();

?>

### app/routes/uploadServices.php

<?php

require\_once("../../resources/config.php");

/\*\* PHPExcel \*/

include\_once( LIBRARY\_PATH.'/PHPExcel/Classes/PHPExcel.php' );

//require\_once '../Build/PHPExcel.phar';

require\_once(LIBRARY\_PATH.'/etraffic/repository/repository.php');

require\_once(LIBRARY\_PATH.'/etraffic/utils/utils.php');

$app->get('/DS/teszt', 'teszt');

$app->post('/DS/singleUpload', 'singleUpload');

$app->post('/DS/batchUpload', 'batchUpload');

function batchUpload() {

global $app;

$errors = array();

if (count($\_FILES)) {

$files = rearrange($\_FILES['files']);

$errors = array();

foreach ($files as $k=>$f) {

$e = checExcelFile($k, $f['name'], $f['tmp\_name']);

if ($e != null) {

$errors[] = array("type"=>$k, "name"=>$f['name'], "error"=>$e);

}

}

if (count($errors) == 0) {

$res = array();

foreach ($files as $k=>$f) {

$r = uploadFile($k, $f['name'], $f['tmp\_name']);

$res[] = array("type"=>$k, "name"=>$f['name'], "result"=>$r);

}

echo json\_encode($res);

}

else {

$app->halt(500, json\_encode($errors));

}

} else {

$app->halt(500, "No file(s) uploaded");

}

}

function singleUpload() {

global $app;

if(isset($\_FILES['singleupload'])){

if($\_FILES['singleupload']['tmp\_name']){

if(!$\_FILES['singleupload']['error'])

{

$arraydata = array();

$inputFile = $\_FILES['singleupload']['tmp\_name'];

$extension = strtoupper(pathinfo($\_FILES['singleupload']['name'], PATHINFO\_EXTENSION));

if($extension == 'XLSX' || $extension == 'ODS'){

//Read spreadsheeet workbook

try {

$inputFileType = PHPExcel\_IOFactory::identify($inputFile);

$objReader = PHPExcel\_IOFactory::createReader($inputFileType);

$objReader->setReadDataOnly(true);

$objPHPExcel = $objReader->load($inputFile);

$objWorksheet = $objPHPExcel->setActiveSheetIndex(0);

$highestRow = $objWorksheet->getHighestRow();

$highestColumn = $objWorksheet->getHighestColumn();

$highestColumnIndex = PHPExcel\_Cell::columnIndexFromString($highestColumn);

$p = json\_decode($\_POST['fileType']);

if (Repository::getInstance()->checkFileType($p->id, $highestColumnIndex)) {

//skip the first header row

for ($row = 2; $row <= $highestRow;++$row)

{

$arraydata[$row-1] = array();

for ($col = 0; $col <$highestColumnIndex;++$col)

{

$value=$objWorksheet->getCellByColumnAndRow($col, $row)->getValue();

$arraydata[$row-1][$col]=$value;

}

}

Repository::getInstance()->uloadRows($p->id, $arraydata);

echo json\_encode(count($arraydata) ." row(s) uploaded");

}

else {

$app->halt(403, "The number of columns does not match with the specification ($highestColumnIndex)");

}

} catch(Exception $e) {

$app->halt(500, "Error: ".$e->getMessage());

}

}

else{

//echo "Please upload an XLSX or ODS file";

$app->halt(500, "Please upload an XLSX or ODS file");

}

}

else{

//echo $\_FILES['spreadsheet']['error'];

$app->halt(500, $\_FILES['spreadsheet']['error']);

}

}

} else {

//echo json\_encode('uploadSingle');

$app->halt(500, 'No file selected');

}

}

function teszt(){

echo json\_encode('hello');

}

?>

### .htaccess

<Files ~ (\.pl)>

Options ExecCGI FollowSymLinks

</Files>

Options +FollowSymLinks +SymLinksIfOwnerMatch

IndexOptions +Charset=UTF-8

RewriteEngine On

# Some hosts may require you to use the `RewriteBase` directive.

# If you need to use the `RewriteBase` directive, it should be the

# absolute physical path to the directory that contains this htaccess file.

#

RewriteBase /~josika/eTraffic\_v1/public\_html/restful

RewriteCond %{REQUEST\_FILENAME} !-s

RewriteCond %{REQUEST\_FILENAME} !-l

RewriteCond %{REQUEST\_FILENAME} !-d

RewriteCond %{REQUEST\_URI} !^/dispatch\.php$

RewriteRule ^.\*$ dispatch.php [NC,L]

# Backend

resources/

## Config

### config.php

<?php

$config = array(

"paths" => array(

"resources" => "../resources/",

"library" => "../resources/library/"

)

);

defined("LIBRARY\_PATH") or define("LIBRARY\_PATH", realpath(dirname(\_\_FILE\_\_) . '/library'));

defined("TEMPLATES\_PATH") or define("TEMPLATES\_PATH", realpath(dirname(\_\_FILE\_\_) . '/templates'));

defined("DICTIONARIES\_PATH") or define("DICTIONARIES\_PATH", realpath(dirname(\_\_FILE\_\_) . '/dictionaries'));

/\*

Error reporting.

\*/

ini\_set("error\_reporting", "true");

error\_reporting(E\_ALL|E\_STRCT);

?>

## library/etraffic/utils

### utils.php

<?php

function rearrange( $arr ){

foreach( $arr as $key => $all ){

foreach( $all as $i => $val ){

$new[$i][$key] = $val;

}

}

return $new;

}

function uploadFile($type, $file, $tmpname) {

try {

$inputFileType = PHPExcel\_IOFactory::identify($tmpname);

$objReader = PHPExcel\_IOFactory::createReader($inputFileType);

$objReader->setReadDataOnly(true);

$objPHPExcel = $objReader->load($tmpname);

$objWorksheet = $objPHPExcel->setActiveSheetIndex(0);

$highestRow = $objWorksheet->getHighestRow();

$highestColumn = $objWorksheet->getHighestColumn();

$highestColumnIndex = PHPExcel\_Cell::columnIndexFromString($highestColumn);

if (Repository::getInstance()->checkFileType($type, $highestColumnIndex)) {

//skip the first header row

for ($row = 2; $row <= $highestRow;++$row)

{

$arraydata[$row-1] = array();

for ($col = 0; $col <$highestColumnIndex;++$col)

{

$value=$objWorksheet->getCellByColumnAndRow($col, $row)->getValue();

$arraydata[$row-1][$col]=$value;

}

}

Repository::getInstance()->uloadRows($type, $arraydata);

return count($arraydata) ." row(s) uploaded";

}

else {

return "The number of columns does not match with the specification ($highestColumnIndex)";

}

} catch(Exception $e) {

return "Error: ".$e->getMessage();

}

}

function checExcelFile($type, $file, $tmpname) {

$inputFile = $tmpname;

$extension = strtoupper(pathinfo($file, PATHINFO\_EXTENSION));

if($extension == 'XLSX' || $extension == 'ODS'){

//Read spreadsheeet workbook

try {

$inputFileType = PHPExcel\_IOFactory::identify($inputFile);

$objReader = PHPExcel\_IOFactory::createReader($inputFileType);

$objReader->setReadDataOnly(true);

$objPHPExcel = $objReader->load($inputFile);

$objWorksheet = $objPHPExcel->setActiveSheetIndex(0);

$highestRow = $objWorksheet->getHighestRow();

$highestColumn = $objWorksheet->getHighestColumn();

$highestColumnIndex = PHPExcel\_Cell::columnIndexFromString($highestColumn);

if (!Repository::getInstance()->checkFileType($type, $highestColumnIndex)) {

return "The number of column does not match";

}

return null;

} catch(Exception $e) {

return "Error: ".$e->getMessage();

}

}

else{

return "Please upload an XLSX or ODS file";

}

}

?>

## Repository

### etraffic/repository/repository.php

<?php

use Propel\Runtime\ActiveQuery\Criteria;

use Map\TrendTableMap;

use Propel\Runtime\Collection\Collection;

class Repository {

private static $\_instance = null;

private function \_\_construct() {

date\_default\_timezone\_set("Europe/Budapest");

}

public function \_\_clone() {

trigger\_error( "Cannot clone instance of Singleton pattern ...", E\_USER\_ERROR );

}

public function \_\_wakeup() {

trigger\_error('Cannot deserialize instance of Singleton pattern ...', E\_USER\_ERROR );

}

public static function getInstance()

{

if( !is\_object(self::$\_instance) )

//or if( is\_null(self::$\_instance) ) or if( self::$\_instance == null )

self::$\_instance = new self;

//or, in PHP 5.3.0

//if (empty(static::$\_instance)) {

// $class = get\_called\_class();

// static::$\_instance = new $class;

//}

return self::$\_instance;

}

public function checkFileType($fileType, $columnNumber) {

switch ($fileType) {

case 1:

return $columnNumber == 4 ? true : false;

break;

case 2:

return $columnNumber == 3 ? true : false;

break;

case 3:

return $columnNumber == 4 ? true : false;

break;

case 4:

return $columnNumber == 11 ? true : false;

break;

case 5:

return $columnNumber == 13 ? true : false;

break;

case 6:

return $columnNumber == 4 ? true : false;

break;

case 7:

return $columnNumber == 15 ? true : false;

break;

case 8:

return $columnNumber == 3 ? true : false;

break;

case 9:

return $columnNumber == 4 ? true : false;

break;

case 10:

return $columnNumber == 5 ? true : false;

break;

case 11:

return $columnNumber == 4 ? true : false;

break;

case 12:

return $columnNumber == 44 ? true : false;

break;

case 13:

return $columnNumber == 11 ? true : false;

break;

case 14:

return $columnNumber == 6 ? true : false;

break;

case 15:

return $columnNumber == 10 ? true : false;

break;

case 16:

return $columnNumber == 5 ? true : false;

break;

case 17:

return $columnNumber == 3 ? true : false;

break;

case 18:

return $columnNumber == 4 ? true : false;

break;

case 19:

return $columnNumber == 3 ? true : false;

break;

case 20:

return $columnNumber == 7 ? true : false;

break;

case 21:

return $columnNumber == 3 ? true : false;

break;

case 22:

return $columnNumber == 3 ? true : false;

break;

case 23:

return $columnNumber == 37 ? true : false;

break;

case 24:

return $columnNumber == 4 ? true : false;

break;

case 25:

return $columnNumber == 7 ? true : false;

break;

case 26:

return $columnNumber == 10 ? true : false;

break;

case 27:

return $columnNumber == 4 ? true : false;

break;

case 28:

return $columnNumber == 3 ? true : false;

break;

case 29:

return $columnNumber == 6 ? true : false;

break;

case 30:

return $columnNumber == 10 ? true : false;

break;

case 31:

return $columnNumber == 11 ? true : false;

break;

case 32:

return $columnNumber == 5 ? true : false;

break;

case 33:

return $columnNumber == 4 ? true : false;

break;

case 34:

return $columnNumber == 15 ? true : false;

break;

case 35:

return $columnNumber == 7 ? true : false;

break;

case 36:

return $columnNumber == 6 ? true : false;

break;

default:

return false;

}

}

public function uloadRows($fileType, $arraydata) {

switch ($fileType) {

case 1:

$this->uploadStg10FonelNagyobbSzervezetek($arraydata);

break;

case 2:

$this->uploadStgBelfoldiOsszesJovedelem($arraydata);

break;

case 3:

$this->uploadStgCegautok($arraydata);

break;

case 4:

$this->uploadStgEgeszsegugyiSzocialisEllatottakElvandoroltak($arraydata);

break;

case 5:

$this->uploadStgEgyebIntezmenyiEllatottsagSulya($arraydata);

break;

case 6:

$this->uploadStgElelmiszeruzletekEsAruhazak($arraydata);

break;

case 7:

$this->uploadStgErintettNepessegSzama($arraydata);

break;

case 8:

$this->uploadStgErtekesitesNettoArbevetele($arraydata);

break;

case 9:

$this->uploadStgFoglalkoztatottakSzama($arraydata);

break;

case 10:

$this->uploadStgJovedelemEsAllaskeresokSzama($arraydata);

break;

case 11:

$this->uploadStgJovedelmiHelyzet($arraydata);

break;

case 12:

$this->uploadStgKeresztmetszetiForgalom($arraydata);

break;

case 13:

$this->uploadStgKeresztmetszetiForgalomV1($arraydata);

break;

case 14:

$this->uploadStgKollegium($arraydata);

break;

case 15:

$this->uploadStgKulturaEsSport($arraydata);

break;

case 16:

$this->uploadStgKulturalisFunkciokAranya($arraydata);

break;

case 17:

$this->uploadStgLakonepesseg($arraydata);

break;

case 18:

$this->uploadStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok($arraydata);

break;

case 19:

$this->uploadStgMukodoVallalkozasokSzama($arraydata);

break;

case 20:

$this->uploadStgMukodoVallalkozasokSzamaAKiemeltIparagakban($arraydata);

break;

case 21:

$this->uploadStgMunkakepesKoruNepesseg($arraydata);

break;

case 22:

$this->uploadStgNemzetkoziTranzitforgalom($arraydata);

break;

case 23:

$this->uploadStgNodeAttribs($arraydata);

break;

case 24:

$this->uploadStgNodes($arraydata);

break;

case 25:

$this->uploadStgOktatasiFerohelyekSzama($arraydata);

break;

case 26:

$this->uploadStgSzabadIskolavalasztasValtozo($arraydata);

break;

case 27:

$this->uploadStgSzallashely($arraydata);

break;

case 28:

$this->uploadStgSzemelygepkocsik($arraydata);

break;

case 29:

$this->uploadStgTanulasiCeluUtazasokAltalanosJellemzese($arraydata);

break;

case 30:

$this->uploadStgTanulokEsHallgatokSzama($arraydata);

break;

case 31:

$this->uploadStgTelepules($arraydata);

break;

case 32:

$this->uploadStgTelepulesKoordinata($arraydata);

break;

case 33:

$this->uploadStgTomegkozlekedes($arraydata);

break;

case 34:

$this->uploadStgUgyintezesAltalanosJellemzese($arraydata);

break;

case 35:

$this->uploadStgUzletekVendeglatohelyekGyogyszertarak($arraydata);

break;

case 36:

$this->uploadStgVandorlasiEgyenlegAranya($arraydata);

break;

default:

return false;

}

}

public function uploadStg10FonelNagyobbSzervezetek($rows) {

foreach($rows as $r) {

$s = new \Stg10FonelNagyobbSzervezetek();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setMukodoVallalatokSzama($r[2]);

$s->setVallalatokAtlagosLetszama($r[3]);

$s->Save();

}

}

public function uploadStgBelfoldiOsszesJovedelem($rows) {

foreach($rows as $r) {

$s = new \StgBelfoldiOsszesJovedelem();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setJovedelem($r[2]);

$s->Save();

}

}

public function uploadStgCegautok($rows) {

foreach($rows as $r) {

$s = new \StgCegautok();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setSzemelygepkocsikSzama($r[2]);

$s->setTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama($r[3]);

$s->Save();

}

}

public function uploadStgEgeszsegugyiSzocialisEllatottakElvandoroltak($rows) {

foreach($rows as $r) {

$s = new \StgEgeszsegugyiSzocialisEllatottakElvandoroltak();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama($r[2]);

$s->setHaziOrvosiEllatasbanRendelesenMegjelentekSzama($r[3]);

$s->setElbocsatottBetegekSzama($r[4]);

$s->setMegjelenesiEsetekSzama($r[5]);

$s->setNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama($r[6]);

$s->setNappaliEllatasbanReszesuloIdoskoruakSzama($r[7]);

$s->setNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama($r[8]);

$s->setNappaliEllatasbanReszesuloSzenvedelybetegekSzama($r[9]);

$s->setOdavandorlasokSzama($r[10]);

$s->Save();

}

}

public function uploadStgEgyebIntezmenyiEllatottsagSulya($rows) {

foreach($rows as $r) {

$s = new \StgEgyebIntezmenyiEllatottsagSulya();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setBankfiokLete($r[2]);

$s->setBenzinkutLete($r[3]);

$s->setKorjegyzosegSzekhelye($r[4]);

$s->setPostahivatalLete($r[5]);

$s->setBirosagUgyeszsegLete($r[6]);

$s->setOkmanyirodaLete($r[7]);

$s->setMegyeiFoldhivatalIlletveKirendeltsegLete($r[8]);

$s->setMunkaugyiKozpontIlletveKirendeltsegLete($r[9]);

$s->setVodafoneUzletLete($r[10]);

$s->setTMobileUzletLete($r[11]);

$s->setTelenorUzletLete($r[12]);

$s->Save();

}

}

public function uploadStgElelmiszeruzletekEsAruhazak($rows) {

foreach($rows as $r) {

$s = new \StgElelmiszeruzletekEsAruhazak();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setKiskereskedelmiUzletekSzama($r[2]);

$s->setElelmiszerVegyesuzletekEsAruhazakSzama($r[3]);

$s->Save();

}

}

public function uploadStgErintettNepessegSzama($rows) {

foreach($rows as $r) {

$s = new \StgErintettNepessegSzama();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setRegisztraltBunelkovetokSzama($r[2]);

$s->setHaztartasiGazfogyasztokSzama($r[3]);

$s->setHaztartasiVillamosenergiaFogyasztokSzama($r[4]);

$s->setElvandorlasokSzama($r[5]);

$s->setAllandoNepessegSzama($r[6]);

$s->setVallalkozasokSzama($r[7]);

$s->setNyilvantartottAllaskeresokSzama($r[8]);

$s->setLakasallomany($r[9]);

$s->setBolcsodebeBeirtGyermekekSzama($r[10]);

$s->setAllandoNepessegbol1859EvesekSzama($r[11]);

$s->setMobilelofizetesekSzama($r[12]);

$s->setOsszesAdofizeto($r[13]);

$s->setOsszesAdo($r[14]);

$s->Save();

}

}

public function uploadStgErtekesitesNettoArbevetele($rows) {

foreach($rows as $r) {

$s = new \StgErtekesitesNettoArbevetele();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setErtekesitesNettoArbevetele($r[2]);

$s->Save();

}

}

public function uploadStgFoglalkoztatottakSzama($rows) {

foreach($rows as $r) {

$s = new \StgFoglalkoztatottakSzama();

$s->setKistersegNev($r[0]);

$s->setKistersegKshkod($r[1]);

$s->setFoglalkoztatottakSzama($r[2]);

$s->setKistersegiFoglalkoztatasiRata($r[3]);

$s->Save();

}

}

public function uploadStgJovedelemEsAllaskeresokSzama($rows) {

foreach($rows as $r) {

$s = new \StgJovedelemEsAllaskeresokSzama();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setOsszesJovedelemFoallasbol($r[2]);

$s->setAllandoNepessegSzama($r[3]);

$s->setP180NaponTuliNyilvantartottAllaskeresokSzama($r[4]);

$s->Save();

}

}

public function uploadStgJovedelmiHelyzet($rows) {

foreach($rows as $r) {

$s = new \StgJovedelmiHelyzet();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setOsszesAdo($r[2]);

$s->setAdofizetokSzama($r[3]);

$s->Save();

}

}

public function uploadStgKeresztmetszetiForgalom($rows) {

foreach($rows as $r) {

$s = new \StgKeresztmetszetiForgalom();

$s->setShapenr($r[0]);

$s->setPartnr($r[1]);

$s->setNrparts($r[2]);

$s->setPointnr($r[3]);

$s->setNrpoints($r[4]);

$s->setX($r[5]);

$s->setY($r[6]);

$s->setMeasure($r[7]);

$s->setKszam($r[8]);

$s->setPkod($r[9]);

$s->setKkod($r[10]);

$s->setVvkod($r[11]);

$s->setKszelv($r[12]);

$s->setVszelv($r[13]);

$s->setRshossz($r[14]);

$s->setAnf($r[15]);

$s->setAnet($r[16]);

$s->setMof($r[17]);

$s->setOngj($r[18]);

$s->setOj($r[19]);

$s->setOmot($r[20]);

$s->setEv($r[21]);

$s->setAsz($r[22]);

$s->setBuszcs($r[23]);

$s->setBusze($r[24]);

$s->setObusz($r[25]);

$s->setNyszer($r[26]);

$s->setPotktgk($r[27]);

$s->setKtgk($r[28]);

$s->setNtgk($r[29]);

$s->setKntgk($r[30]);

$s->setOtgk($r[31]);

$s->setSzgk($r[32]);

$s->setOszgk($r[33]);

$s->setMkp($r[34]);

$s->setKpf($r[35]);

$s->setLassu($r[36]);

$s->setSpec($r[37]);

$s->setFmegb($r[38]);

$s->setAdatforr($r[39]);

$s->setSzamlnap($r[40]);

$s->setJelleg1($r[41]);

$s->setJelleg2($r[42]);

$s->setFmegj($r[43]);

$s->Save();

}

}

public function uploadStgKeresztmetszetiForgalomV1 ($rows) {

foreach($rows as $r) {

$s = new \StgKeresztmetszetiForgalomV1();

$s->setPartnr($r[0]);

$s->setNrparts($r[1]);

$s->setPointnr($r[2]);

$s->setNrpoints($r[3]);

$s->setX($r[4]);

$s->setY($r[5]);

$s->setMeasure($r[6]);

$s->setKezd($r[7]);

$s->setVege($r[8]);

$s->setKszam($r[9]);

$s->setPkod($r[10]);

$s->Save();

}

}

public function uploadStgKollegium($rows) {

foreach($rows as $r) {

$s = new \StgKollegium();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setKollegiumbanLakoNappaliAltTanulokSzama($r[2]);

$s->setKollegiumbanLakoNappaliKozepiskolaiTanulokSzama($r[3]);

$s->setKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama($r[4]);

$s->setKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama($r[5]);

$s->Save();

}

}

public function uploadStgKulturaEsSport($rows) {

foreach($rows as $r) {

$s = new \StgKulturaEsSport();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setFilmszinhaz($r[2]);

$s->setKozmuvelodesiIntezmeny($r[3]);

$s->setMuzeum($r[4]);

$s->setPiac($r[5]);

$s->setSportcsarnokSportpalya($r[6]);

$s->setStrand($r[7]);

$s->setTelepulesiKonyvtar($r[8]);

$s->setUszodaFurdoGyogyfurdo($r[9]);

$s->Save();

}

}

public function uploadStgKulturalisFunkciokAranya($rows) {

foreach($rows as $r) {

$s = new \StgKulturalisFunkciokAranya();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setKulturalisRendezvenyekenResztvevokSzama($r[2]);

$s->setMozilatogatasokSzama($r[3]);

$s->setMuzeumiLatogatokSzama($r[4]);

$s->Save();

}

}

public function uploadStgLakonepesseg($rows) {

foreach($rows as $r) {

$s = new \StgLakonepesseg();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setNepesseg($r[2]);

$s->Save();

}

}

public function uploadStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok($rows) {

foreach($rows as $r) {

$s = new \StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama($r[2]);

$s->setMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama($r[3]);

$s->Save();

}

}

public function uploadStgMukodoVallalkozasokSzama($rows) {

foreach($rows as $r) {

$s = new \StgMukodoVallalkozasokSzama();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setVallalkozasokSzama($r[2]);

$s->Save();

}

}

public function uploadStgMukodoVallalkozasokSzamaAKiemeltIparagakban($rows) {

foreach($rows as $r) {

$s = new \StgMukodoVallalkozasokSzamaAKiemeltIparagakban();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setFeldolgozoiparbanMukodoVallalkozasokSzama($r[2]);

$s->setKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama($r[3]);

$s->setSzallitasbanRaktarozasbanMukodoVallalkozasokSzama($r[4]);

$s->setAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama($r[5]);

$s->setEgyebSzolgaltatasbanMukodoVallalkozasokSzama($r[6]);

$s->Save();

}

}

public function uploadStgMunkakepesKoruNepesseg($rows) {

foreach($rows as $r) {

$s = new \StgMunkakepesKoruNepesseg();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setNepesseg($r[2]);

$s->Save();

}

}

public function uploadStgNemzetkoziTranzitforgalom($rows) {

foreach($rows as $r) {

$s = new \StgNemzetkoziTranzitforgalom();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setHataratlepokSzama($r[2]);

$s->Save();

}

}

public function uploadStgNodeAttribs($rows) {

foreach($rows as $r) {

$s = new \StgNodeAttribs();

$s->setShapeId($r[0]);

$s->setKszam($r[1]);

$s->setPkod($r[2]);

$s->setKkod($r[3]);

$s->setVvkod($r[4]);

$s->setKszelv($r[5]);

$s->setVszelv($r[6]);

$s->setRshossz($r[7]);

$s->setAnf($r[8]);

$s->setAnet($r[9]);

$s->setMof($r[10]);

$s->setOngj($r[11]);

$s->setOj($r[12]);

$s->setOmot($r[13]);

$s->setEv($r[14]);

$s->setAsz($r[15]);

$s->setBuszcs($r[16]);

$s->setBusze($r[17]);

$s->setObusz($r[18]);

$s->setNyszer($r[19]);

$s->setPotktgk($r[20]);

$s->setKtgk($r[21]);

$s->setNtgk($r[22]);

$s->setKntgk($r[23]);

$s->setOtgk($r[24]);

$s->setSzgk($r[25]);

$s->setOszgk($r[26]);

$s->setMkp($r[27]);

$s->setKpf($r[28]);

$s->setLassu($r[29]);

$s->setSpec($r[30]);

$s->setFmegb($r[31]);

$s->setAdatforr($r[32]);

$s->setSzamlnap($r[33]);

$s->setJelleg1($r[34]);

$s->setJelleg2($r[35]);

$s->setFmegj($r[36]);

// print\_r($s->toJSON()); die;

$s->Save();

// die;

}

}

public function uploadStgNodes($rows) {

foreach($rows as $r) {

$s = new \StgNodes();

$s->setShapeId($r[0]);

$s->setX($r[1]);

$s->setY($r[2]);

$s->setNumber($r[3]);

$s->Save();

}

}

public function uploadStgOktatasiFerohelyekSzama($rows) {

foreach($rows as $r) {

$s = new \StgOktatasiFerohelyekSzama();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setNappaliAltIskolaiOsztalyokSzama($r[2]);

$s->setNappaliKozepiskolaiOsztalyokSzama($r[3]);

$s->setNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama($r[4]);

$s->setNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben($r[5]);

$s->setNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben($r[6]);

$s->Save();

}

}

public function uploadStgSzabadIskolavalasztasValtozo($rows) {

foreach($rows as $r) {

$s = new \StgSzabadIskolavalasztasValtozo();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setAllandoNepessegbol613EvesekSzama($r[2]);

$s->setAllandoNepessegbol14EvesekSzama($r[3]);

$s->setAllandoNepessegbol1517EvesekSzama($r[4]);

$s->setNappaliAltalanosIskolaiTanulokSzama($r[5]);

$s->setNappaliKozepiskolaiTanulokSzama($r[6]);

$s->setNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama($r[7]);

$s->setMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama($r[8]);

$s->setMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama($r[9]);

$s->Save();

}

}

public function uploadStgSzallashely($rows) {

foreach($rows as $r) {

$s = new \StgSzallashely();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setFizetovendeglatasSzallashelyeinekSzama($r[2]);

$s->setOsszesKereskedelmiSzallashelyFerohelyeinekSzama($r[3]);

$s->Save();

}

}

public function uploadStgSzemelygepkocsik($rows) {

foreach($rows as $r) {

$s = new \StgSzemelygepkocsik();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setSzemelygepkocsikSzama($r[2]);

$s->Save();

}

}

public function uploadStgTanulasiCeluUtazasokAltalanosJellemzese($rows) {

foreach($rows as $r) {

$s = new \StgTanulasiCeluUtazasokAltalanosJellemzese();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setNappaliAltalanosIskolaiOktatas($r[2]);

$s->setNappaliKozepiskolaiEsFelnottoktatasiOktatas($r[3]);

$s->setFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama($r[4]);

$s->setNappaliEsNemNappaliFelsooktatas($r[5]);

$s->Save();

}

}

public function uploadStgTanulokEsHallgatokSzama($rows) {

foreach($rows as $r) {

$s = new \StgTanulokEsHallgatokSzama();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setNappaliAltIskolaiTanulokSzama($r[2]);

$s->setFelnottoktatasbanAltIskolaiTanulokSzama($r[3]);

$s->setNappaliKozepiskolaiTanulokSzama($r[4]);

$s->setFelnottoktatasbanKozepiskolaiTanulokSzama($r[5]);

$s->setNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama($r[6]);

$s->setFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama($r[7]);

$s->setNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama($r[8]);

$s->setNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama($r[9]);

$s->Save();

}

}

public function uploadStgTelepules($rows) {

foreach($rows as $r) {

$s = new \StgTelepules();

$s->setTelepulesNev($r[0]);

$s->setKshKod($r[1]);

$s->setJogallas($r[2]);

$s->setMegye($r[3]);

$s->setKistersegKod($r[4]);

$s->setKistersegNev($r[5]);

$s->setKistersegSzekhelye($r[6]);

$s->setEovX($r[7]);

$s->setEovY($r[8]);

$s->setWgs84Lat($r[9]);

$s->setWgs84Lob($r[10]);

$s->Save();

}

}

public function uploadStgTelepulesKoordinata($rows) {

foreach($rows as $r) {

$s = new \StgTelepulesKoordinata();

$s->setTelepulesNev($r[0]);

$s->setEovX($r[1]);

$s->setEovY($r[2]);

$s->setWgs84Lat($r[3]);

$s->setWgs84Lob($r[4]);

$s->Save();

}

}

public function uploadStgTomegkozlekedes($rows) {

foreach($rows as $r) {

$s = new \StgTomegkozlekedes();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setVasutallomasLete($r[2]);

$s->setKozvetlenJaratokSzamaNapontaAMegyeszekhelyre($r[3]);

$s->Save();

}

}

public function uploadStgUgyintezesAltalanosJellemzese($rows) {

foreach($rows as $r) {

$s = new \StgUgyintezesAltalanosJellemzese();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setEgeszsegugyiEllatasokSzama($r[2]);

$s->setPostaiGyogyszertariSzolgaltatasokSzama($r[3]);

$s->setBuntetoperekSzama($r[4]);

$s->setKereskedelmiSzolgaltatasokSzama($r[5]);

$s->setSzocialisSzolgaltatasokSzama($r[6]);

$s->setOkmanyirodaiUgyintezesekSzama($r[7]);

$s->setBankokSzama($r[8]);

$s->setBenzinkutakSzama($r[9]);

$s->setKorjegyzosegekFoldhivatalokSzama($r[10]);

$s->setKozmuszolgaltatasok($r[11]);

$s->setMunkaugyiSzolgaltatasokSzama($r[12]);

$s->setBolcsodeiSzolgaltatasokSzama($r[13]);

$s->setMobiltelefonUgyintezesiSzolgaltatasokSzama($r[14]);

$s->Save();

}

}

public function uploadStgUzletekVendeglatohelyekGyogyszertarak($rows) {

foreach($rows as $r) {

$s = new \StgUzletekVendeglatohelyekGyogyszertarak();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setElelmiszerVegyesuzletekAruhazakSzama($r[2]);

$s->setKiskereskedelmiUzletekSzama($r[3]);

$s->setVendeglatohelyekSzama($r[4]);

$s->setFiokgyogyszertarakSzama($r[5]);

$s->setGyogyszertarakSzama($r[6]);

$s->Save();

}

}

public function uploadStgVandorlasiEgyenlegAranya($rows) {

foreach($rows as $r) {

$s = new \StgVandorlasiEgyenlegAranya();

$s->setTelepulesNev($r[0]);

$s->setTelepulesKshkod($r[1]);

$s->setOdavandorlasokSzama($r[2]);

$s->setElvandorlasokSzama($r[3]);

$s->setElveszuletesekSzama($r[4]);

$s->setHalalozasokSzama($r[5]);

$s->Save();

}

}

}

?>

## Entity

### etraffic/orm/generated-classes/Base

#### Stg10FonelNagyobbSzervezetek.php

<?php

namespace Base;

use \Stg10FonelNagyobbSzervezetekQuery as ChildStg10FonelNagyobbSzervezetekQuery;

use \Exception;

use \PDO;

use Map\Stg10FonelNagyobbSzervezetekTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class Stg10FonelNagyobbSzervezetek implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\Stg10FonelNagyobbSzervezetekTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the mukodo\_vallalatok\_szama field.

\* @var int

\*/

protected $mukodo\_vallalatok\_szama;

/\*\*

\* The value for the vallalatok\_atlagos\_letszama field.

\* @var int

\*/

protected $vallalatok\_atlagos\_letszama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\Stg10FonelNagyobbSzervezetek object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>Stg10FonelNagyobbSzervezetek</code> instance. If

\* <code>obj</code> is an instance of <code>Stg10FonelNagyobbSzervezetek</code>, delegates to

\* <code>equals(Stg10FonelNagyobbSzervezetek)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|Stg10FonelNagyobbSzervezetek The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [mukodo\_vallalatok\_szama] column value.

\*

\* @return int

\*/

public function getMukodoVallalatokSzama()

{

return $this->mukodo\_vallalatok\_szama;

}

/\*\*

\* Get the [vallalatok\_atlagos\_letszama] column value.

\*

\* @return int

\*/

public function getVallalatokAtlagosLetszama()

{

return $this->vallalatok\_atlagos\_letszama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : Stg10FonelNagyobbSzervezetekTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : Stg10FonelNagyobbSzervezetekTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : Stg10FonelNagyobbSzervezetekTableMap::translateFieldName('MukodoVallalatokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->mukodo\_vallalatok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : Stg10FonelNagyobbSzervezetekTableMap::translateFieldName('VallalatokAtlagosLetszama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->vallalatok\_atlagos\_letszama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 4; // 4 = Stg10FonelNagyobbSzervezetekTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\Stg10FonelNagyobbSzervezetek'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\Stg10FonelNagyobbSzervezetek The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[Stg10FonelNagyobbSzervezetekTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\Stg10FonelNagyobbSzervezetek The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[Stg10FonelNagyobbSzervezetekTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [mukodo\_vallalatok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\Stg10FonelNagyobbSzervezetek The current object (for fluent API support)

\*/

public function setMukodoVallalatokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->mukodo\_vallalatok\_szama !== $v) {

$this->mukodo\_vallalatok\_szama = $v;

$this->modifiedColumns[Stg10FonelNagyobbSzervezetekTableMap::COL\_MUKODO\_VALLALATOK\_SZAMA] = true;

}

return $this;

} // setMukodoVallalatokSzama()

/\*\*

\* Set the value of [vallalatok\_atlagos\_letszama] column.

\*

\* @param int $v new value

\* @return $this|\Stg10FonelNagyobbSzervezetek The current object (for fluent API support)

\*/

public function setVallalatokAtlagosLetszama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->vallalatok\_atlagos\_letszama !== $v) {

$this->vallalatok\_atlagos\_letszama = $v;

$this->modifiedColumns[Stg10FonelNagyobbSzervezetekTableMap::COL\_VALLALATOK\_ATLAGOS\_LETSZAMA] = true;

}

return $this;

} // setVallalatokAtlagosLetszama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(Stg10FonelNagyobbSzervezetekTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStg10FonelNagyobbSzervezetekQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see Stg10FonelNagyobbSzervezetek::setDeleted()

\* @see Stg10FonelNagyobbSzervezetek::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(Stg10FonelNagyobbSzervezetekTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStg10FonelNagyobbSzervezetekQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(Stg10FonelNagyobbSzervezetekTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

Stg10FonelNagyobbSzervezetekTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(Stg10FonelNagyobbSzervezetekTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(Stg10FonelNagyobbSzervezetekTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(Stg10FonelNagyobbSzervezetekTableMap::COL\_MUKODO\_VALLALATOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'MUKODO\_VALLALATOK\_SZAMA';

}

if ($this->isColumnModified(Stg10FonelNagyobbSzervezetekTableMap::COL\_VALLALATOK\_ATLAGOS\_LETSZAMA)) {

$modifiedColumns[':p' . $index++] = 'VALLALATOK\_ATLAGOS\_LETSZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_10\_fonel\_nagyobb\_szervezetek (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'MUKODO\_VALLALATOK\_SZAMA':

$stmt->bindValue($identifier, $this->mukodo\_vallalatok\_szama, PDO::PARAM\_INT);

break;

case 'VALLALATOK\_ATLAGOS\_LETSZAMA':

$stmt->bindValue($identifier, $this->vallalatok\_atlagos\_letszama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = Stg10FonelNagyobbSzervezetekTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getMukodoVallalatokSzama();

break;

case 3:

return $this->getVallalatokAtlagosLetszama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['Stg10FonelNagyobbSzervezetek'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['Stg10FonelNagyobbSzervezetek'][$this->getPrimaryKey()] = true;

$keys = Stg10FonelNagyobbSzervezetekTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getMukodoVallalatokSzama(),

$keys[3] => $this->getVallalatokAtlagosLetszama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\Stg10FonelNagyobbSzervezetek

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = Stg10FonelNagyobbSzervezetekTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\Stg10FonelNagyobbSzervezetek

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setMukodoVallalatokSzama($value);

break;

case 3:

$this->setVallalatokAtlagosLetszama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = Stg10FonelNagyobbSzervezetekTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setMukodoVallalatokSzama($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setVallalatokAtlagosLetszama($arr[$keys[3]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\Stg10FonelNagyobbSzervezetek The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(Stg10FonelNagyobbSzervezetekTableMap::DATABASE\_NAME);

if ($this->isColumnModified(Stg10FonelNagyobbSzervezetekTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(Stg10FonelNagyobbSzervezetekTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(Stg10FonelNagyobbSzervezetekTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(Stg10FonelNagyobbSzervezetekTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(Stg10FonelNagyobbSzervezetekTableMap::COL\_MUKODO\_VALLALATOK\_SZAMA)) {

$criteria->add(Stg10FonelNagyobbSzervezetekTableMap::COL\_MUKODO\_VALLALATOK\_SZAMA, $this->mukodo\_vallalatok\_szama);

}

if ($this->isColumnModified(Stg10FonelNagyobbSzervezetekTableMap::COL\_VALLALATOK\_ATLAGOS\_LETSZAMA)) {

$criteria->add(Stg10FonelNagyobbSzervezetekTableMap::COL\_VALLALATOK\_ATLAGOS\_LETSZAMA, $this->vallalatok\_atlagos\_letszama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The Stg10FonelNagyobbSzervezetek object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \Stg10FonelNagyobbSzervezetek (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setMukodoVallalatokSzama($this->getMukodoVallalatokSzama());

$copyObj->setVallalatokAtlagosLetszama($this->getVallalatokAtlagosLetszama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \Stg10FonelNagyobbSzervezetek Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->mukodo\_vallalatok\_szama = null;

$this->vallalatok\_atlagos\_letszama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(Stg10FonelNagyobbSzervezetekTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### Stg10FonelNagyobbSzervezetekQuery.php

<?php

namespace Base;

use \Stg10FonelNagyobbSzervezetek as ChildStg10FonelNagyobbSzervezetek;

use \Stg10FonelNagyobbSzervezetekQuery as ChildStg10FonelNagyobbSzervezetekQuery;

use \Exception;

use Map\Stg10FonelNagyobbSzervezetekTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_10\_fonel\_nagyobb\_szervezetek' table.

\*

\*

\*

\* @method ChildStg10FonelNagyobbSzervezetekQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStg10FonelNagyobbSzervezetekQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStg10FonelNagyobbSzervezetekQuery orderByMukodoVallalatokSzama($order = Criteria::ASC) Order by the mukodo\_vallalatok\_szama column

\* @method ChildStg10FonelNagyobbSzervezetekQuery orderByVallalatokAtlagosLetszama($order = Criteria::ASC) Order by the vallalatok\_atlagos\_letszama column

\*

\* @method ChildStg10FonelNagyobbSzervezetekQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStg10FonelNagyobbSzervezetekQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStg10FonelNagyobbSzervezetekQuery groupByMukodoVallalatokSzama() Group by the mukodo\_vallalatok\_szama column

\* @method ChildStg10FonelNagyobbSzervezetekQuery groupByVallalatokAtlagosLetszama() Group by the vallalatok\_atlagos\_letszama column

\*

\* @method ChildStg10FonelNagyobbSzervezetekQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStg10FonelNagyobbSzervezetekQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStg10FonelNagyobbSzervezetekQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStg10FonelNagyobbSzervezetek findOne(ConnectionInterface $con = null) Return the first ChildStg10FonelNagyobbSzervezetek matching the query

\* @method ChildStg10FonelNagyobbSzervezetek findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStg10FonelNagyobbSzervezetek matching the query, or a new ChildStg10FonelNagyobbSzervezetek object populated from the query conditions when no match is found

\*

\* @method ChildStg10FonelNagyobbSzervezetek findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStg10FonelNagyobbSzervezetek filtered by the telepules\_nev column

\* @method ChildStg10FonelNagyobbSzervezetek findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStg10FonelNagyobbSzervezetek filtered by the telepules\_KSHKOD column

\* @method ChildStg10FonelNagyobbSzervezetek findOneByMukodoVallalatokSzama(int $mukodo\_vallalatok\_szama) Return the first ChildStg10FonelNagyobbSzervezetek filtered by the mukodo\_vallalatok\_szama column

\* @method ChildStg10FonelNagyobbSzervezetek findOneByVallalatokAtlagosLetszama(int $vallalatok\_atlagos\_letszama) Return the first ChildStg10FonelNagyobbSzervezetek filtered by the vallalatok\_atlagos\_letszama column

\*

\* @method ChildStg10FonelNagyobbSzervezetek[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStg10FonelNagyobbSzervezetek objects based on current ModelCriteria

\* @method ChildStg10FonelNagyobbSzervezetek[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStg10FonelNagyobbSzervezetek objects filtered by the telepules\_nev column

\* @method ChildStg10FonelNagyobbSzervezetek[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStg10FonelNagyobbSzervezetek objects filtered by the telepules\_KSHKOD column

\* @method ChildStg10FonelNagyobbSzervezetek[]|ObjectCollection findByMukodoVallalatokSzama(int $mukodo\_vallalatok\_szama) Return ChildStg10FonelNagyobbSzervezetek objects filtered by the mukodo\_vallalatok\_szama column

\* @method ChildStg10FonelNagyobbSzervezetek[]|ObjectCollection findByVallalatokAtlagosLetszama(int $vallalatok\_atlagos\_letszama) Return ChildStg10FonelNagyobbSzervezetek objects filtered by the vallalatok\_atlagos\_letszama column

\* @method ChildStg10FonelNagyobbSzervezetek[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class Stg10FonelNagyobbSzervezetekQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\Stg10FonelNagyobbSzervezetekQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\Stg10FonelNagyobbSzervezetek', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStg10FonelNagyobbSzervezetekQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStg10FonelNagyobbSzervezetekQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStg10FonelNagyobbSzervezetekQuery) {

return $criteria;

}

$query = new ChildStg10FonelNagyobbSzervezetekQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStg10FonelNagyobbSzervezetek|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The Stg10FonelNagyobbSzervezetek object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The Stg10FonelNagyobbSzervezetek object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStg10FonelNagyobbSzervezetekQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The Stg10FonelNagyobbSzervezetek object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStg10FonelNagyobbSzervezetekQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The Stg10FonelNagyobbSzervezetek object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStg10FonelNagyobbSzervezetekQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(Stg10FonelNagyobbSzervezetekTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStg10FonelNagyobbSzervezetekQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(Stg10FonelNagyobbSzervezetekTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the mukodo\_vallalatok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByMukodoVallalatokSzama(1234); // WHERE mukodo\_vallalatok\_szama = 1234

\* $query->filterByMukodoVallalatokSzama(array(12, 34)); // WHERE mukodo\_vallalatok\_szama IN (12, 34)

\* $query->filterByMukodoVallalatokSzama(array('min' => 12)); // WHERE mukodo\_vallalatok\_szama > 12

\* </code>

\*

\* @param mixed $mukodoVallalatokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStg10FonelNagyobbSzervezetekQuery The current query, for fluid interface

\*/

public function filterByMukodoVallalatokSzama($mukodoVallalatokSzama = null, $comparison = null)

{

if (is\_array($mukodoVallalatokSzama)) {

$useMinMax = false;

if (isset($mukodoVallalatokSzama['min'])) {

$this->addUsingAlias(Stg10FonelNagyobbSzervezetekTableMap::COL\_MUKODO\_VALLALATOK\_SZAMA, $mukodoVallalatokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($mukodoVallalatokSzama['max'])) {

$this->addUsingAlias(Stg10FonelNagyobbSzervezetekTableMap::COL\_MUKODO\_VALLALATOK\_SZAMA, $mukodoVallalatokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(Stg10FonelNagyobbSzervezetekTableMap::COL\_MUKODO\_VALLALATOK\_SZAMA, $mukodoVallalatokSzama, $comparison);

}

/\*\*

\* Filter the query on the vallalatok\_atlagos\_letszama column

\*

\* Example usage:

\* <code>

\* $query->filterByVallalatokAtlagosLetszama(1234); // WHERE vallalatok\_atlagos\_letszama = 1234

\* $query->filterByVallalatokAtlagosLetszama(array(12, 34)); // WHERE vallalatok\_atlagos\_letszama IN (12, 34)

\* $query->filterByVallalatokAtlagosLetszama(array('min' => 12)); // WHERE vallalatok\_atlagos\_letszama > 12

\* </code>

\*

\* @param mixed $vallalatokAtlagosLetszama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStg10FonelNagyobbSzervezetekQuery The current query, for fluid interface

\*/

public function filterByVallalatokAtlagosLetszama($vallalatokAtlagosLetszama = null, $comparison = null)

{

if (is\_array($vallalatokAtlagosLetszama)) {

$useMinMax = false;

if (isset($vallalatokAtlagosLetszama['min'])) {

$this->addUsingAlias(Stg10FonelNagyobbSzervezetekTableMap::COL\_VALLALATOK\_ATLAGOS\_LETSZAMA, $vallalatokAtlagosLetszama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($vallalatokAtlagosLetszama['max'])) {

$this->addUsingAlias(Stg10FonelNagyobbSzervezetekTableMap::COL\_VALLALATOK\_ATLAGOS\_LETSZAMA, $vallalatokAtlagosLetszama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(Stg10FonelNagyobbSzervezetekTableMap::COL\_VALLALATOK\_ATLAGOS\_LETSZAMA, $vallalatokAtlagosLetszama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStg10FonelNagyobbSzervezetek $stg10FonelNagyobbSzervezetek Object to remove from the list of results

\*

\* @return $this|ChildStg10FonelNagyobbSzervezetekQuery The current query, for fluid interface

\*/

public function prune($stg10FonelNagyobbSzervezetek = null)

{

if ($stg10FonelNagyobbSzervezetek) {

throw new LogicException('Stg10FonelNagyobbSzervezetek object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_10\_fonel\_nagyobb\_szervezetek table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(Stg10FonelNagyobbSzervezetekTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

Stg10FonelNagyobbSzervezetekTableMap::clearInstancePool();

Stg10FonelNagyobbSzervezetekTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(Stg10FonelNagyobbSzervezetekTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(Stg10FonelNagyobbSzervezetekTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

Stg10FonelNagyobbSzervezetekTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

Stg10FonelNagyobbSzervezetekTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // Stg10FonelNagyobbSzervezetekQuery

#### StgBelfoldiOsszesJovedelem.php

<?php

namespace Base;

use \StgBelfoldiOsszesJovedelemQuery as ChildStgBelfoldiOsszesJovedelemQuery;

use \Exception;

use \PDO;

use Map\StgBelfoldiOsszesJovedelemTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgBelfoldiOsszesJovedelem implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgBelfoldiOsszesJovedelemTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the jovedelem field.

\* @var double

\*/

protected $jovedelem;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgBelfoldiOsszesJovedelem object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgBelfoldiOsszesJovedelem</code> instance. If

\* <code>obj</code> is an instance of <code>StgBelfoldiOsszesJovedelem</code>, delegates to

\* <code>equals(StgBelfoldiOsszesJovedelem)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgBelfoldiOsszesJovedelem The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [jovedelem] column value.

\*

\* @return double

\*/

public function getJovedelem()

{

return $this->jovedelem;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgBelfoldiOsszesJovedelemTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgBelfoldiOsszesJovedelemTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgBelfoldiOsszesJovedelemTableMap::translateFieldName('Jovedelem', TableMap::TYPE\_PHPNAME, $indexType)];

$this->jovedelem = (null !== $col) ? (double) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 3; // 3 = StgBelfoldiOsszesJovedelemTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgBelfoldiOsszesJovedelem'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgBelfoldiOsszesJovedelem The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgBelfoldiOsszesJovedelem The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [jovedelem] column.

\*

\* @param double $v new value

\* @return $this|\StgBelfoldiOsszesJovedelem The current object (for fluent API support)

\*/

public function setJovedelem($v)

{

if ($v !== null) {

$v = (double) $v;

}

if ($this->jovedelem !== $v) {

$this->jovedelem = $v;

$this->modifiedColumns[StgBelfoldiOsszesJovedelemTableMap::COL\_JOVEDELEM] = true;

}

return $this;

} // setJovedelem()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgBelfoldiOsszesJovedelemTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgBelfoldiOsszesJovedelemQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgBelfoldiOsszesJovedelem::setDeleted()

\* @see StgBelfoldiOsszesJovedelem::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgBelfoldiOsszesJovedelemTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgBelfoldiOsszesJovedelemQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgBelfoldiOsszesJovedelemTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgBelfoldiOsszesJovedelemTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgBelfoldiOsszesJovedelemTableMap::COL\_JOVEDELEM)) {

$modifiedColumns[':p' . $index++] = 'JOVEDELEM';

}

$sql = sprintf(

'INSERT INTO stg\_belfoldi\_osszes\_jovedelem (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'JOVEDELEM':

$stmt->bindValue($identifier, $this->jovedelem, PDO::PARAM\_STR);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgBelfoldiOsszesJovedelemTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getJovedelem();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgBelfoldiOsszesJovedelem'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgBelfoldiOsszesJovedelem'][$this->getPrimaryKey()] = true;

$keys = StgBelfoldiOsszesJovedelemTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getJovedelem(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgBelfoldiOsszesJovedelem

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgBelfoldiOsszesJovedelemTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgBelfoldiOsszesJovedelem

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setJovedelem($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgBelfoldiOsszesJovedelemTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setJovedelem($arr[$keys[2]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgBelfoldiOsszesJovedelem The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgBelfoldiOsszesJovedelemTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgBelfoldiOsszesJovedelemTableMap::COL\_JOVEDELEM)) {

$criteria->add(StgBelfoldiOsszesJovedelemTableMap::COL\_JOVEDELEM, $this->jovedelem);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

$criteria = new Criteria(StgBelfoldiOsszesJovedelemTableMap::DATABASE\_NAME);

$criteria->add(StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = null !== $this->getTelepulesNev();

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns the primary key for this object (row).

\* @return string

\*/

public function getPrimaryKey()

{

return $this->getTelepulesNev();

}

/\*\*

\* Generic method to set the primary key (telepules\_nev column).

\*

\* @param string $key Primary key.

\* @return void

\*/

public function setPrimaryKey($key)

{

$this->setTelepulesNev($key);

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return null === $this->getTelepulesNev();

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgBelfoldiOsszesJovedelem (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setJovedelem($this->getJovedelem());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgBelfoldiOsszesJovedelem Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->jovedelem = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgBelfoldiOsszesJovedelemTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgBelfoldiOsszesJovedelemQuery.php

<?php

namespace Base;

use \StgBelfoldiOsszesJovedelem as ChildStgBelfoldiOsszesJovedelem;

use \StgBelfoldiOsszesJovedelemQuery as ChildStgBelfoldiOsszesJovedelemQuery;

use \Exception;

use \PDO;

use Map\StgBelfoldiOsszesJovedelemTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_belfoldi\_osszes\_jovedelem' table.

\*

\*

\*

\* @method ChildStgBelfoldiOsszesJovedelemQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgBelfoldiOsszesJovedelemQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgBelfoldiOsszesJovedelemQuery orderByJovedelem($order = Criteria::ASC) Order by the jovedelem column

\*

\* @method ChildStgBelfoldiOsszesJovedelemQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgBelfoldiOsszesJovedelemQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgBelfoldiOsszesJovedelemQuery groupByJovedelem() Group by the jovedelem column

\*

\* @method ChildStgBelfoldiOsszesJovedelemQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgBelfoldiOsszesJovedelemQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgBelfoldiOsszesJovedelemQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgBelfoldiOsszesJovedelem findOne(ConnectionInterface $con = null) Return the first ChildStgBelfoldiOsszesJovedelem matching the query

\* @method ChildStgBelfoldiOsszesJovedelem findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgBelfoldiOsszesJovedelem matching the query, or a new ChildStgBelfoldiOsszesJovedelem object populated from the query conditions when no match is found

\*

\* @method ChildStgBelfoldiOsszesJovedelem findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgBelfoldiOsszesJovedelem filtered by the telepules\_nev column

\* @method ChildStgBelfoldiOsszesJovedelem findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgBelfoldiOsszesJovedelem filtered by the telepules\_KSHKOD column

\* @method ChildStgBelfoldiOsszesJovedelem findOneByJovedelem(double $jovedelem) Return the first ChildStgBelfoldiOsszesJovedelem filtered by the jovedelem column

\*

\* @method ChildStgBelfoldiOsszesJovedelem[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgBelfoldiOsszesJovedelem objects based on current ModelCriteria

\* @method ChildStgBelfoldiOsszesJovedelem[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgBelfoldiOsszesJovedelem objects filtered by the telepules\_nev column

\* @method ChildStgBelfoldiOsszesJovedelem[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgBelfoldiOsszesJovedelem objects filtered by the telepules\_KSHKOD column

\* @method ChildStgBelfoldiOsszesJovedelem[]|ObjectCollection findByJovedelem(double $jovedelem) Return ChildStgBelfoldiOsszesJovedelem objects filtered by the jovedelem column

\* @method ChildStgBelfoldiOsszesJovedelem[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgBelfoldiOsszesJovedelemQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgBelfoldiOsszesJovedelemQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgBelfoldiOsszesJovedelem', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgBelfoldiOsszesJovedelemQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgBelfoldiOsszesJovedelemQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgBelfoldiOsszesJovedelemQuery) {

return $criteria;

}

$query = new ChildStgBelfoldiOsszesJovedelemQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgBelfoldiOsszesJovedelem|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

if ($key === null) {

return null;

}

if ((null !== ($obj = StgBelfoldiOsszesJovedelemTableMap::getInstanceFromPool((string) $key))) && !$this->formatter) {

// the object is already in the instance pool

return $obj;

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgBelfoldiOsszesJovedelemTableMap::DATABASE\_NAME);

}

$this->basePreSelect($con);

if ($this->formatter || $this->modelAlias || $this->with || $this->select

|| $this->selectColumns || $this->asColumns || $this->selectModifiers

|| $this->map || $this->having || $this->joins) {

return $this->findPkComplex($key, $con);

} else {

return $this->findPkSimple($key, $con);

}

}

/\*\*

\* Find object by primary key using raw SQL to go fast.

\* Bypass doSelect() and the object formatter by using generated code.

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con A connection object

\*

\* @return ChildStgBelfoldiOsszesJovedelem A model object, or null if the key is not found

\*/

protected function findPkSimple($key, ConnectionInterface $con)

{

$sql = 'SELECT TELEPULES\_NEV, TELEPULES\_KSHKOD, JOVEDELEM FROM stg\_belfoldi\_osszes\_jovedelem WHERE TELEPULES\_NEV = :p0';

try {

$stmt = $con->prepare($sql);

$stmt->bindValue(':p0', $key, PDO::PARAM\_STR);

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute SELECT statement [%s]', $sql), 0, $e);

}

$obj = null;

if ($row = $stmt->fetch(\PDO::FETCH\_NUM)) {

/\*\* @var ChildStgBelfoldiOsszesJovedelem $obj \*/

$obj = new ChildStgBelfoldiOsszesJovedelem();

$obj->hydrate($row);

StgBelfoldiOsszesJovedelemTableMap::addInstanceToPool($obj, (string) $key);

}

$stmt->closeCursor();

return $obj;

}

/\*\*

\* Find object by primary key.

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con A connection object

\*

\* @return ChildStgBelfoldiOsszesJovedelem|array|mixed the result, formatted by the current formatter

\*/

protected function findPkComplex($key, ConnectionInterface $con)

{

// As the query uses a PK condition, no limit(1) is necessary.

$criteria = $this->isKeepQuery() ? clone $this : $this;

$dataFetcher = $criteria

->filterByPrimaryKey($key)

->doSelect($con);

return $criteria->getFormatter()->init($criteria)->formatOne($dataFetcher);

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(12, 56, 832), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getReadConnection($this->getDbName());

}

$this->basePreSelect($con);

$criteria = $this->isKeepQuery() ? clone $this : $this;

$dataFetcher = $criteria

->filterByPrimaryKeys($keys)

->doSelect($con);

return $criteria->getFormatter()->init($criteria)->format($dataFetcher);

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgBelfoldiOsszesJovedelemQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

return $this->addUsingAlias(StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_NEV, $key, Criteria::EQUAL);

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgBelfoldiOsszesJovedelemQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

return $this->addUsingAlias(StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_NEV, $keys, Criteria::IN);

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgBelfoldiOsszesJovedelemQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgBelfoldiOsszesJovedelemQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the jovedelem column

\*

\* Example usage:

\* <code>

\* $query->filterByJovedelem(1234); // WHERE jovedelem = 1234

\* $query->filterByJovedelem(array(12, 34)); // WHERE jovedelem IN (12, 34)

\* $query->filterByJovedelem(array('min' => 12)); // WHERE jovedelem > 12

\* </code>

\*

\* @param mixed $jovedelem The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgBelfoldiOsszesJovedelemQuery The current query, for fluid interface

\*/

public function filterByJovedelem($jovedelem = null, $comparison = null)

{

if (is\_array($jovedelem)) {

$useMinMax = false;

if (isset($jovedelem['min'])) {

$this->addUsingAlias(StgBelfoldiOsszesJovedelemTableMap::COL\_JOVEDELEM, $jovedelem['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($jovedelem['max'])) {

$this->addUsingAlias(StgBelfoldiOsszesJovedelemTableMap::COL\_JOVEDELEM, $jovedelem['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgBelfoldiOsszesJovedelemTableMap::COL\_JOVEDELEM, $jovedelem, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgBelfoldiOsszesJovedelem $stgBelfoldiOsszesJovedelem Object to remove from the list of results

\*

\* @return $this|ChildStgBelfoldiOsszesJovedelemQuery The current query, for fluid interface

\*/

public function prune($stgBelfoldiOsszesJovedelem = null)

{

if ($stgBelfoldiOsszesJovedelem) {

$this->addUsingAlias(StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_NEV, $stgBelfoldiOsszesJovedelem->getTelepulesNev(), Criteria::NOT\_EQUAL);

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_belfoldi\_osszes\_jovedelem table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgBelfoldiOsszesJovedelemTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgBelfoldiOsszesJovedelemTableMap::clearInstancePool();

StgBelfoldiOsszesJovedelemTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgBelfoldiOsszesJovedelemTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgBelfoldiOsszesJovedelemTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgBelfoldiOsszesJovedelemTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgBelfoldiOsszesJovedelemTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgBelfoldiOsszesJovedelemQuery

#### StgCegautok.php

<?php

namespace Base;

use \StgCegautokQuery as ChildStgCegautokQuery;

use \Exception;

use \PDO;

use Map\StgCegautokTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgCegautok implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgCegautokTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the szemelygepkocsik\_szama field.

\* @var int

\*/

protected $szemelygepkocsik\_szama;

/\*\*

\* The value for the termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama field.

\* @var int

\*/

protected $termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgCegautok object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgCegautok</code> instance. If

\* <code>obj</code> is an instance of <code>StgCegautok</code>, delegates to

\* <code>equals(StgCegautok)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgCegautok The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [szemelygepkocsik\_szama] column value.

\*

\* @return int

\*/

public function getSzemelygepkocsikSzama()

{

return $this->szemelygepkocsik\_szama;

}

/\*\*

\* Get the [termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama] column value.

\*

\* @return int

\*/

public function getTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama()

{

return $this->termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgCegautokTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgCegautokTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgCegautokTableMap::translateFieldName('SzemelygepkocsikSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->szemelygepkocsik\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgCegautokTableMap::translateFieldName('TermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 4; // 4 = StgCegautokTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgCegautok'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgCegautok The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgCegautokTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgCegautok The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgCegautokTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [szemelygepkocsik\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgCegautok The current object (for fluent API support)

\*/

public function setSzemelygepkocsikSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->szemelygepkocsik\_szama !== $v) {

$this->szemelygepkocsik\_szama = $v;

$this->modifiedColumns[StgCegautokTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA] = true;

}

return $this;

} // setSzemelygepkocsikSzama()

/\*\*

\* Set the value of [termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgCegautok The current object (for fluent API support)

\*/

public function setTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama !== $v) {

$this->termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama = $v;

$this->modifiedColumns[StgCegautokTableMap::COL\_TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA] = true;

}

return $this;

} // setTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgCegautokTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgCegautokQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgCegautok::setDeleted()

\* @see StgCegautok::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgCegautokTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgCegautokQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgCegautokTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgCegautokTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgCegautokTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgCegautokTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgCegautokTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'SZEMELYGEPKOCSIK\_SZAMA';

}

if ($this->isColumnModified(StgCegautokTableMap::COL\_TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_cegautok (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'SZEMELYGEPKOCSIK\_SZAMA':

$stmt->bindValue($identifier, $this->szemelygepkocsik\_szama, PDO::PARAM\_INT);

break;

case 'TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA':

$stmt->bindValue($identifier, $this->termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgCegautokTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getSzemelygepkocsikSzama();

break;

case 3:

return $this->getTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgCegautok'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgCegautok'][$this->getPrimaryKey()] = true;

$keys = StgCegautokTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getSzemelygepkocsikSzama(),

$keys[3] => $this->getTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgCegautok

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgCegautokTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgCegautok

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setSzemelygepkocsikSzama($value);

break;

case 3:

$this->setTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgCegautokTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setSzemelygepkocsikSzama($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama($arr[$keys[3]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgCegautok The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgCegautokTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgCegautokTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgCegautokTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgCegautokTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgCegautokTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgCegautokTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA)) {

$criteria->add(StgCegautokTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA, $this->szemelygepkocsik\_szama);

}

if ($this->isColumnModified(StgCegautokTableMap::COL\_TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA)) {

$criteria->add(StgCegautokTableMap::COL\_TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA, $this->termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgCegautok object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgCegautok (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setSzemelygepkocsikSzama($this->getSzemelygepkocsikSzama());

$copyObj->setTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama($this->getTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgCegautok Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->szemelygepkocsik\_szama = null;

$this->termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgCegautokTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgCegautokQuery.php

<?php

namespace Base;

use \StgCegautok as ChildStgCegautok;

use \StgCegautokQuery as ChildStgCegautokQuery;

use \Exception;

use Map\StgCegautokTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_cegautok' table.

\*

\*

\*

\* @method ChildStgCegautokQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgCegautokQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgCegautokQuery orderBySzemelygepkocsikSzama($order = Criteria::ASC) Order by the szemelygepkocsik\_szama column

\* @method ChildStgCegautokQuery orderByTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama($order = Criteria::ASC) Order by the termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama column

\*

\* @method ChildStgCegautokQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgCegautokQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgCegautokQuery groupBySzemelygepkocsikSzama() Group by the szemelygepkocsik\_szama column

\* @method ChildStgCegautokQuery groupByTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama() Group by the termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama column

\*

\* @method ChildStgCegautokQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgCegautokQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgCegautokQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgCegautok findOne(ConnectionInterface $con = null) Return the first ChildStgCegautok matching the query

\* @method ChildStgCegautok findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgCegautok matching the query, or a new ChildStgCegautok object populated from the query conditions when no match is found

\*

\* @method ChildStgCegautok findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgCegautok filtered by the telepules\_nev column

\* @method ChildStgCegautok findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgCegautok filtered by the telepules\_KSHKOD column

\* @method ChildStgCegautok findOneBySzemelygepkocsikSzama(int $szemelygepkocsik\_szama) Return the first ChildStgCegautok filtered by the szemelygepkocsik\_szama column

\* @method ChildStgCegautok findOneByTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama(int $termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama) Return the first ChildStgCegautok filtered by the termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama column

\*

\* @method ChildStgCegautok[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgCegautok objects based on current ModelCriteria

\* @method ChildStgCegautok[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgCegautok objects filtered by the telepules\_nev column

\* @method ChildStgCegautok[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgCegautok objects filtered by the telepules\_KSHKOD column

\* @method ChildStgCegautok[]|ObjectCollection findBySzemelygepkocsikSzama(int $szemelygepkocsik\_szama) Return ChildStgCegautok objects filtered by the szemelygepkocsik\_szama column

\* @method ChildStgCegautok[]|ObjectCollection findByTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama(int $termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama) Return ChildStgCegautok objects filtered by the termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama column

\* @method ChildStgCegautok[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgCegautokQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgCegautokQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgCegautok', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgCegautokQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgCegautokQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgCegautokQuery) {

return $criteria;

}

$query = new ChildStgCegautokQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgCegautok|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgCegautok object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgCegautok object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgCegautokQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgCegautok object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgCegautokQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgCegautok object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgCegautokQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgCegautokTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgCegautokQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgCegautokTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the szemelygepkocsik\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterBySzemelygepkocsikSzama(1234); // WHERE szemelygepkocsik\_szama = 1234

\* $query->filterBySzemelygepkocsikSzama(array(12, 34)); // WHERE szemelygepkocsik\_szama IN (12, 34)

\* $query->filterBySzemelygepkocsikSzama(array('min' => 12)); // WHERE szemelygepkocsik\_szama > 12

\* </code>

\*

\* @param mixed $szemelygepkocsikSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgCegautokQuery The current query, for fluid interface

\*/

public function filterBySzemelygepkocsikSzama($szemelygepkocsikSzama = null, $comparison = null)

{

if (is\_array($szemelygepkocsikSzama)) {

$useMinMax = false;

if (isset($szemelygepkocsikSzama['min'])) {

$this->addUsingAlias(StgCegautokTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA, $szemelygepkocsikSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($szemelygepkocsikSzama['max'])) {

$this->addUsingAlias(StgCegautokTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA, $szemelygepkocsikSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgCegautokTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA, $szemelygepkocsikSzama, $comparison);

}

/\*\*

\* Filter the query on the termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama(1234); // WHERE termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama = 1234

\* $query->filterByTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama(array(12, 34)); // WHERE termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama IN (12, 34)

\* $query->filterByTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama(array('min' => 12)); // WHERE termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama > 12

\* </code>

\*

\* @param mixed $termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgCegautokQuery The current query, for fluid interface

\*/

public function filterByTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama($termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama = null, $comparison = null)

{

if (is\_array($termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama)) {

$useMinMax = false;

if (isset($termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama['min'])) {

$this->addUsingAlias(StgCegautokTableMap::COL\_TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA, $termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama['max'])) {

$this->addUsingAlias(StgCegautokTableMap::COL\_TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA, $termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgCegautokTableMap::COL\_TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA, $termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgCegautok $stgCegautok Object to remove from the list of results

\*

\* @return $this|ChildStgCegautokQuery The current query, for fluid interface

\*/

public function prune($stgCegautok = null)

{

if ($stgCegautok) {

throw new LogicException('StgCegautok object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_cegautok table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgCegautokTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgCegautokTableMap::clearInstancePool();

StgCegautokTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgCegautokTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgCegautokTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgCegautokTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgCegautokTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgCegautokQuery

<?php

namespace Base;

use \StgCegautok as ChildStgCegautok;

use \StgCegautokQuery as ChildStgCegautokQuery;

use \Exception;

use Map\StgCegautokTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_cegautok' table.

\*

\*

\*

\* @method ChildStgCegautokQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgCegautokQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgCegautokQuery orderBySzemelygepkocsikSzama($order = Criteria::ASC) Order by the szemelygepkocsik\_szama column

\* @method ChildStgCegautokQuery orderByTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama($order = Criteria::ASC) Order by the termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama column

\*

\* @method ChildStgCegautokQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgCegautokQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgCegautokQuery groupBySzemelygepkocsikSzama() Group by the szemelygepkocsik\_szama column

\* @method ChildStgCegautokQuery groupByTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama() Group by the termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama column

\*

\* @method ChildStgCegautokQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgCegautokQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgCegautokQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgCegautok findOne(ConnectionInterface $con = null) Return the first ChildStgCegautok matching the query

\* @method ChildStgCegautok findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgCegautok matching the query, or a new ChildStgCegautok object populated from the query conditions when no match is found

\*

\* @method ChildStgCegautok findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgCegautok filtered by the telepules\_nev column

\* @method ChildStgCegautok findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgCegautok filtered by the telepules\_KSHKOD column

\* @method ChildStgCegautok findOneBySzemelygepkocsikSzama(int $szemelygepkocsik\_szama) Return the first ChildStgCegautok filtered by the szemelygepkocsik\_szama column

\* @method ChildStgCegautok findOneByTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama(int $termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama) Return the first ChildStgCegautok filtered by the termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama column

\*

\* @method ChildStgCegautok[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgCegautok objects based on current ModelCriteria

\* @method ChildStgCegautok[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgCegautok objects filtered by the telepules\_nev column

\* @method ChildStgCegautok[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgCegautok objects filtered by the telepules\_KSHKOD column

\* @method ChildStgCegautok[]|ObjectCollection findBySzemelygepkocsikSzama(int $szemelygepkocsik\_szama) Return ChildStgCegautok objects filtered by the szemelygepkocsik\_szama column

\* @method ChildStgCegautok[]|ObjectCollection findByTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama(int $termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama) Return ChildStgCegautok objects filtered by the termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama column

\* @method ChildStgCegautok[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgCegautokQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgCegautokQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgCegautok', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgCegautokQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgCegautokQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgCegautokQuery) {

return $criteria;

}

$query = new ChildStgCegautokQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgCegautok|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgCegautok object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgCegautok object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgCegautokQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgCegautok object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgCegautokQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgCegautok object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgCegautokQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgCegautokTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgCegautokQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgCegautokTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the szemelygepkocsik\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterBySzemelygepkocsikSzama(1234); // WHERE szemelygepkocsik\_szama = 1234

\* $query->filterBySzemelygepkocsikSzama(array(12, 34)); // WHERE szemelygepkocsik\_szama IN (12, 34)

\* $query->filterBySzemelygepkocsikSzama(array('min' => 12)); // WHERE szemelygepkocsik\_szama > 12

\* </code>

\*

\* @param mixed $szemelygepkocsikSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgCegautokQuery The current query, for fluid interface

\*/

public function filterBySzemelygepkocsikSzama($szemelygepkocsikSzama = null, $comparison = null)

{

if (is\_array($szemelygepkocsikSzama)) {

$useMinMax = false;

if (isset($szemelygepkocsikSzama['min'])) {

$this->addUsingAlias(StgCegautokTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA, $szemelygepkocsikSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($szemelygepkocsikSzama['max'])) {

$this->addUsingAlias(StgCegautokTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA, $szemelygepkocsikSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgCegautokTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA, $szemelygepkocsikSzama, $comparison);

}

/\*\*

\* Filter the query on the termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama(1234); // WHERE termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama = 1234

\* $query->filterByTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama(array(12, 34)); // WHERE termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama IN (12, 34)

\* $query->filterByTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama(array('min' => 12)); // WHERE termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama > 12

\* </code>

\*

\* @param mixed $termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgCegautokQuery The current query, for fluid interface

\*/

public function filterByTermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama($termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama = null, $comparison = null)

{

if (is\_array($termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama)) {

$useMinMax = false;

if (isset($termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama['min'])) {

$this->addUsingAlias(StgCegautokTableMap::COL\_TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA, $termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama['max'])) {

$this->addUsingAlias(StgCegautokTableMap::COL\_TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA, $termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgCegautokTableMap::COL\_TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA, $termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgCegautok $stgCegautok Object to remove from the list of results

\*

\* @return $this|ChildStgCegautokQuery The current query, for fluid interface

\*/

public function prune($stgCegautok = null)

{

if ($stgCegautok) {

throw new LogicException('StgCegautok object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_cegautok table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgCegautokTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgCegautokTableMap::clearInstancePool();

StgCegautokTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgCegautokTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgCegautokTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgCegautokTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgCegautokTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgCegautokQuery

#### StgEgeszsegugyiSzocialisEllatottakElvandoroltak.php

<?php

namespace Base;

use \StgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery as ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery;

use \Exception;

use \PDO;

use Map\StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgEgeszsegugyiSzocialisEllatottakElvandoroltak implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama field.

\* @var int

\*/

protected $hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama;

/\*\*

\* The value for the hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama field.

\* @var int

\*/

protected $hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama;

/\*\*

\* The value for the elbocsatott\_betegek\_szama field.

\* @var int

\*/

protected $elbocsatott\_betegek\_szama;

/\*\*

\* The value for the megjelenesi\_esetek\_szama field.

\* @var int

\*/

protected $megjelenesi\_esetek\_szama;

/\*\*

\* The value for the nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama field.

\* @var int

\*/

protected $nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama;

/\*\*

\* The value for the nappali\_ellatasban\_reszesulo\_idoskoruak\_szama field.

\* @var int

\*/

protected $nappali\_ellatasban\_reszesulo\_idoskoruak\_szama;

/\*\*

\* The value for the nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama field.

\* @var int

\*/

protected $nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama;

/\*\*

\* The value for the nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama field.

\* @var int

\*/

protected $nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama;

/\*\*

\* The value for the odavandorlasok\_szama field.

\* @var int

\*/

protected $odavandorlasok\_szama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgEgeszsegugyiSzocialisEllatottakElvandoroltak object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgEgeszsegugyiSzocialisEllatottakElvandoroltak</code> instance. If

\* <code>obj</code> is an instance of <code>StgEgeszsegugyiSzocialisEllatottakElvandoroltak</code>, delegates to

\* <code>equals(StgEgeszsegugyiSzocialisEllatottakElvandoroltak)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgEgeszsegugyiSzocialisEllatottakElvandoroltak The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama] column value.

\*

\* @return int

\*/

public function getHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama()

{

return $this->hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama;

}

/\*\*

\* Get the [hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama] column value.

\*

\* @return int

\*/

public function getHaziOrvosiEllatasbanRendelesenMegjelentekSzama()

{

return $this->hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama;

}

/\*\*

\* Get the [elbocsatott\_betegek\_szama] column value.

\*

\* @return int

\*/

public function getElbocsatottBetegekSzama()

{

return $this->elbocsatott\_betegek\_szama;

}

/\*\*

\* Get the [megjelenesi\_esetek\_szama] column value.

\*

\* @return int

\*/

public function getMegjelenesiEsetekSzama()

{

return $this->megjelenesi\_esetek\_szama;

}

/\*\*

\* Get the [nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama] column value.

\*

\* @return int

\*/

public function getNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama()

{

return $this->nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama;

}

/\*\*

\* Get the [nappali\_ellatasban\_reszesulo\_idoskoruak\_szama] column value.

\*

\* @return int

\*/

public function getNappaliEllatasbanReszesuloIdoskoruakSzama()

{

return $this->nappali\_ellatasban\_reszesulo\_idoskoruak\_szama;

}

/\*\*

\* Get the [nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama] column value.

\*

\* @return int

\*/

public function getNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama()

{

return $this->nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama;

}

/\*\*

\* Get the [nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama] column value.

\*

\* @return int

\*/

public function getNappaliEllatasbanReszesuloSzenvedelybetegekSzama()

{

return $this->nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama;

}

/\*\*

\* Get the [odavandorlasok\_szama] column value.

\*

\* @return int

\*/

public function getOdavandorlasokSzama()

{

return $this->odavandorlasok\_szama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::translateFieldName('HaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::translateFieldName('HaziOrvosiEllatasbanRendelesenMegjelentekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::translateFieldName('ElbocsatottBetegekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->elbocsatott\_betegek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 5 + $startcol : StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::translateFieldName('MegjelenesiEsetekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->megjelenesi\_esetek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 6 + $startcol : StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::translateFieldName('NappaliEllatasbanReszesuloFogyatekosSzemelyekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 7 + $startcol : StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::translateFieldName('NappaliEllatasbanReszesuloIdoskoruakSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_ellatasban\_reszesulo\_idoskoruak\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 8 + $startcol : StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::translateFieldName('NappaliEllatasbanReszesuloPszichiatriaiBetegekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 9 + $startcol : StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::translateFieldName('NappaliEllatasbanReszesuloSzenvedelybetegekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 10 + $startcol : StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::translateFieldName('OdavandorlasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->odavandorlasok\_szama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 11; // 11 = StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgEgeszsegugyiSzocialisEllatottakElvandoroltak'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgEgeszsegugyiSzocialisEllatottakElvandoroltak The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgEgeszsegugyiSzocialisEllatottakElvandoroltak The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgEgeszsegugyiSzocialisEllatottakElvandoroltak The current object (for fluent API support)

\*/

public function setHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama !== $v) {

$this->hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama = $v;

$this->modifiedColumns[StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA] = true;

}

return $this;

} // setHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama()

/\*\*

\* Set the value of [hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgEgeszsegugyiSzocialisEllatottakElvandoroltak The current object (for fluent API support)

\*/

public function setHaziOrvosiEllatasbanRendelesenMegjelentekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama !== $v) {

$this->hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama = $v;

$this->modifiedColumns[StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA] = true;

}

return $this;

} // setHaziOrvosiEllatasbanRendelesenMegjelentekSzama()

/\*\*

\* Set the value of [elbocsatott\_betegek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgEgeszsegugyiSzocialisEllatottakElvandoroltak The current object (for fluent API support)

\*/

public function setElbocsatottBetegekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->elbocsatott\_betegek\_szama !== $v) {

$this->elbocsatott\_betegek\_szama = $v;

$this->modifiedColumns[StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ELBOCSATOTT\_BETEGEK\_SZAMA] = true;

}

return $this;

} // setElbocsatottBetegekSzama()

/\*\*

\* Set the value of [megjelenesi\_esetek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgEgeszsegugyiSzocialisEllatottakElvandoroltak The current object (for fluent API support)

\*/

public function setMegjelenesiEsetekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->megjelenesi\_esetek\_szama !== $v) {

$this->megjelenesi\_esetek\_szama = $v;

$this->modifiedColumns[StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_MEGJELENESI\_ESETEK\_SZAMA] = true;

}

return $this;

} // setMegjelenesiEsetekSzama()

/\*\*

\* Set the value of [nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgEgeszsegugyiSzocialisEllatottakElvandoroltak The current object (for fluent API support)

\*/

public function setNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama !== $v) {

$this->nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama = $v;

$this->modifiedColumns[StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA] = true;

}

return $this;

} // setNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama()

/\*\*

\* Set the value of [nappali\_ellatasban\_reszesulo\_idoskoruak\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgEgeszsegugyiSzocialisEllatottakElvandoroltak The current object (for fluent API support)

\*/

public function setNappaliEllatasbanReszesuloIdoskoruakSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_ellatasban\_reszesulo\_idoskoruak\_szama !== $v) {

$this->nappali\_ellatasban\_reszesulo\_idoskoruak\_szama = $v;

$this->modifiedColumns[StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA] = true;

}

return $this;

} // setNappaliEllatasbanReszesuloIdoskoruakSzama()

/\*\*

\* Set the value of [nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgEgeszsegugyiSzocialisEllatottakElvandoroltak The current object (for fluent API support)

\*/

public function setNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama !== $v) {

$this->nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama = $v;

$this->modifiedColumns[StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA] = true;

}

return $this;

} // setNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama()

/\*\*

\* Set the value of [nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgEgeszsegugyiSzocialisEllatottakElvandoroltak The current object (for fluent API support)

\*/

public function setNappaliEllatasbanReszesuloSzenvedelybetegekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama !== $v) {

$this->nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama = $v;

$this->modifiedColumns[StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA] = true;

}

return $this;

} // setNappaliEllatasbanReszesuloSzenvedelybetegekSzama()

/\*\*

\* Set the value of [odavandorlasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgEgeszsegugyiSzocialisEllatottakElvandoroltak The current object (for fluent API support)

\*/

public function setOdavandorlasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->odavandorlasok\_szama !== $v) {

$this->odavandorlasok\_szama = $v;

$this->modifiedColumns[StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ODAVANDORLASOK\_SZAMA] = true;

}

return $this;

} // setOdavandorlasokSzama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgEgeszsegugyiSzocialisEllatottakElvandoroltak::setDeleted()

\* @see StgEgeszsegugyiSzocialisEllatottakElvandoroltak::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA';

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA';

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ELBOCSATOTT\_BETEGEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'ELBOCSATOTT\_BETEGEK\_SZAMA';

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_MEGJELENESI\_ESETEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'MEGJELENESI\_ESETEK\_SZAMA';

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA';

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA';

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA';

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA';

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ODAVANDORLASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'ODAVANDORLASOK\_SZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA':

$stmt->bindValue($identifier, $this->hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama, PDO::PARAM\_INT);

break;

case 'HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA':

$stmt->bindValue($identifier, $this->hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama, PDO::PARAM\_INT);

break;

case 'ELBOCSATOTT\_BETEGEK\_SZAMA':

$stmt->bindValue($identifier, $this->elbocsatott\_betegek\_szama, PDO::PARAM\_INT);

break;

case 'MEGJELENESI\_ESETEK\_SZAMA':

$stmt->bindValue($identifier, $this->megjelenesi\_esetek\_szama, PDO::PARAM\_INT);

break;

case 'NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA':

$stmt->bindValue($identifier, $this->nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama, PDO::PARAM\_INT);

break;

case 'NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA':

$stmt->bindValue($identifier, $this->nappali\_ellatasban\_reszesulo\_idoskoruak\_szama, PDO::PARAM\_INT);

break;

case 'NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA':

$stmt->bindValue($identifier, $this->nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama, PDO::PARAM\_INT);

break;

case 'NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA':

$stmt->bindValue($identifier, $this->nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama, PDO::PARAM\_INT);

break;

case 'ODAVANDORLASOK\_SZAMA':

$stmt->bindValue($identifier, $this->odavandorlasok\_szama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama();

break;

case 3:

return $this->getHaziOrvosiEllatasbanRendelesenMegjelentekSzama();

break;

case 4:

return $this->getElbocsatottBetegekSzama();

break;

case 5:

return $this->getMegjelenesiEsetekSzama();

break;

case 6:

return $this->getNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama();

break;

case 7:

return $this->getNappaliEllatasbanReszesuloIdoskoruakSzama();

break;

case 8:

return $this->getNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama();

break;

case 9:

return $this->getNappaliEllatasbanReszesuloSzenvedelybetegekSzama();

break;

case 10:

return $this->getOdavandorlasokSzama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgEgeszsegugyiSzocialisEllatottakElvandoroltak'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgEgeszsegugyiSzocialisEllatottakElvandoroltak'][$this->getPrimaryKey()] = true;

$keys = StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama(),

$keys[3] => $this->getHaziOrvosiEllatasbanRendelesenMegjelentekSzama(),

$keys[4] => $this->getElbocsatottBetegekSzama(),

$keys[5] => $this->getMegjelenesiEsetekSzama(),

$keys[6] => $this->getNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama(),

$keys[7] => $this->getNappaliEllatasbanReszesuloIdoskoruakSzama(),

$keys[8] => $this->getNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama(),

$keys[9] => $this->getNappaliEllatasbanReszesuloSzenvedelybetegekSzama(),

$keys[10] => $this->getOdavandorlasokSzama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgEgeszsegugyiSzocialisEllatottakElvandoroltak

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgEgeszsegugyiSzocialisEllatottakElvandoroltak

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama($value);

break;

case 3:

$this->setHaziOrvosiEllatasbanRendelesenMegjelentekSzama($value);

break;

case 4:

$this->setElbocsatottBetegekSzama($value);

break;

case 5:

$this->setMegjelenesiEsetekSzama($value);

break;

case 6:

$this->setNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama($value);

break;

case 7:

$this->setNappaliEllatasbanReszesuloIdoskoruakSzama($value);

break;

case 8:

$this->setNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama($value);

break;

case 9:

$this->setNappaliEllatasbanReszesuloSzenvedelybetegekSzama($value);

break;

case 10:

$this->setOdavandorlasokSzama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setHaziOrvosiEllatasbanRendelesenMegjelentekSzama($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setElbocsatottBetegekSzama($arr[$keys[4]]);

}

if (array\_key\_exists($keys[5], $arr)) {

$this->setMegjelenesiEsetekSzama($arr[$keys[5]]);

}

if (array\_key\_exists($keys[6], $arr)) {

$this->setNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama($arr[$keys[6]]);

}

if (array\_key\_exists($keys[7], $arr)) {

$this->setNappaliEllatasbanReszesuloIdoskoruakSzama($arr[$keys[7]]);

}

if (array\_key\_exists($keys[8], $arr)) {

$this->setNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama($arr[$keys[8]]);

}

if (array\_key\_exists($keys[9], $arr)) {

$this->setNappaliEllatasbanReszesuloSzenvedelybetegekSzama($arr[$keys[9]]);

}

if (array\_key\_exists($keys[10], $arr)) {

$this->setOdavandorlasokSzama($arr[$keys[10]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgEgeszsegugyiSzocialisEllatottakElvandoroltak The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA)) {

$criteria->add(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA, $this->hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama);

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA)) {

$criteria->add(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA, $this->hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama);

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ELBOCSATOTT\_BETEGEK\_SZAMA)) {

$criteria->add(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ELBOCSATOTT\_BETEGEK\_SZAMA, $this->elbocsatott\_betegek\_szama);

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_MEGJELENESI\_ESETEK\_SZAMA)) {

$criteria->add(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_MEGJELENESI\_ESETEK\_SZAMA, $this->megjelenesi\_esetek\_szama);

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA)) {

$criteria->add(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA, $this->nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama);

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA)) {

$criteria->add(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA, $this->nappali\_ellatasban\_reszesulo\_idoskoruak\_szama);

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA)) {

$criteria->add(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA, $this->nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama);

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA)) {

$criteria->add(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA, $this->nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama);

}

if ($this->isColumnModified(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ODAVANDORLASOK\_SZAMA)) {

$criteria->add(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ODAVANDORLASOK\_SZAMA, $this->odavandorlasok\_szama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgEgeszsegugyiSzocialisEllatottakElvandoroltak object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgEgeszsegugyiSzocialisEllatottakElvandoroltak (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama($this->getHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama());

$copyObj->setHaziOrvosiEllatasbanRendelesenMegjelentekSzama($this->getHaziOrvosiEllatasbanRendelesenMegjelentekSzama());

$copyObj->setElbocsatottBetegekSzama($this->getElbocsatottBetegekSzama());

$copyObj->setMegjelenesiEsetekSzama($this->getMegjelenesiEsetekSzama());

$copyObj->setNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama($this->getNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama());

$copyObj->setNappaliEllatasbanReszesuloIdoskoruakSzama($this->getNappaliEllatasbanReszesuloIdoskoruakSzama());

$copyObj->setNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama($this->getNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama());

$copyObj->setNappaliEllatasbanReszesuloSzenvedelybetegekSzama($this->getNappaliEllatasbanReszesuloSzenvedelybetegekSzama());

$copyObj->setOdavandorlasokSzama($this->getOdavandorlasokSzama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgEgeszsegugyiSzocialisEllatottakElvandoroltak Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama = null;

$this->hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama = null;

$this->elbocsatott\_betegek\_szama = null;

$this->megjelenesi\_esetek\_szama = null;

$this->nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama = null;

$this->nappali\_ellatasban\_reszesulo\_idoskoruak\_szama = null;

$this->nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama = null;

$this->nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama = null;

$this->odavandorlasok\_szama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery.php

<?php

namespace Base;

use \StgEgeszsegugyiSzocialisEllatottakElvandoroltak as ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak;

use \StgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery as ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery;

use \Exception;

use Map\StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak' table.

\*

\*

\*

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery orderByHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama($order = Criteria::ASC) Order by the hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery orderByHaziOrvosiEllatasbanRendelesenMegjelentekSzama($order = Criteria::ASC) Order by the hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery orderByElbocsatottBetegekSzama($order = Criteria::ASC) Order by the elbocsatott\_betegek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery orderByMegjelenesiEsetekSzama($order = Criteria::ASC) Order by the megjelenesi\_esetek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery orderByNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama($order = Criteria::ASC) Order by the nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery orderByNappaliEllatasbanReszesuloIdoskoruakSzama($order = Criteria::ASC) Order by the nappali\_ellatasban\_reszesulo\_idoskoruak\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery orderByNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama($order = Criteria::ASC) Order by the nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery orderByNappaliEllatasbanReszesuloSzenvedelybetegekSzama($order = Criteria::ASC) Order by the nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery orderByOdavandorlasokSzama($order = Criteria::ASC) Order by the odavandorlasok\_szama column

\*

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery groupByHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama() Group by the hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery groupByHaziOrvosiEllatasbanRendelesenMegjelentekSzama() Group by the hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery groupByElbocsatottBetegekSzama() Group by the elbocsatott\_betegek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery groupByMegjelenesiEsetekSzama() Group by the megjelenesi\_esetek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery groupByNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama() Group by the nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery groupByNappaliEllatasbanReszesuloIdoskoruakSzama() Group by the nappali\_ellatasban\_reszesulo\_idoskoruak\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery groupByNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama() Group by the nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery groupByNappaliEllatasbanReszesuloSzenvedelybetegekSzama() Group by the nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery groupByOdavandorlasokSzama() Group by the odavandorlasok\_szama column

\*

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak findOne(ConnectionInterface $con = null) Return the first ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak matching the query

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak matching the query, or a new ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak object populated from the query conditions when no match is found

\*

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak filtered by the telepules\_nev column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak filtered by the telepules\_KSHKOD column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak findOneByHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama(int $hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama) Return the first ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak filtered by the hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak findOneByHaziOrvosiEllatasbanRendelesenMegjelentekSzama(int $hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama) Return the first ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak filtered by the hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak findOneByElbocsatottBetegekSzama(int $elbocsatott\_betegek\_szama) Return the first ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak filtered by the elbocsatott\_betegek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak findOneByMegjelenesiEsetekSzama(int $megjelenesi\_esetek\_szama) Return the first ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak filtered by the megjelenesi\_esetek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak findOneByNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama(int $nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama) Return the first ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak filtered by the nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak findOneByNappaliEllatasbanReszesuloIdoskoruakSzama(int $nappali\_ellatasban\_reszesulo\_idoskoruak\_szama) Return the first ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak filtered by the nappali\_ellatasban\_reszesulo\_idoskoruak\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak findOneByNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama(int $nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama) Return the first ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak filtered by the nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak findOneByNappaliEllatasbanReszesuloSzenvedelybetegekSzama(int $nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama) Return the first ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak filtered by the nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak findOneByOdavandorlasokSzama(int $odavandorlasok\_szama) Return the first ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak filtered by the odavandorlasok\_szama column

\*

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak objects based on current ModelCriteria

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak objects filtered by the telepules\_nev column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak objects filtered by the telepules\_KSHKOD column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak[]|ObjectCollection findByHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama(int $hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama) Return ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak objects filtered by the hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak[]|ObjectCollection findByHaziOrvosiEllatasbanRendelesenMegjelentekSzama(int $hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama) Return ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak objects filtered by the hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak[]|ObjectCollection findByElbocsatottBetegekSzama(int $elbocsatott\_betegek\_szama) Return ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak objects filtered by the elbocsatott\_betegek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak[]|ObjectCollection findByMegjelenesiEsetekSzama(int $megjelenesi\_esetek\_szama) Return ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak objects filtered by the megjelenesi\_esetek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak[]|ObjectCollection findByNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama(int $nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama) Return ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak objects filtered by the nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak[]|ObjectCollection findByNappaliEllatasbanReszesuloIdoskoruakSzama(int $nappali\_ellatasban\_reszesulo\_idoskoruak\_szama) Return ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak objects filtered by the nappali\_ellatasban\_reszesulo\_idoskoruak\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak[]|ObjectCollection findByNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama(int $nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama) Return ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak objects filtered by the nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak[]|ObjectCollection findByNappaliEllatasbanReszesuloSzenvedelybetegekSzama(int $nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama) Return ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak objects filtered by the nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak[]|ObjectCollection findByOdavandorlasokSzama(int $odavandorlasok\_szama) Return ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak objects filtered by the odavandorlasok\_szama column

\* @method ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgEgeszsegugyiSzocialisEllatottakElvandoroltak', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery) {

return $criteria;

}

$query = new ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgEgeszsegugyiSzocialisEllatottakElvandoroltak object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgEgeszsegugyiSzocialisEllatottakElvandoroltak object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgEgeszsegugyiSzocialisEllatottakElvandoroltak object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgEgeszsegugyiSzocialisEllatottakElvandoroltak object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama(1234); // WHERE hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama = 1234

\* $query->filterByHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama(array(12, 34)); // WHERE hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama IN (12, 34)

\* $query->filterByHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama(array('min' => 12)); // WHERE hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama > 12

\* </code>

\*

\* @param mixed $haziGyermekorvosiEllatasbanRendelesenMegjelentekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery The current query, for fluid interface

\*/

public function filterByHaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama($haziGyermekorvosiEllatasbanRendelesenMegjelentekSzama = null, $comparison = null)

{

if (is\_array($haziGyermekorvosiEllatasbanRendelesenMegjelentekSzama)) {

$useMinMax = false;

if (isset($haziGyermekorvosiEllatasbanRendelesenMegjelentekSzama['min'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA, $haziGyermekorvosiEllatasbanRendelesenMegjelentekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($haziGyermekorvosiEllatasbanRendelesenMegjelentekSzama['max'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA, $haziGyermekorvosiEllatasbanRendelesenMegjelentekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA, $haziGyermekorvosiEllatasbanRendelesenMegjelentekSzama, $comparison);

}

/\*\*

\* Filter the query on the hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByHaziOrvosiEllatasbanRendelesenMegjelentekSzama(1234); // WHERE hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama = 1234

\* $query->filterByHaziOrvosiEllatasbanRendelesenMegjelentekSzama(array(12, 34)); // WHERE hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama IN (12, 34)

\* $query->filterByHaziOrvosiEllatasbanRendelesenMegjelentekSzama(array('min' => 12)); // WHERE hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama > 12

\* </code>

\*

\* @param mixed $haziOrvosiEllatasbanRendelesenMegjelentekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery The current query, for fluid interface

\*/

public function filterByHaziOrvosiEllatasbanRendelesenMegjelentekSzama($haziOrvosiEllatasbanRendelesenMegjelentekSzama = null, $comparison = null)

{

if (is\_array($haziOrvosiEllatasbanRendelesenMegjelentekSzama)) {

$useMinMax = false;

if (isset($haziOrvosiEllatasbanRendelesenMegjelentekSzama['min'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA, $haziOrvosiEllatasbanRendelesenMegjelentekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($haziOrvosiEllatasbanRendelesenMegjelentekSzama['max'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA, $haziOrvosiEllatasbanRendelesenMegjelentekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA, $haziOrvosiEllatasbanRendelesenMegjelentekSzama, $comparison);

}

/\*\*

\* Filter the query on the elbocsatott\_betegek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByElbocsatottBetegekSzama(1234); // WHERE elbocsatott\_betegek\_szama = 1234

\* $query->filterByElbocsatottBetegekSzama(array(12, 34)); // WHERE elbocsatott\_betegek\_szama IN (12, 34)

\* $query->filterByElbocsatottBetegekSzama(array('min' => 12)); // WHERE elbocsatott\_betegek\_szama > 12

\* </code>

\*

\* @param mixed $elbocsatottBetegekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery The current query, for fluid interface

\*/

public function filterByElbocsatottBetegekSzama($elbocsatottBetegekSzama = null, $comparison = null)

{

if (is\_array($elbocsatottBetegekSzama)) {

$useMinMax = false;

if (isset($elbocsatottBetegekSzama['min'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ELBOCSATOTT\_BETEGEK\_SZAMA, $elbocsatottBetegekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($elbocsatottBetegekSzama['max'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ELBOCSATOTT\_BETEGEK\_SZAMA, $elbocsatottBetegekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ELBOCSATOTT\_BETEGEK\_SZAMA, $elbocsatottBetegekSzama, $comparison);

}

/\*\*

\* Filter the query on the megjelenesi\_esetek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByMegjelenesiEsetekSzama(1234); // WHERE megjelenesi\_esetek\_szama = 1234

\* $query->filterByMegjelenesiEsetekSzama(array(12, 34)); // WHERE megjelenesi\_esetek\_szama IN (12, 34)

\* $query->filterByMegjelenesiEsetekSzama(array('min' => 12)); // WHERE megjelenesi\_esetek\_szama > 12

\* </code>

\*

\* @param mixed $megjelenesiEsetekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery The current query, for fluid interface

\*/

public function filterByMegjelenesiEsetekSzama($megjelenesiEsetekSzama = null, $comparison = null)

{

if (is\_array($megjelenesiEsetekSzama)) {

$useMinMax = false;

if (isset($megjelenesiEsetekSzama['min'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_MEGJELENESI\_ESETEK\_SZAMA, $megjelenesiEsetekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($megjelenesiEsetekSzama['max'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_MEGJELENESI\_ESETEK\_SZAMA, $megjelenesiEsetekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_MEGJELENESI\_ESETEK\_SZAMA, $megjelenesiEsetekSzama, $comparison);

}

/\*\*

\* Filter the query on the nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama(1234); // WHERE nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama = 1234

\* $query->filterByNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama(array(12, 34)); // WHERE nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama IN (12, 34)

\* $query->filterByNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama(array('min' => 12)); // WHERE nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama > 12

\* </code>

\*

\* @param mixed $nappaliEllatasbanReszesuloFogyatekosSzemelyekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery The current query, for fluid interface

\*/

public function filterByNappaliEllatasbanReszesuloFogyatekosSzemelyekSzama($nappaliEllatasbanReszesuloFogyatekosSzemelyekSzama = null, $comparison = null)

{

if (is\_array($nappaliEllatasbanReszesuloFogyatekosSzemelyekSzama)) {

$useMinMax = false;

if (isset($nappaliEllatasbanReszesuloFogyatekosSzemelyekSzama['min'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA, $nappaliEllatasbanReszesuloFogyatekosSzemelyekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliEllatasbanReszesuloFogyatekosSzemelyekSzama['max'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA, $nappaliEllatasbanReszesuloFogyatekosSzemelyekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA, $nappaliEllatasbanReszesuloFogyatekosSzemelyekSzama, $comparison);

}

/\*\*

\* Filter the query on the nappali\_ellatasban\_reszesulo\_idoskoruak\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliEllatasbanReszesuloIdoskoruakSzama(1234); // WHERE nappali\_ellatasban\_reszesulo\_idoskoruak\_szama = 1234

\* $query->filterByNappaliEllatasbanReszesuloIdoskoruakSzama(array(12, 34)); // WHERE nappali\_ellatasban\_reszesulo\_idoskoruak\_szama IN (12, 34)

\* $query->filterByNappaliEllatasbanReszesuloIdoskoruakSzama(array('min' => 12)); // WHERE nappali\_ellatasban\_reszesulo\_idoskoruak\_szama > 12

\* </code>

\*

\* @param mixed $nappaliEllatasbanReszesuloIdoskoruakSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery The current query, for fluid interface

\*/

public function filterByNappaliEllatasbanReszesuloIdoskoruakSzama($nappaliEllatasbanReszesuloIdoskoruakSzama = null, $comparison = null)

{

if (is\_array($nappaliEllatasbanReszesuloIdoskoruakSzama)) {

$useMinMax = false;

if (isset($nappaliEllatasbanReszesuloIdoskoruakSzama['min'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA, $nappaliEllatasbanReszesuloIdoskoruakSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliEllatasbanReszesuloIdoskoruakSzama['max'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA, $nappaliEllatasbanReszesuloIdoskoruakSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA, $nappaliEllatasbanReszesuloIdoskoruakSzama, $comparison);

}

/\*\*

\* Filter the query on the nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama(1234); // WHERE nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama = 1234

\* $query->filterByNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama(array(12, 34)); // WHERE nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama IN (12, 34)

\* $query->filterByNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama(array('min' => 12)); // WHERE nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama > 12

\* </code>

\*

\* @param mixed $nappaliEllatasbanReszesuloPszichiatriaiBetegekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery The current query, for fluid interface

\*/

public function filterByNappaliEllatasbanReszesuloPszichiatriaiBetegekSzama($nappaliEllatasbanReszesuloPszichiatriaiBetegekSzama = null, $comparison = null)

{

if (is\_array($nappaliEllatasbanReszesuloPszichiatriaiBetegekSzama)) {

$useMinMax = false;

if (isset($nappaliEllatasbanReszesuloPszichiatriaiBetegekSzama['min'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA, $nappaliEllatasbanReszesuloPszichiatriaiBetegekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliEllatasbanReszesuloPszichiatriaiBetegekSzama['max'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA, $nappaliEllatasbanReszesuloPszichiatriaiBetegekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA, $nappaliEllatasbanReszesuloPszichiatriaiBetegekSzama, $comparison);

}

/\*\*

\* Filter the query on the nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliEllatasbanReszesuloSzenvedelybetegekSzama(1234); // WHERE nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama = 1234

\* $query->filterByNappaliEllatasbanReszesuloSzenvedelybetegekSzama(array(12, 34)); // WHERE nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama IN (12, 34)

\* $query->filterByNappaliEllatasbanReszesuloSzenvedelybetegekSzama(array('min' => 12)); // WHERE nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama > 12

\* </code>

\*

\* @param mixed $nappaliEllatasbanReszesuloSzenvedelybetegekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery The current query, for fluid interface

\*/

public function filterByNappaliEllatasbanReszesuloSzenvedelybetegekSzama($nappaliEllatasbanReszesuloSzenvedelybetegekSzama = null, $comparison = null)

{

if (is\_array($nappaliEllatasbanReszesuloSzenvedelybetegekSzama)) {

$useMinMax = false;

if (isset($nappaliEllatasbanReszesuloSzenvedelybetegekSzama['min'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA, $nappaliEllatasbanReszesuloSzenvedelybetegekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliEllatasbanReszesuloSzenvedelybetegekSzama['max'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA, $nappaliEllatasbanReszesuloSzenvedelybetegekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA, $nappaliEllatasbanReszesuloSzenvedelybetegekSzama, $comparison);

}

/\*\*

\* Filter the query on the odavandorlasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByOdavandorlasokSzama(1234); // WHERE odavandorlasok\_szama = 1234

\* $query->filterByOdavandorlasokSzama(array(12, 34)); // WHERE odavandorlasok\_szama IN (12, 34)

\* $query->filterByOdavandorlasokSzama(array('min' => 12)); // WHERE odavandorlasok\_szama > 12

\* </code>

\*

\* @param mixed $odavandorlasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery The current query, for fluid interface

\*/

public function filterByOdavandorlasokSzama($odavandorlasokSzama = null, $comparison = null)

{

if (is\_array($odavandorlasokSzama)) {

$useMinMax = false;

if (isset($odavandorlasokSzama['min'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ODAVANDORLASOK\_SZAMA, $odavandorlasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($odavandorlasokSzama['max'])) {

$this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ODAVANDORLASOK\_SZAMA, $odavandorlasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ODAVANDORLASOK\_SZAMA, $odavandorlasokSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltak $stgEgeszsegugyiSzocialisEllatottakElvandoroltak Object to remove from the list of results

\*

\* @return $this|ChildStgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery The current query, for fluid interface

\*/

public function prune($stgEgeszsegugyiSzocialisEllatottakElvandoroltak = null)

{

if ($stgEgeszsegugyiSzocialisEllatottakElvandoroltak) {

throw new LogicException('StgEgeszsegugyiSzocialisEllatottakElvandoroltak object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::clearInstancePool();

StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery

#### StgEgyebIntezmenyiEllatottsagSulya.php

<?php

namespace Base;

use \StgEgyebIntezmenyiEllatottsagSulyaQuery as ChildStgEgyebIntezmenyiEllatottsagSulyaQuery;

use \Exception;

use \PDO;

use Map\StgEgyebIntezmenyiEllatottsagSulyaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgEgyebIntezmenyiEllatottsagSulya implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgEgyebIntezmenyiEllatottsagSulyaTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the bankfiok\_lete field.

\* @var boolean

\*/

protected $bankfiok\_lete;

/\*\*

\* The value for the benzinkut\_lete field.

\* @var boolean

\*/

protected $benzinkut\_lete;

/\*\*

\* The value for the korjegyzoseg\_szekhelye field.

\* @var boolean

\*/

protected $korjegyzoseg\_szekhelye;

/\*\*

\* The value for the postahivatal\_lete field.

\* @var boolean

\*/

protected $postahivatal\_lete;

/\*\*

\* The value for the birosag\_ugyeszseg\_lete field.

\* @var boolean

\*/

protected $birosag\_ugyeszseg\_lete;

/\*\*

\* The value for the okmanyiroda\_lete field.

\* @var boolean

\*/

protected $okmanyiroda\_lete;

/\*\*

\* The value for the megyei\_foldhivatal\_illetve\_kirendeltseg\_lete field.

\* @var boolean

\*/

protected $megyei\_foldhivatal\_illetve\_kirendeltseg\_lete;

/\*\*

\* The value for the munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete field.

\* @var boolean

\*/

protected $munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete;

/\*\*

\* The value for the vodafone\_uzlet\_lete field.

\* @var boolean

\*/

protected $vodafone\_uzlet\_lete;

/\*\*

\* The value for the t\_mobile\_uzlet\_lete field.

\* @var boolean

\*/

protected $t\_mobile\_uzlet\_lete;

/\*\*

\* The value for the telenor\_uzlet\_lete field.

\* @var boolean

\*/

protected $telenor\_uzlet\_lete;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgEgyebIntezmenyiEllatottsagSulya object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgEgyebIntezmenyiEllatottsagSulya</code> instance. If

\* <code>obj</code> is an instance of <code>StgEgyebIntezmenyiEllatottsagSulya</code>, delegates to

\* <code>equals(StgEgyebIntezmenyiEllatottsagSulya)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgEgyebIntezmenyiEllatottsagSulya The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [bankfiok\_lete] column value.

\*

\* @return boolean

\*/

public function getBankfiokLete()

{

return $this->bankfiok\_lete;

}

/\*\*

\* Get the [bankfiok\_lete] column value.

\*

\* @return boolean

\*/

public function isBankfiokLete()

{

return $this->getBankfiokLete();

}

/\*\*

\* Get the [benzinkut\_lete] column value.

\*

\* @return boolean

\*/

public function getBenzinkutLete()

{

return $this->benzinkut\_lete;

}

/\*\*

\* Get the [benzinkut\_lete] column value.

\*

\* @return boolean

\*/

public function isBenzinkutLete()

{

return $this->getBenzinkutLete();

}

/\*\*

\* Get the [korjegyzoseg\_szekhelye] column value.

\*

\* @return boolean

\*/

public function getKorjegyzosegSzekhelye()

{

return $this->korjegyzoseg\_szekhelye;

}

/\*\*

\* Get the [korjegyzoseg\_szekhelye] column value.

\*

\* @return boolean

\*/

public function isKorjegyzosegSzekhelye()

{

return $this->getKorjegyzosegSzekhelye();

}

/\*\*

\* Get the [postahivatal\_lete] column value.

\*

\* @return boolean

\*/

public function getPostahivatalLete()

{

return $this->postahivatal\_lete;

}

/\*\*

\* Get the [postahivatal\_lete] column value.

\*

\* @return boolean

\*/

public function isPostahivatalLete()

{

return $this->getPostahivatalLete();

}

/\*\*

\* Get the [birosag\_ugyeszseg\_lete] column value.

\*

\* @return boolean

\*/

public function getBirosagUgyeszsegLete()

{

return $this->birosag\_ugyeszseg\_lete;

}

/\*\*

\* Get the [birosag\_ugyeszseg\_lete] column value.

\*

\* @return boolean

\*/

public function isBirosagUgyeszsegLete()

{

return $this->getBirosagUgyeszsegLete();

}

/\*\*

\* Get the [okmanyiroda\_lete] column value.

\*

\* @return boolean

\*/

public function getOkmanyirodaLete()

{

return $this->okmanyiroda\_lete;

}

/\*\*

\* Get the [okmanyiroda\_lete] column value.

\*

\* @return boolean

\*/

public function isOkmanyirodaLete()

{

return $this->getOkmanyirodaLete();

}

/\*\*

\* Get the [megyei\_foldhivatal\_illetve\_kirendeltseg\_lete] column value.

\*

\* @return boolean

\*/

public function getMegyeiFoldhivatalIlletveKirendeltsegLete()

{

return $this->megyei\_foldhivatal\_illetve\_kirendeltseg\_lete;

}

/\*\*

\* Get the [megyei\_foldhivatal\_illetve\_kirendeltseg\_lete] column value.

\*

\* @return boolean

\*/

public function isMegyeiFoldhivatalIlletveKirendeltsegLete()

{

return $this->getMegyeiFoldhivatalIlletveKirendeltsegLete();

}

/\*\*

\* Get the [munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete] column value.

\*

\* @return boolean

\*/

public function getMunkaugyiKozpontIlletveKirendeltsegLete()

{

return $this->munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete;

}

/\*\*

\* Get the [munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete] column value.

\*

\* @return boolean

\*/

public function isMunkaugyiKozpontIlletveKirendeltsegLete()

{

return $this->getMunkaugyiKozpontIlletveKirendeltsegLete();

}

/\*\*

\* Get the [vodafone\_uzlet\_lete] column value.

\*

\* @return boolean

\*/

public function getVodafoneUzletLete()

{

return $this->vodafone\_uzlet\_lete;

}

/\*\*

\* Get the [vodafone\_uzlet\_lete] column value.

\*

\* @return boolean

\*/

public function isVodafoneUzletLete()

{

return $this->getVodafoneUzletLete();

}

/\*\*

\* Get the [t\_mobile\_uzlet\_lete] column value.

\*

\* @return boolean

\*/

public function getTMobileUzletLete()

{

return $this->t\_mobile\_uzlet\_lete;

}

/\*\*

\* Get the [t\_mobile\_uzlet\_lete] column value.

\*

\* @return boolean

\*/

public function isTMobileUzletLete()

{

return $this->getTMobileUzletLete();

}

/\*\*

\* Get the [telenor\_uzlet\_lete] column value.

\*

\* @return boolean

\*/

public function getTelenorUzletLete()

{

return $this->telenor\_uzlet\_lete;

}

/\*\*

\* Get the [telenor\_uzlet\_lete] column value.

\*

\* @return boolean

\*/

public function isTelenorUzletLete()

{

return $this->getTelenorUzletLete();

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgEgyebIntezmenyiEllatottsagSulyaTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgEgyebIntezmenyiEllatottsagSulyaTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgEgyebIntezmenyiEllatottsagSulyaTableMap::translateFieldName('BankfiokLete', TableMap::TYPE\_PHPNAME, $indexType)];

$this->bankfiok\_lete = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgEgyebIntezmenyiEllatottsagSulyaTableMap::translateFieldName('BenzinkutLete', TableMap::TYPE\_PHPNAME, $indexType)];

$this->benzinkut\_lete = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgEgyebIntezmenyiEllatottsagSulyaTableMap::translateFieldName('KorjegyzosegSzekhelye', TableMap::TYPE\_PHPNAME, $indexType)];

$this->korjegyzoseg\_szekhelye = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 5 + $startcol : StgEgyebIntezmenyiEllatottsagSulyaTableMap::translateFieldName('PostahivatalLete', TableMap::TYPE\_PHPNAME, $indexType)];

$this->postahivatal\_lete = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 6 + $startcol : StgEgyebIntezmenyiEllatottsagSulyaTableMap::translateFieldName('BirosagUgyeszsegLete', TableMap::TYPE\_PHPNAME, $indexType)];

$this->birosag\_ugyeszseg\_lete = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 7 + $startcol : StgEgyebIntezmenyiEllatottsagSulyaTableMap::translateFieldName('OkmanyirodaLete', TableMap::TYPE\_PHPNAME, $indexType)];

$this->okmanyiroda\_lete = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 8 + $startcol : StgEgyebIntezmenyiEllatottsagSulyaTableMap::translateFieldName('MegyeiFoldhivatalIlletveKirendeltsegLete', TableMap::TYPE\_PHPNAME, $indexType)];

$this->megyei\_foldhivatal\_illetve\_kirendeltseg\_lete = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 9 + $startcol : StgEgyebIntezmenyiEllatottsagSulyaTableMap::translateFieldName('MunkaugyiKozpontIlletveKirendeltsegLete', TableMap::TYPE\_PHPNAME, $indexType)];

$this->munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 10 + $startcol : StgEgyebIntezmenyiEllatottsagSulyaTableMap::translateFieldName('VodafoneUzletLete', TableMap::TYPE\_PHPNAME, $indexType)];

$this->vodafone\_uzlet\_lete = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 11 + $startcol : StgEgyebIntezmenyiEllatottsagSulyaTableMap::translateFieldName('TMobileUzletLete', TableMap::TYPE\_PHPNAME, $indexType)];

$this->t\_mobile\_uzlet\_lete = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 12 + $startcol : StgEgyebIntezmenyiEllatottsagSulyaTableMap::translateFieldName('TelenorUzletLete', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telenor\_uzlet\_lete = (null !== $col) ? (boolean) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 13; // 13 = StgEgyebIntezmenyiEllatottsagSulyaTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgEgyebIntezmenyiEllatottsagSulya'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgEgyebIntezmenyiEllatottsagSulya The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgEgyebIntezmenyiEllatottsagSulya The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Sets the value of the [bankfiok\_lete] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgEgyebIntezmenyiEllatottsagSulya The current object (for fluent API support)

\*/

public function setBankfiokLete($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->bankfiok\_lete !== $v) {

$this->bankfiok\_lete = $v;

$this->modifiedColumns[StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BANKFIOK\_LETE] = true;

}

return $this;

} // setBankfiokLete()

/\*\*

\* Sets the value of the [benzinkut\_lete] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgEgyebIntezmenyiEllatottsagSulya The current object (for fluent API support)

\*/

public function setBenzinkutLete($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->benzinkut\_lete !== $v) {

$this->benzinkut\_lete = $v;

$this->modifiedColumns[StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BENZINKUT\_LETE] = true;

}

return $this;

} // setBenzinkutLete()

/\*\*

\* Sets the value of the [korjegyzoseg\_szekhelye] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgEgyebIntezmenyiEllatottsagSulya The current object (for fluent API support)

\*/

public function setKorjegyzosegSzekhelye($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->korjegyzoseg\_szekhelye !== $v) {

$this->korjegyzoseg\_szekhelye = $v;

$this->modifiedColumns[StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_KORJEGYZOSEG\_SZEKHELYE] = true;

}

return $this;

} // setKorjegyzosegSzekhelye()

/\*\*

\* Sets the value of the [postahivatal\_lete] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgEgyebIntezmenyiEllatottsagSulya The current object (for fluent API support)

\*/

public function setPostahivatalLete($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->postahivatal\_lete !== $v) {

$this->postahivatal\_lete = $v;

$this->modifiedColumns[StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_POSTAHIVATAL\_LETE] = true;

}

return $this;

} // setPostahivatalLete()

/\*\*

\* Sets the value of the [birosag\_ugyeszseg\_lete] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgEgyebIntezmenyiEllatottsagSulya The current object (for fluent API support)

\*/

public function setBirosagUgyeszsegLete($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->birosag\_ugyeszseg\_lete !== $v) {

$this->birosag\_ugyeszseg\_lete = $v;

$this->modifiedColumns[StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BIROSAG\_UGYESZSEG\_LETE] = true;

}

return $this;

} // setBirosagUgyeszsegLete()

/\*\*

\* Sets the value of the [okmanyiroda\_lete] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgEgyebIntezmenyiEllatottsagSulya The current object (for fluent API support)

\*/

public function setOkmanyirodaLete($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->okmanyiroda\_lete !== $v) {

$this->okmanyiroda\_lete = $v;

$this->modifiedColumns[StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_OKMANYIRODA\_LETE] = true;

}

return $this;

} // setOkmanyirodaLete()

/\*\*

\* Sets the value of the [megyei\_foldhivatal\_illetve\_kirendeltseg\_lete] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgEgyebIntezmenyiEllatottsagSulya The current object (for fluent API support)

\*/

public function setMegyeiFoldhivatalIlletveKirendeltsegLete($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->megyei\_foldhivatal\_illetve\_kirendeltseg\_lete !== $v) {

$this->megyei\_foldhivatal\_illetve\_kirendeltseg\_lete = $v;

$this->modifiedColumns[StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_MEGYEI\_FOLDHIVATAL\_ILLETVE\_KIRENDELTSEG\_LETE] = true;

}

return $this;

} // setMegyeiFoldhivatalIlletveKirendeltsegLete()

/\*\*

\* Sets the value of the [munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgEgyebIntezmenyiEllatottsagSulya The current object (for fluent API support)

\*/

public function setMunkaugyiKozpontIlletveKirendeltsegLete($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete !== $v) {

$this->munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete = $v;

$this->modifiedColumns[StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_MUNKAUGYI\_KOZPONT\_ILLETVE\_KIRENDELTSEG\_LETE] = true;

}

return $this;

} // setMunkaugyiKozpontIlletveKirendeltsegLete()

/\*\*

\* Sets the value of the [vodafone\_uzlet\_lete] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgEgyebIntezmenyiEllatottsagSulya The current object (for fluent API support)

\*/

public function setVodafoneUzletLete($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->vodafone\_uzlet\_lete !== $v) {

$this->vodafone\_uzlet\_lete = $v;

$this->modifiedColumns[StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_VODAFONE\_UZLET\_LETE] = true;

}

return $this;

} // setVodafoneUzletLete()

/\*\*

\* Sets the value of the [t\_mobile\_uzlet\_lete] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgEgyebIntezmenyiEllatottsagSulya The current object (for fluent API support)

\*/

public function setTMobileUzletLete($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->t\_mobile\_uzlet\_lete !== $v) {

$this->t\_mobile\_uzlet\_lete = $v;

$this->modifiedColumns[StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_T\_MOBILE\_UZLET\_LETE] = true;

}

return $this;

} // setTMobileUzletLete()

/\*\*

\* Sets the value of the [telenor\_uzlet\_lete] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgEgyebIntezmenyiEllatottsagSulya The current object (for fluent API support)

\*/

public function setTelenorUzletLete($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->telenor\_uzlet\_lete !== $v) {

$this->telenor\_uzlet\_lete = $v;

$this->modifiedColumns[StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELENOR\_UZLET\_LETE] = true;

}

return $this;

} // setTelenorUzletLete()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgEgyebIntezmenyiEllatottsagSulyaTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgEgyebIntezmenyiEllatottsagSulyaQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgEgyebIntezmenyiEllatottsagSulya::setDeleted()

\* @see StgEgyebIntezmenyiEllatottsagSulya::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgEgyebIntezmenyiEllatottsagSulyaTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgEgyebIntezmenyiEllatottsagSulyaQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgEgyebIntezmenyiEllatottsagSulyaTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgEgyebIntezmenyiEllatottsagSulyaTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BANKFIOK\_LETE)) {

$modifiedColumns[':p' . $index++] = 'BANKFIOK\_LETE';

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BENZINKUT\_LETE)) {

$modifiedColumns[':p' . $index++] = 'BENZINKUT\_LETE';

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_KORJEGYZOSEG\_SZEKHELYE)) {

$modifiedColumns[':p' . $index++] = 'KORJEGYZOSEG\_SZEKHELYE';

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_POSTAHIVATAL\_LETE)) {

$modifiedColumns[':p' . $index++] = 'POSTAHIVATAL\_LETE';

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BIROSAG\_UGYESZSEG\_LETE)) {

$modifiedColumns[':p' . $index++] = 'BIROSAG\_UGYESZSEG\_LETE';

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_OKMANYIRODA\_LETE)) {

$modifiedColumns[':p' . $index++] = 'OKMANYIRODA\_LETE';

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_MEGYEI\_FOLDHIVATAL\_ILLETVE\_KIRENDELTSEG\_LETE)) {

$modifiedColumns[':p' . $index++] = 'MEGYEI\_FOLDHIVATAL\_ILLETVE\_KIRENDELTSEG\_LETE';

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_MUNKAUGYI\_KOZPONT\_ILLETVE\_KIRENDELTSEG\_LETE)) {

$modifiedColumns[':p' . $index++] = 'MUNKAUGYI\_KOZPONT\_ILLETVE\_KIRENDELTSEG\_LETE';

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_VODAFONE\_UZLET\_LETE)) {

$modifiedColumns[':p' . $index++] = 'VODAFONE\_UZLET\_LETE';

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_T\_MOBILE\_UZLET\_LETE)) {

$modifiedColumns[':p' . $index++] = 'T\_MOBILE\_UZLET\_LETE';

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELENOR\_UZLET\_LETE)) {

$modifiedColumns[':p' . $index++] = 'TELENOR\_UZLET\_LETE';

}

$sql = sprintf(

'INSERT INTO stg\_egyeb\_intezmenyi\_ellatottsag\_sulya (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'BANKFIOK\_LETE':

$stmt->bindValue($identifier, (int) $this->bankfiok\_lete, PDO::PARAM\_INT);

break;

case 'BENZINKUT\_LETE':

$stmt->bindValue($identifier, (int) $this->benzinkut\_lete, PDO::PARAM\_INT);

break;

case 'KORJEGYZOSEG\_SZEKHELYE':

$stmt->bindValue($identifier, (int) $this->korjegyzoseg\_szekhelye, PDO::PARAM\_INT);

break;

case 'POSTAHIVATAL\_LETE':

$stmt->bindValue($identifier, (int) $this->postahivatal\_lete, PDO::PARAM\_INT);

break;

case 'BIROSAG\_UGYESZSEG\_LETE':

$stmt->bindValue($identifier, (int) $this->birosag\_ugyeszseg\_lete, PDO::PARAM\_INT);

break;

case 'OKMANYIRODA\_LETE':

$stmt->bindValue($identifier, (int) $this->okmanyiroda\_lete, PDO::PARAM\_INT);

break;

case 'MEGYEI\_FOLDHIVATAL\_ILLETVE\_KIRENDELTSEG\_LETE':

$stmt->bindValue($identifier, (int) $this->megyei\_foldhivatal\_illetve\_kirendeltseg\_lete, PDO::PARAM\_INT);

break;

case 'MUNKAUGYI\_KOZPONT\_ILLETVE\_KIRENDELTSEG\_LETE':

$stmt->bindValue($identifier, (int) $this->munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete, PDO::PARAM\_INT);

break;

case 'VODAFONE\_UZLET\_LETE':

$stmt->bindValue($identifier, (int) $this->vodafone\_uzlet\_lete, PDO::PARAM\_INT);

break;

case 'T\_MOBILE\_UZLET\_LETE':

$stmt->bindValue($identifier, (int) $this->t\_mobile\_uzlet\_lete, PDO::PARAM\_INT);

break;

case 'TELENOR\_UZLET\_LETE':

$stmt->bindValue($identifier, (int) $this->telenor\_uzlet\_lete, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgEgyebIntezmenyiEllatottsagSulyaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getBankfiokLete();

break;

case 3:

return $this->getBenzinkutLete();

break;

case 4:

return $this->getKorjegyzosegSzekhelye();

break;

case 5:

return $this->getPostahivatalLete();

break;

case 6:

return $this->getBirosagUgyeszsegLete();

break;

case 7:

return $this->getOkmanyirodaLete();

break;

case 8:

return $this->getMegyeiFoldhivatalIlletveKirendeltsegLete();

break;

case 9:

return $this->getMunkaugyiKozpontIlletveKirendeltsegLete();

break;

case 10:

return $this->getVodafoneUzletLete();

break;

case 11:

return $this->getTMobileUzletLete();

break;

case 12:

return $this->getTelenorUzletLete();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgEgyebIntezmenyiEllatottsagSulya'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgEgyebIntezmenyiEllatottsagSulya'][$this->getPrimaryKey()] = true;

$keys = StgEgyebIntezmenyiEllatottsagSulyaTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getBankfiokLete(),

$keys[3] => $this->getBenzinkutLete(),

$keys[4] => $this->getKorjegyzosegSzekhelye(),

$keys[5] => $this->getPostahivatalLete(),

$keys[6] => $this->getBirosagUgyeszsegLete(),

$keys[7] => $this->getOkmanyirodaLete(),

$keys[8] => $this->getMegyeiFoldhivatalIlletveKirendeltsegLete(),

$keys[9] => $this->getMunkaugyiKozpontIlletveKirendeltsegLete(),

$keys[10] => $this->getVodafoneUzletLete(),

$keys[11] => $this->getTMobileUzletLete(),

$keys[12] => $this->getTelenorUzletLete(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgEgyebIntezmenyiEllatottsagSulya

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgEgyebIntezmenyiEllatottsagSulyaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgEgyebIntezmenyiEllatottsagSulya

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setBankfiokLete($value);

break;

case 3:

$this->setBenzinkutLete($value);

break;

case 4:

$this->setKorjegyzosegSzekhelye($value);

break;

case 5:

$this->setPostahivatalLete($value);

break;

case 6:

$this->setBirosagUgyeszsegLete($value);

break;

case 7:

$this->setOkmanyirodaLete($value);

break;

case 8:

$this->setMegyeiFoldhivatalIlletveKirendeltsegLete($value);

break;

case 9:

$this->setMunkaugyiKozpontIlletveKirendeltsegLete($value);

break;

case 10:

$this->setVodafoneUzletLete($value);

break;

case 11:

$this->setTMobileUzletLete($value);

break;

case 12:

$this->setTelenorUzletLete($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgEgyebIntezmenyiEllatottsagSulyaTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setBankfiokLete($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setBenzinkutLete($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setKorjegyzosegSzekhelye($arr[$keys[4]]);

}

if (array\_key\_exists($keys[5], $arr)) {

$this->setPostahivatalLete($arr[$keys[5]]);

}

if (array\_key\_exists($keys[6], $arr)) {

$this->setBirosagUgyeszsegLete($arr[$keys[6]]);

}

if (array\_key\_exists($keys[7], $arr)) {

$this->setOkmanyirodaLete($arr[$keys[7]]);

}

if (array\_key\_exists($keys[8], $arr)) {

$this->setMegyeiFoldhivatalIlletveKirendeltsegLete($arr[$keys[8]]);

}

if (array\_key\_exists($keys[9], $arr)) {

$this->setMunkaugyiKozpontIlletveKirendeltsegLete($arr[$keys[9]]);

}

if (array\_key\_exists($keys[10], $arr)) {

$this->setVodafoneUzletLete($arr[$keys[10]]);

}

if (array\_key\_exists($keys[11], $arr)) {

$this->setTMobileUzletLete($arr[$keys[11]]);

}

if (array\_key\_exists($keys[12], $arr)) {

$this->setTelenorUzletLete($arr[$keys[12]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgEgyebIntezmenyiEllatottsagSulya The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgEgyebIntezmenyiEllatottsagSulyaTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BANKFIOK\_LETE)) {

$criteria->add(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BANKFIOK\_LETE, $this->bankfiok\_lete);

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BENZINKUT\_LETE)) {

$criteria->add(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BENZINKUT\_LETE, $this->benzinkut\_lete);

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_KORJEGYZOSEG\_SZEKHELYE)) {

$criteria->add(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_KORJEGYZOSEG\_SZEKHELYE, $this->korjegyzoseg\_szekhelye);

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_POSTAHIVATAL\_LETE)) {

$criteria->add(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_POSTAHIVATAL\_LETE, $this->postahivatal\_lete);

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BIROSAG\_UGYESZSEG\_LETE)) {

$criteria->add(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BIROSAG\_UGYESZSEG\_LETE, $this->birosag\_ugyeszseg\_lete);

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_OKMANYIRODA\_LETE)) {

$criteria->add(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_OKMANYIRODA\_LETE, $this->okmanyiroda\_lete);

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_MEGYEI\_FOLDHIVATAL\_ILLETVE\_KIRENDELTSEG\_LETE)) {

$criteria->add(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_MEGYEI\_FOLDHIVATAL\_ILLETVE\_KIRENDELTSEG\_LETE, $this->megyei\_foldhivatal\_illetve\_kirendeltseg\_lete);

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_MUNKAUGYI\_KOZPONT\_ILLETVE\_KIRENDELTSEG\_LETE)) {

$criteria->add(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_MUNKAUGYI\_KOZPONT\_ILLETVE\_KIRENDELTSEG\_LETE, $this->munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete);

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_VODAFONE\_UZLET\_LETE)) {

$criteria->add(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_VODAFONE\_UZLET\_LETE, $this->vodafone\_uzlet\_lete);

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_T\_MOBILE\_UZLET\_LETE)) {

$criteria->add(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_T\_MOBILE\_UZLET\_LETE, $this->t\_mobile\_uzlet\_lete);

}

if ($this->isColumnModified(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELENOR\_UZLET\_LETE)) {

$criteria->add(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELENOR\_UZLET\_LETE, $this->telenor\_uzlet\_lete);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgEgyebIntezmenyiEllatottsagSulya object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgEgyebIntezmenyiEllatottsagSulya (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setBankfiokLete($this->getBankfiokLete());

$copyObj->setBenzinkutLete($this->getBenzinkutLete());

$copyObj->setKorjegyzosegSzekhelye($this->getKorjegyzosegSzekhelye());

$copyObj->setPostahivatalLete($this->getPostahivatalLete());

$copyObj->setBirosagUgyeszsegLete($this->getBirosagUgyeszsegLete());

$copyObj->setOkmanyirodaLete($this->getOkmanyirodaLete());

$copyObj->setMegyeiFoldhivatalIlletveKirendeltsegLete($this->getMegyeiFoldhivatalIlletveKirendeltsegLete());

$copyObj->setMunkaugyiKozpontIlletveKirendeltsegLete($this->getMunkaugyiKozpontIlletveKirendeltsegLete());

$copyObj->setVodafoneUzletLete($this->getVodafoneUzletLete());

$copyObj->setTMobileUzletLete($this->getTMobileUzletLete());

$copyObj->setTelenorUzletLete($this->getTelenorUzletLete());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgEgyebIntezmenyiEllatottsagSulya Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->bankfiok\_lete = null;

$this->benzinkut\_lete = null;

$this->korjegyzoseg\_szekhelye = null;

$this->postahivatal\_lete = null;

$this->birosag\_ugyeszseg\_lete = null;

$this->okmanyiroda\_lete = null;

$this->megyei\_foldhivatal\_illetve\_kirendeltseg\_lete = null;

$this->munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete = null;

$this->vodafone\_uzlet\_lete = null;

$this->t\_mobile\_uzlet\_lete = null;

$this->telenor\_uzlet\_lete = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgEgyebIntezmenyiEllatottsagSulyaTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgEgyebIntezmenyiEllatottsagSulyaQuery.php

<?php

namespace Base;

use \StgEgyebIntezmenyiEllatottsagSulya as ChildStgEgyebIntezmenyiEllatottsagSulya;

use \StgEgyebIntezmenyiEllatottsagSulyaQuery as ChildStgEgyebIntezmenyiEllatottsagSulyaQuery;

use \Exception;

use Map\StgEgyebIntezmenyiEllatottsagSulyaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_egyeb\_intezmenyi\_ellatottsag\_sulya' table.

\*

\*

\*

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery orderByBankfiokLete($order = Criteria::ASC) Order by the bankfiok\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery orderByBenzinkutLete($order = Criteria::ASC) Order by the benzinkut\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery orderByKorjegyzosegSzekhelye($order = Criteria::ASC) Order by the korjegyzoseg\_szekhelye column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery orderByPostahivatalLete($order = Criteria::ASC) Order by the postahivatal\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery orderByBirosagUgyeszsegLete($order = Criteria::ASC) Order by the birosag\_ugyeszseg\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery orderByOkmanyirodaLete($order = Criteria::ASC) Order by the okmanyiroda\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery orderByMegyeiFoldhivatalIlletveKirendeltsegLete($order = Criteria::ASC) Order by the megyei\_foldhivatal\_illetve\_kirendeltseg\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery orderByMunkaugyiKozpontIlletveKirendeltsegLete($order = Criteria::ASC) Order by the munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery orderByVodafoneUzletLete($order = Criteria::ASC) Order by the vodafone\_uzlet\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery orderByTMobileUzletLete($order = Criteria::ASC) Order by the t\_mobile\_uzlet\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery orderByTelenorUzletLete($order = Criteria::ASC) Order by the telenor\_uzlet\_lete column

\*

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery groupByBankfiokLete() Group by the bankfiok\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery groupByBenzinkutLete() Group by the benzinkut\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery groupByKorjegyzosegSzekhelye() Group by the korjegyzoseg\_szekhelye column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery groupByPostahivatalLete() Group by the postahivatal\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery groupByBirosagUgyeszsegLete() Group by the birosag\_ugyeszseg\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery groupByOkmanyirodaLete() Group by the okmanyiroda\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery groupByMegyeiFoldhivatalIlletveKirendeltsegLete() Group by the megyei\_foldhivatal\_illetve\_kirendeltseg\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery groupByMunkaugyiKozpontIlletveKirendeltsegLete() Group by the munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery groupByVodafoneUzletLete() Group by the vodafone\_uzlet\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery groupByTMobileUzletLete() Group by the t\_mobile\_uzlet\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery groupByTelenorUzletLete() Group by the telenor\_uzlet\_lete column

\*

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgEgyebIntezmenyiEllatottsagSulyaQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya findOne(ConnectionInterface $con = null) Return the first ChildStgEgyebIntezmenyiEllatottsagSulya matching the query

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgEgyebIntezmenyiEllatottsagSulya matching the query, or a new ChildStgEgyebIntezmenyiEllatottsagSulya object populated from the query conditions when no match is found

\*

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgEgyebIntezmenyiEllatottsagSulya filtered by the telepules\_nev column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgEgyebIntezmenyiEllatottsagSulya filtered by the telepules\_KSHKOD column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya findOneByBankfiokLete(boolean $bankfiok\_lete) Return the first ChildStgEgyebIntezmenyiEllatottsagSulya filtered by the bankfiok\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya findOneByBenzinkutLete(boolean $benzinkut\_lete) Return the first ChildStgEgyebIntezmenyiEllatottsagSulya filtered by the benzinkut\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya findOneByKorjegyzosegSzekhelye(boolean $korjegyzoseg\_szekhelye) Return the first ChildStgEgyebIntezmenyiEllatottsagSulya filtered by the korjegyzoseg\_szekhelye column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya findOneByPostahivatalLete(boolean $postahivatal\_lete) Return the first ChildStgEgyebIntezmenyiEllatottsagSulya filtered by the postahivatal\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya findOneByBirosagUgyeszsegLete(boolean $birosag\_ugyeszseg\_lete) Return the first ChildStgEgyebIntezmenyiEllatottsagSulya filtered by the birosag\_ugyeszseg\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya findOneByOkmanyirodaLete(boolean $okmanyiroda\_lete) Return the first ChildStgEgyebIntezmenyiEllatottsagSulya filtered by the okmanyiroda\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya findOneByMegyeiFoldhivatalIlletveKirendeltsegLete(boolean $megyei\_foldhivatal\_illetve\_kirendeltseg\_lete) Return the first ChildStgEgyebIntezmenyiEllatottsagSulya filtered by the megyei\_foldhivatal\_illetve\_kirendeltseg\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya findOneByMunkaugyiKozpontIlletveKirendeltsegLete(boolean $munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete) Return the first ChildStgEgyebIntezmenyiEllatottsagSulya filtered by the munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya findOneByVodafoneUzletLete(boolean $vodafone\_uzlet\_lete) Return the first ChildStgEgyebIntezmenyiEllatottsagSulya filtered by the vodafone\_uzlet\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya findOneByTMobileUzletLete(boolean $t\_mobile\_uzlet\_lete) Return the first ChildStgEgyebIntezmenyiEllatottsagSulya filtered by the t\_mobile\_uzlet\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya findOneByTelenorUzletLete(boolean $telenor\_uzlet\_lete) Return the first ChildStgEgyebIntezmenyiEllatottsagSulya filtered by the telenor\_uzlet\_lete column

\*

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgEgyebIntezmenyiEllatottsagSulya objects based on current ModelCriteria

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgEgyebIntezmenyiEllatottsagSulya objects filtered by the telepules\_nev column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgEgyebIntezmenyiEllatottsagSulya objects filtered by the telepules\_KSHKOD column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya[]|ObjectCollection findByBankfiokLete(boolean $bankfiok\_lete) Return ChildStgEgyebIntezmenyiEllatottsagSulya objects filtered by the bankfiok\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya[]|ObjectCollection findByBenzinkutLete(boolean $benzinkut\_lete) Return ChildStgEgyebIntezmenyiEllatottsagSulya objects filtered by the benzinkut\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya[]|ObjectCollection findByKorjegyzosegSzekhelye(boolean $korjegyzoseg\_szekhelye) Return ChildStgEgyebIntezmenyiEllatottsagSulya objects filtered by the korjegyzoseg\_szekhelye column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya[]|ObjectCollection findByPostahivatalLete(boolean $postahivatal\_lete) Return ChildStgEgyebIntezmenyiEllatottsagSulya objects filtered by the postahivatal\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya[]|ObjectCollection findByBirosagUgyeszsegLete(boolean $birosag\_ugyeszseg\_lete) Return ChildStgEgyebIntezmenyiEllatottsagSulya objects filtered by the birosag\_ugyeszseg\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya[]|ObjectCollection findByOkmanyirodaLete(boolean $okmanyiroda\_lete) Return ChildStgEgyebIntezmenyiEllatottsagSulya objects filtered by the okmanyiroda\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya[]|ObjectCollection findByMegyeiFoldhivatalIlletveKirendeltsegLete(boolean $megyei\_foldhivatal\_illetve\_kirendeltseg\_lete) Return ChildStgEgyebIntezmenyiEllatottsagSulya objects filtered by the megyei\_foldhivatal\_illetve\_kirendeltseg\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya[]|ObjectCollection findByMunkaugyiKozpontIlletveKirendeltsegLete(boolean $munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete) Return ChildStgEgyebIntezmenyiEllatottsagSulya objects filtered by the munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya[]|ObjectCollection findByVodafoneUzletLete(boolean $vodafone\_uzlet\_lete) Return ChildStgEgyebIntezmenyiEllatottsagSulya objects filtered by the vodafone\_uzlet\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya[]|ObjectCollection findByTMobileUzletLete(boolean $t\_mobile\_uzlet\_lete) Return ChildStgEgyebIntezmenyiEllatottsagSulya objects filtered by the t\_mobile\_uzlet\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya[]|ObjectCollection findByTelenorUzletLete(boolean $telenor\_uzlet\_lete) Return ChildStgEgyebIntezmenyiEllatottsagSulya objects filtered by the telenor\_uzlet\_lete column

\* @method ChildStgEgyebIntezmenyiEllatottsagSulya[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgEgyebIntezmenyiEllatottsagSulyaQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgEgyebIntezmenyiEllatottsagSulyaQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgEgyebIntezmenyiEllatottsagSulya', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgEgyebIntezmenyiEllatottsagSulyaQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgEgyebIntezmenyiEllatottsagSulyaQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgEgyebIntezmenyiEllatottsagSulyaQuery) {

return $criteria;

}

$query = new ChildStgEgyebIntezmenyiEllatottsagSulyaQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgEgyebIntezmenyiEllatottsagSulya|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgEgyebIntezmenyiEllatottsagSulya object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgEgyebIntezmenyiEllatottsagSulya object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgEgyebIntezmenyiEllatottsagSulyaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgEgyebIntezmenyiEllatottsagSulya object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgEgyebIntezmenyiEllatottsagSulyaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgEgyebIntezmenyiEllatottsagSulya object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgyebIntezmenyiEllatottsagSulyaQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgyebIntezmenyiEllatottsagSulyaQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the bankfiok\_lete column

\*

\* Example usage:

\* <code>

\* $query->filterByBankfiokLete(true); // WHERE bankfiok\_lete = true

\* $query->filterByBankfiokLete('yes'); // WHERE bankfiok\_lete = true

\* </code>

\*

\* @param boolean|string $bankfiokLete The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgyebIntezmenyiEllatottsagSulyaQuery The current query, for fluid interface

\*/

public function filterByBankfiokLete($bankfiokLete = null, $comparison = null)

{

if (is\_string($bankfiokLete)) {

$bankfiokLete = in\_array(strtolower($bankfiokLete), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BANKFIOK\_LETE, $bankfiokLete, $comparison);

}

/\*\*

\* Filter the query on the benzinkut\_lete column

\*

\* Example usage:

\* <code>

\* $query->filterByBenzinkutLete(true); // WHERE benzinkut\_lete = true

\* $query->filterByBenzinkutLete('yes'); // WHERE benzinkut\_lete = true

\* </code>

\*

\* @param boolean|string $benzinkutLete The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgyebIntezmenyiEllatottsagSulyaQuery The current query, for fluid interface

\*/

public function filterByBenzinkutLete($benzinkutLete = null, $comparison = null)

{

if (is\_string($benzinkutLete)) {

$benzinkutLete = in\_array(strtolower($benzinkutLete), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BENZINKUT\_LETE, $benzinkutLete, $comparison);

}

/\*\*

\* Filter the query on the korjegyzoseg\_szekhelye column

\*

\* Example usage:

\* <code>

\* $query->filterByKorjegyzosegSzekhelye(true); // WHERE korjegyzoseg\_szekhelye = true

\* $query->filterByKorjegyzosegSzekhelye('yes'); // WHERE korjegyzoseg\_szekhelye = true

\* </code>

\*

\* @param boolean|string $korjegyzosegSzekhelye The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgyebIntezmenyiEllatottsagSulyaQuery The current query, for fluid interface

\*/

public function filterByKorjegyzosegSzekhelye($korjegyzosegSzekhelye = null, $comparison = null)

{

if (is\_string($korjegyzosegSzekhelye)) {

$korjegyzosegSzekhelye = in\_array(strtolower($korjegyzosegSzekhelye), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_KORJEGYZOSEG\_SZEKHELYE, $korjegyzosegSzekhelye, $comparison);

}

/\*\*

\* Filter the query on the postahivatal\_lete column

\*

\* Example usage:

\* <code>

\* $query->filterByPostahivatalLete(true); // WHERE postahivatal\_lete = true

\* $query->filterByPostahivatalLete('yes'); // WHERE postahivatal\_lete = true

\* </code>

\*

\* @param boolean|string $postahivatalLete The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgyebIntezmenyiEllatottsagSulyaQuery The current query, for fluid interface

\*/

public function filterByPostahivatalLete($postahivatalLete = null, $comparison = null)

{

if (is\_string($postahivatalLete)) {

$postahivatalLete = in\_array(strtolower($postahivatalLete), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_POSTAHIVATAL\_LETE, $postahivatalLete, $comparison);

}

/\*\*

\* Filter the query on the birosag\_ugyeszseg\_lete column

\*

\* Example usage:

\* <code>

\* $query->filterByBirosagUgyeszsegLete(true); // WHERE birosag\_ugyeszseg\_lete = true

\* $query->filterByBirosagUgyeszsegLete('yes'); // WHERE birosag\_ugyeszseg\_lete = true

\* </code>

\*

\* @param boolean|string $birosagUgyeszsegLete The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgyebIntezmenyiEllatottsagSulyaQuery The current query, for fluid interface

\*/

public function filterByBirosagUgyeszsegLete($birosagUgyeszsegLete = null, $comparison = null)

{

if (is\_string($birosagUgyeszsegLete)) {

$birosagUgyeszsegLete = in\_array(strtolower($birosagUgyeszsegLete), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BIROSAG\_UGYESZSEG\_LETE, $birosagUgyeszsegLete, $comparison);

}

/\*\*

\* Filter the query on the okmanyiroda\_lete column

\*

\* Example usage:

\* <code>

\* $query->filterByOkmanyirodaLete(true); // WHERE okmanyiroda\_lete = true

\* $query->filterByOkmanyirodaLete('yes'); // WHERE okmanyiroda\_lete = true

\* </code>

\*

\* @param boolean|string $okmanyirodaLete The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgyebIntezmenyiEllatottsagSulyaQuery The current query, for fluid interface

\*/

public function filterByOkmanyirodaLete($okmanyirodaLete = null, $comparison = null)

{

if (is\_string($okmanyirodaLete)) {

$okmanyirodaLete = in\_array(strtolower($okmanyirodaLete), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_OKMANYIRODA\_LETE, $okmanyirodaLete, $comparison);

}

/\*\*

\* Filter the query on the megyei\_foldhivatal\_illetve\_kirendeltseg\_lete column

\*

\* Example usage:

\* <code>

\* $query->filterByMegyeiFoldhivatalIlletveKirendeltsegLete(true); // WHERE megyei\_foldhivatal\_illetve\_kirendeltseg\_lete = true

\* $query->filterByMegyeiFoldhivatalIlletveKirendeltsegLete('yes'); // WHERE megyei\_foldhivatal\_illetve\_kirendeltseg\_lete = true

\* </code>

\*

\* @param boolean|string $megyeiFoldhivatalIlletveKirendeltsegLete The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgyebIntezmenyiEllatottsagSulyaQuery The current query, for fluid interface

\*/

public function filterByMegyeiFoldhivatalIlletveKirendeltsegLete($megyeiFoldhivatalIlletveKirendeltsegLete = null, $comparison = null)

{

if (is\_string($megyeiFoldhivatalIlletveKirendeltsegLete)) {

$megyeiFoldhivatalIlletveKirendeltsegLete = in\_array(strtolower($megyeiFoldhivatalIlletveKirendeltsegLete), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_MEGYEI\_FOLDHIVATAL\_ILLETVE\_KIRENDELTSEG\_LETE, $megyeiFoldhivatalIlletveKirendeltsegLete, $comparison);

}

/\*\*

\* Filter the query on the munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete column

\*

\* Example usage:

\* <code>

\* $query->filterByMunkaugyiKozpontIlletveKirendeltsegLete(true); // WHERE munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete = true

\* $query->filterByMunkaugyiKozpontIlletveKirendeltsegLete('yes'); // WHERE munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete = true

\* </code>

\*

\* @param boolean|string $munkaugyiKozpontIlletveKirendeltsegLete The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgyebIntezmenyiEllatottsagSulyaQuery The current query, for fluid interface

\*/

public function filterByMunkaugyiKozpontIlletveKirendeltsegLete($munkaugyiKozpontIlletveKirendeltsegLete = null, $comparison = null)

{

if (is\_string($munkaugyiKozpontIlletveKirendeltsegLete)) {

$munkaugyiKozpontIlletveKirendeltsegLete = in\_array(strtolower($munkaugyiKozpontIlletveKirendeltsegLete), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_MUNKAUGYI\_KOZPONT\_ILLETVE\_KIRENDELTSEG\_LETE, $munkaugyiKozpontIlletveKirendeltsegLete, $comparison);

}

/\*\*

\* Filter the query on the vodafone\_uzlet\_lete column

\*

\* Example usage:

\* <code>

\* $query->filterByVodafoneUzletLete(true); // WHERE vodafone\_uzlet\_lete = true

\* $query->filterByVodafoneUzletLete('yes'); // WHERE vodafone\_uzlet\_lete = true

\* </code>

\*

\* @param boolean|string $vodafoneUzletLete The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgyebIntezmenyiEllatottsagSulyaQuery The current query, for fluid interface

\*/

public function filterByVodafoneUzletLete($vodafoneUzletLete = null, $comparison = null)

{

if (is\_string($vodafoneUzletLete)) {

$vodafoneUzletLete = in\_array(strtolower($vodafoneUzletLete), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_VODAFONE\_UZLET\_LETE, $vodafoneUzletLete, $comparison);

}

/\*\*

\* Filter the query on the t\_mobile\_uzlet\_lete column

\*

\* Example usage:

\* <code>

\* $query->filterByTMobileUzletLete(true); // WHERE t\_mobile\_uzlet\_lete = true

\* $query->filterByTMobileUzletLete('yes'); // WHERE t\_mobile\_uzlet\_lete = true

\* </code>

\*

\* @param boolean|string $tMobileUzletLete The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgyebIntezmenyiEllatottsagSulyaQuery The current query, for fluid interface

\*/

public function filterByTMobileUzletLete($tMobileUzletLete = null, $comparison = null)

{

if (is\_string($tMobileUzletLete)) {

$tMobileUzletLete = in\_array(strtolower($tMobileUzletLete), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_T\_MOBILE\_UZLET\_LETE, $tMobileUzletLete, $comparison);

}

/\*\*

\* Filter the query on the telenor\_uzlet\_lete column

\*

\* Example usage:

\* <code>

\* $query->filterByTelenorUzletLete(true); // WHERE telenor\_uzlet\_lete = true

\* $query->filterByTelenorUzletLete('yes'); // WHERE telenor\_uzlet\_lete = true

\* </code>

\*

\* @param boolean|string $telenorUzletLete The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgEgyebIntezmenyiEllatottsagSulyaQuery The current query, for fluid interface

\*/

public function filterByTelenorUzletLete($telenorUzletLete = null, $comparison = null)

{

if (is\_string($telenorUzletLete)) {

$telenorUzletLete = in\_array(strtolower($telenorUzletLete), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELENOR\_UZLET\_LETE, $telenorUzletLete, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgEgyebIntezmenyiEllatottsagSulya $stgEgyebIntezmenyiEllatottsagSulya Object to remove from the list of results

\*

\* @return $this|ChildStgEgyebIntezmenyiEllatottsagSulyaQuery The current query, for fluid interface

\*/

public function prune($stgEgyebIntezmenyiEllatottsagSulya = null)

{

if ($stgEgyebIntezmenyiEllatottsagSulya) {

throw new LogicException('StgEgyebIntezmenyiEllatottsagSulya object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_egyeb\_intezmenyi\_ellatottsag\_sulya table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgEgyebIntezmenyiEllatottsagSulyaTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgEgyebIntezmenyiEllatottsagSulyaTableMap::clearInstancePool();

StgEgyebIntezmenyiEllatottsagSulyaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgEgyebIntezmenyiEllatottsagSulyaTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgEgyebIntezmenyiEllatottsagSulyaTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgEgyebIntezmenyiEllatottsagSulyaTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgEgyebIntezmenyiEllatottsagSulyaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgEgyebIntezmenyiEllatottsagSulyaQuery

#### StgElelmiszeruzletekEsAruhazak.php

<?php

namespace Base;

use \StgElelmiszeruzletekEsAruhazakQuery as ChildStgElelmiszeruzletekEsAruhazakQuery;

use \Exception;

use \PDO;

use Map\StgElelmiszeruzletekEsAruhazakTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgElelmiszeruzletekEsAruhazak implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgElelmiszeruzletekEsAruhazakTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the kiskereskedelmi\_uzletek\_szama field.

\* @var int

\*/

protected $kiskereskedelmi\_uzletek\_szama;

/\*\*

\* The value for the elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama field.

\* @var int

\*/

protected $elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgElelmiszeruzletekEsAruhazak object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgElelmiszeruzletekEsAruhazak</code> instance. If

\* <code>obj</code> is an instance of <code>StgElelmiszeruzletekEsAruhazak</code>, delegates to

\* <code>equals(StgElelmiszeruzletekEsAruhazak)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgElelmiszeruzletekEsAruhazak The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [kiskereskedelmi\_uzletek\_szama] column value.

\*

\* @return int

\*/

public function getKiskereskedelmiUzletekSzama()

{

return $this->kiskereskedelmi\_uzletek\_szama;

}

/\*\*

\* Get the [elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama] column value.

\*

\* @return int

\*/

public function getElelmiszerVegyesuzletekEsAruhazakSzama()

{

return $this->elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgElelmiszeruzletekEsAruhazakTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgElelmiszeruzletekEsAruhazakTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgElelmiszeruzletekEsAruhazakTableMap::translateFieldName('KiskereskedelmiUzletekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kiskereskedelmi\_uzletek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgElelmiszeruzletekEsAruhazakTableMap::translateFieldName('ElelmiszerVegyesuzletekEsAruhazakSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 4; // 4 = StgElelmiszeruzletekEsAruhazakTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgElelmiszeruzletekEsAruhazak'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgElelmiszeruzletekEsAruhazak The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgElelmiszeruzletekEsAruhazakTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgElelmiszeruzletekEsAruhazak The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgElelmiszeruzletekEsAruhazakTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [kiskereskedelmi\_uzletek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgElelmiszeruzletekEsAruhazak The current object (for fluent API support)

\*/

public function setKiskereskedelmiUzletekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->kiskereskedelmi\_uzletek\_szama !== $v) {

$this->kiskereskedelmi\_uzletek\_szama = $v;

$this->modifiedColumns[StgElelmiszeruzletekEsAruhazakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA] = true;

}

return $this;

} // setKiskereskedelmiUzletekSzama()

/\*\*

\* Set the value of [elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgElelmiszeruzletekEsAruhazak The current object (for fluent API support)

\*/

public function setElelmiszerVegyesuzletekEsAruhazakSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama !== $v) {

$this->elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama = $v;

$this->modifiedColumns[StgElelmiszeruzletekEsAruhazakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA] = true;

}

return $this;

} // setElelmiszerVegyesuzletekEsAruhazakSzama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgElelmiszeruzletekEsAruhazakTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgElelmiszeruzletekEsAruhazakQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgElelmiszeruzletekEsAruhazak::setDeleted()

\* @see StgElelmiszeruzletekEsAruhazak::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgElelmiszeruzletekEsAruhazakTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgElelmiszeruzletekEsAruhazakQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgElelmiszeruzletekEsAruhazakTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgElelmiszeruzletekEsAruhazakTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgElelmiszeruzletekEsAruhazakTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgElelmiszeruzletekEsAruhazakTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgElelmiszeruzletekEsAruhazakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'KISKERESKEDELMI\_UZLETEK\_SZAMA';

}

if ($this->isColumnModified(StgElelmiszeruzletekEsAruhazakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_elelmiszeruzletek\_es\_aruhazak (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'KISKERESKEDELMI\_UZLETEK\_SZAMA':

$stmt->bindValue($identifier, $this->kiskereskedelmi\_uzletek\_szama, PDO::PARAM\_INT);

break;

case 'ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA':

$stmt->bindValue($identifier, $this->elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgElelmiszeruzletekEsAruhazakTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getKiskereskedelmiUzletekSzama();

break;

case 3:

return $this->getElelmiszerVegyesuzletekEsAruhazakSzama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgElelmiszeruzletekEsAruhazak'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgElelmiszeruzletekEsAruhazak'][$this->getPrimaryKey()] = true;

$keys = StgElelmiszeruzletekEsAruhazakTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getKiskereskedelmiUzletekSzama(),

$keys[3] => $this->getElelmiszerVegyesuzletekEsAruhazakSzama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgElelmiszeruzletekEsAruhazak

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgElelmiszeruzletekEsAruhazakTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgElelmiszeruzletekEsAruhazak

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setKiskereskedelmiUzletekSzama($value);

break;

case 3:

$this->setElelmiszerVegyesuzletekEsAruhazakSzama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgElelmiszeruzletekEsAruhazakTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setKiskereskedelmiUzletekSzama($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setElelmiszerVegyesuzletekEsAruhazakSzama($arr[$keys[3]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgElelmiszeruzletekEsAruhazak The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgElelmiszeruzletekEsAruhazakTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgElelmiszeruzletekEsAruhazakTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgElelmiszeruzletekEsAruhazakTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgElelmiszeruzletekEsAruhazakTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgElelmiszeruzletekEsAruhazakTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgElelmiszeruzletekEsAruhazakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA)) {

$criteria->add(StgElelmiszeruzletekEsAruhazakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA, $this->kiskereskedelmi\_uzletek\_szama);

}

if ($this->isColumnModified(StgElelmiszeruzletekEsAruhazakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA)) {

$criteria->add(StgElelmiszeruzletekEsAruhazakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA, $this->elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgElelmiszeruzletekEsAruhazak object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgElelmiszeruzletekEsAruhazak (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setKiskereskedelmiUzletekSzama($this->getKiskereskedelmiUzletekSzama());

$copyObj->setElelmiszerVegyesuzletekEsAruhazakSzama($this->getElelmiszerVegyesuzletekEsAruhazakSzama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgElelmiszeruzletekEsAruhazak Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->kiskereskedelmi\_uzletek\_szama = null;

$this->elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgElelmiszeruzletekEsAruhazakTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgElelmiszeruzletekEsAruhazakQuery.php

<?php

namespace Base;

use \StgElelmiszeruzletekEsAruhazak as ChildStgElelmiszeruzletekEsAruhazak;

use \StgElelmiszeruzletekEsAruhazakQuery as ChildStgElelmiszeruzletekEsAruhazakQuery;

use \Exception;

use Map\StgElelmiszeruzletekEsAruhazakTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_elelmiszeruzletek\_es\_aruhazak' table.

\*

\*

\*

\* @method ChildStgElelmiszeruzletekEsAruhazakQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgElelmiszeruzletekEsAruhazakQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgElelmiszeruzletekEsAruhazakQuery orderByKiskereskedelmiUzletekSzama($order = Criteria::ASC) Order by the kiskereskedelmi\_uzletek\_szama column

\* @method ChildStgElelmiszeruzletekEsAruhazakQuery orderByElelmiszerVegyesuzletekEsAruhazakSzama($order = Criteria::ASC) Order by the elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama column

\*

\* @method ChildStgElelmiszeruzletekEsAruhazakQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgElelmiszeruzletekEsAruhazakQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgElelmiszeruzletekEsAruhazakQuery groupByKiskereskedelmiUzletekSzama() Group by the kiskereskedelmi\_uzletek\_szama column

\* @method ChildStgElelmiszeruzletekEsAruhazakQuery groupByElelmiszerVegyesuzletekEsAruhazakSzama() Group by the elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama column

\*

\* @method ChildStgElelmiszeruzletekEsAruhazakQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgElelmiszeruzletekEsAruhazakQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgElelmiszeruzletekEsAruhazakQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgElelmiszeruzletekEsAruhazak findOne(ConnectionInterface $con = null) Return the first ChildStgElelmiszeruzletekEsAruhazak matching the query

\* @method ChildStgElelmiszeruzletekEsAruhazak findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgElelmiszeruzletekEsAruhazak matching the query, or a new ChildStgElelmiszeruzletekEsAruhazak object populated from the query conditions when no match is found

\*

\* @method ChildStgElelmiszeruzletekEsAruhazak findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgElelmiszeruzletekEsAruhazak filtered by the telepules\_nev column

\* @method ChildStgElelmiszeruzletekEsAruhazak findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgElelmiszeruzletekEsAruhazak filtered by the telepules\_KSHKOD column

\* @method ChildStgElelmiszeruzletekEsAruhazak findOneByKiskereskedelmiUzletekSzama(int $kiskereskedelmi\_uzletek\_szama) Return the first ChildStgElelmiszeruzletekEsAruhazak filtered by the kiskereskedelmi\_uzletek\_szama column

\* @method ChildStgElelmiszeruzletekEsAruhazak findOneByElelmiszerVegyesuzletekEsAruhazakSzama(int $elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama) Return the first ChildStgElelmiszeruzletekEsAruhazak filtered by the elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama column

\*

\* @method ChildStgElelmiszeruzletekEsAruhazak[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgElelmiszeruzletekEsAruhazak objects based on current ModelCriteria

\* @method ChildStgElelmiszeruzletekEsAruhazak[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgElelmiszeruzletekEsAruhazak objects filtered by the telepules\_nev column

\* @method ChildStgElelmiszeruzletekEsAruhazak[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgElelmiszeruzletekEsAruhazak objects filtered by the telepules\_KSHKOD column

\* @method ChildStgElelmiszeruzletekEsAruhazak[]|ObjectCollection findByKiskereskedelmiUzletekSzama(int $kiskereskedelmi\_uzletek\_szama) Return ChildStgElelmiszeruzletekEsAruhazak objects filtered by the kiskereskedelmi\_uzletek\_szama column

\* @method ChildStgElelmiszeruzletekEsAruhazak[]|ObjectCollection findByElelmiszerVegyesuzletekEsAruhazakSzama(int $elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama) Return ChildStgElelmiszeruzletekEsAruhazak objects filtered by the elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama column

\* @method ChildStgElelmiszeruzletekEsAruhazak[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgElelmiszeruzletekEsAruhazakQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgElelmiszeruzletekEsAruhazakQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgElelmiszeruzletekEsAruhazak', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgElelmiszeruzletekEsAruhazakQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgElelmiszeruzletekEsAruhazakQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgElelmiszeruzletekEsAruhazakQuery) {

return $criteria;

}

$query = new ChildStgElelmiszeruzletekEsAruhazakQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgElelmiszeruzletekEsAruhazak|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgElelmiszeruzletekEsAruhazak object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgElelmiszeruzletekEsAruhazak object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgElelmiszeruzletekEsAruhazakQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgElelmiszeruzletekEsAruhazak object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgElelmiszeruzletekEsAruhazakQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgElelmiszeruzletekEsAruhazak object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgElelmiszeruzletekEsAruhazakQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgElelmiszeruzletekEsAruhazakTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgElelmiszeruzletekEsAruhazakQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgElelmiszeruzletekEsAruhazakTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the kiskereskedelmi\_uzletek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByKiskereskedelmiUzletekSzama(1234); // WHERE kiskereskedelmi\_uzletek\_szama = 1234

\* $query->filterByKiskereskedelmiUzletekSzama(array(12, 34)); // WHERE kiskereskedelmi\_uzletek\_szama IN (12, 34)

\* $query->filterByKiskereskedelmiUzletekSzama(array('min' => 12)); // WHERE kiskereskedelmi\_uzletek\_szama > 12

\* </code>

\*

\* @param mixed $kiskereskedelmiUzletekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgElelmiszeruzletekEsAruhazakQuery The current query, for fluid interface

\*/

public function filterByKiskereskedelmiUzletekSzama($kiskereskedelmiUzletekSzama = null, $comparison = null)

{

if (is\_array($kiskereskedelmiUzletekSzama)) {

$useMinMax = false;

if (isset($kiskereskedelmiUzletekSzama['min'])) {

$this->addUsingAlias(StgElelmiszeruzletekEsAruhazakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA, $kiskereskedelmiUzletekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($kiskereskedelmiUzletekSzama['max'])) {

$this->addUsingAlias(StgElelmiszeruzletekEsAruhazakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA, $kiskereskedelmiUzletekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgElelmiszeruzletekEsAruhazakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA, $kiskereskedelmiUzletekSzama, $comparison);

}

/\*\*

\* Filter the query on the elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByElelmiszerVegyesuzletekEsAruhazakSzama(1234); // WHERE elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama = 1234

\* $query->filterByElelmiszerVegyesuzletekEsAruhazakSzama(array(12, 34)); // WHERE elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama IN (12, 34)

\* $query->filterByElelmiszerVegyesuzletekEsAruhazakSzama(array('min' => 12)); // WHERE elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama > 12

\* </code>

\*

\* @param mixed $elelmiszerVegyesuzletekEsAruhazakSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgElelmiszeruzletekEsAruhazakQuery The current query, for fluid interface

\*/

public function filterByElelmiszerVegyesuzletekEsAruhazakSzama($elelmiszerVegyesuzletekEsAruhazakSzama = null, $comparison = null)

{

if (is\_array($elelmiszerVegyesuzletekEsAruhazakSzama)) {

$useMinMax = false;

if (isset($elelmiszerVegyesuzletekEsAruhazakSzama['min'])) {

$this->addUsingAlias(StgElelmiszeruzletekEsAruhazakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA, $elelmiszerVegyesuzletekEsAruhazakSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($elelmiszerVegyesuzletekEsAruhazakSzama['max'])) {

$this->addUsingAlias(StgElelmiszeruzletekEsAruhazakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA, $elelmiszerVegyesuzletekEsAruhazakSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgElelmiszeruzletekEsAruhazakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA, $elelmiszerVegyesuzletekEsAruhazakSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgElelmiszeruzletekEsAruhazak $stgElelmiszeruzletekEsAruhazak Object to remove from the list of results

\*

\* @return $this|ChildStgElelmiszeruzletekEsAruhazakQuery The current query, for fluid interface

\*/

public function prune($stgElelmiszeruzletekEsAruhazak = null)

{

if ($stgElelmiszeruzletekEsAruhazak) {

throw new LogicException('StgElelmiszeruzletekEsAruhazak object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_elelmiszeruzletek\_es\_aruhazak table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgElelmiszeruzletekEsAruhazakTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgElelmiszeruzletekEsAruhazakTableMap::clearInstancePool();

StgElelmiszeruzletekEsAruhazakTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgElelmiszeruzletekEsAruhazakTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgElelmiszeruzletekEsAruhazakTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgElelmiszeruzletekEsAruhazakTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgElelmiszeruzletekEsAruhazakTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgElelmiszeruzletekEsAruhazakQuery

#### StgErintettNepessegSzama.php

<?php

namespace Base;

use \StgErintettNepessegSzamaQuery as ChildStgErintettNepessegSzamaQuery;

use \Exception;

use \PDO;

use Map\StgErintettNepessegSzamaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgErintettNepessegSzama implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgErintettNepessegSzamaTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the regisztralt\_bunelkovetok\_szama field.

\* @var int

\*/

protected $regisztralt\_bunelkovetok\_szama;

/\*\*

\* The value for the haztartasi\_gazfogyasztok\_szama field.

\* @var int

\*/

protected $haztartasi\_gazfogyasztok\_szama;

/\*\*

\* The value for the haztartasi\_villamosenergia\_fogyasztok\_szama field.

\* @var int

\*/

protected $haztartasi\_villamosenergia\_fogyasztok\_szama;

/\*\*

\* The value for the elvandorlasok\_szama field.

\* @var int

\*/

protected $elvandorlasok\_szama;

/\*\*

\* The value for the allando\_nepesseg\_szama field.

\* @var int

\*/

protected $allando\_nepesseg\_szama;

/\*\*

\* The value for the vallalkozasok\_szama field.

\* @var int

\*/

protected $vallalkozasok\_szama;

/\*\*

\* The value for the nyilvantartott\_allaskeresok\_szama field.

\* @var int

\*/

protected $nyilvantartott\_allaskeresok\_szama;

/\*\*

\* The value for the lakasallomany field.

\* @var int

\*/

protected $lakasallomany;

/\*\*

\* The value for the bolcsodebe\_beirt\_gyermekek\_szama field.

\* @var int

\*/

protected $bolcsodebe\_beirt\_gyermekek\_szama;

/\*\*

\* The value for the allando\_nepessegbol\_18\_59\_evesek\_szama field.

\* @var int

\*/

protected $allando\_nepessegbol\_18\_59\_evesek\_szama;

/\*\*

\* The value for the mobilelofizetesek\_szama field.

\* @var int

\*/

protected $mobilelofizetesek\_szama;

/\*\*

\* The value for the osszes\_adofizeto field.

\* @var int

\*/

protected $osszes\_adofizeto;

/\*\*

\* The value for the osszes\_ado field.

\* @var int

\*/

protected $osszes\_ado;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgErintettNepessegSzama object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgErintettNepessegSzama</code> instance. If

\* <code>obj</code> is an instance of <code>StgErintettNepessegSzama</code>, delegates to

\* <code>equals(StgErintettNepessegSzama)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgErintettNepessegSzama The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [regisztralt\_bunelkovetok\_szama] column value.

\*

\* @return int

\*/

public function getRegisztraltBunelkovetokSzama()

{

return $this->regisztralt\_bunelkovetok\_szama;

}

/\*\*

\* Get the [haztartasi\_gazfogyasztok\_szama] column value.

\*

\* @return int

\*/

public function getHaztartasiGazfogyasztokSzama()

{

return $this->haztartasi\_gazfogyasztok\_szama;

}

/\*\*

\* Get the [haztartasi\_villamosenergia\_fogyasztok\_szama] column value.

\*

\* @return int

\*/

public function getHaztartasiVillamosenergiaFogyasztokSzama()

{

return $this->haztartasi\_villamosenergia\_fogyasztok\_szama;

}

/\*\*

\* Get the [elvandorlasok\_szama] column value.

\*

\* @return int

\*/

public function getElvandorlasokSzama()

{

return $this->elvandorlasok\_szama;

}

/\*\*

\* Get the [allando\_nepesseg\_szama] column value.

\*

\* @return int

\*/

public function getAllandoNepessegSzama()

{

return $this->allando\_nepesseg\_szama;

}

/\*\*

\* Get the [vallalkozasok\_szama] column value.

\*

\* @return int

\*/

public function getVallalkozasokSzama()

{

return $this->vallalkozasok\_szama;

}

/\*\*

\* Get the [nyilvantartott\_allaskeresok\_szama] column value.

\*

\* @return int

\*/

public function getNyilvantartottAllaskeresokSzama()

{

return $this->nyilvantartott\_allaskeresok\_szama;

}

/\*\*

\* Get the [lakasallomany] column value.

\*

\* @return int

\*/

public function getLakasallomany()

{

return $this->lakasallomany;

}

/\*\*

\* Get the [bolcsodebe\_beirt\_gyermekek\_szama] column value.

\*

\* @return int

\*/

public function getBolcsodebeBeirtGyermekekSzama()

{

return $this->bolcsodebe\_beirt\_gyermekek\_szama;

}

/\*\*

\* Get the [allando\_nepessegbol\_18\_59\_evesek\_szama] column value.

\*

\* @return int

\*/

public function getAllandoNepessegbol1859EvesekSzama()

{

return $this->allando\_nepessegbol\_18\_59\_evesek\_szama;

}

/\*\*

\* Get the [mobilelofizetesek\_szama] column value.

\*

\* @return int

\*/

public function getMobilelofizetesekSzama()

{

return $this->mobilelofizetesek\_szama;

}

/\*\*

\* Get the [osszes\_adofizeto] column value.

\*

\* @return int

\*/

public function getOsszesAdofizeto()

{

return $this->osszes\_adofizeto;

}

/\*\*

\* Get the [osszes\_ado] column value.

\*

\* @return int

\*/

public function getOsszesAdo()

{

return $this->osszes\_ado;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgErintettNepessegSzamaTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgErintettNepessegSzamaTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgErintettNepessegSzamaTableMap::translateFieldName('RegisztraltBunelkovetokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->regisztralt\_bunelkovetok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgErintettNepessegSzamaTableMap::translateFieldName('HaztartasiGazfogyasztokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->haztartasi\_gazfogyasztok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgErintettNepessegSzamaTableMap::translateFieldName('HaztartasiVillamosenergiaFogyasztokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->haztartasi\_villamosenergia\_fogyasztok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 5 + $startcol : StgErintettNepessegSzamaTableMap::translateFieldName('ElvandorlasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->elvandorlasok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 6 + $startcol : StgErintettNepessegSzamaTableMap::translateFieldName('AllandoNepessegSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->allando\_nepesseg\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 7 + $startcol : StgErintettNepessegSzamaTableMap::translateFieldName('VallalkozasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->vallalkozasok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 8 + $startcol : StgErintettNepessegSzamaTableMap::translateFieldName('NyilvantartottAllaskeresokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nyilvantartott\_allaskeresok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 9 + $startcol : StgErintettNepessegSzamaTableMap::translateFieldName('Lakasallomany', TableMap::TYPE\_PHPNAME, $indexType)];

$this->lakasallomany = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 10 + $startcol : StgErintettNepessegSzamaTableMap::translateFieldName('BolcsodebeBeirtGyermekekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->bolcsodebe\_beirt\_gyermekek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 11 + $startcol : StgErintettNepessegSzamaTableMap::translateFieldName('AllandoNepessegbol1859EvesekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->allando\_nepessegbol\_18\_59\_evesek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 12 + $startcol : StgErintettNepessegSzamaTableMap::translateFieldName('MobilelofizetesekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->mobilelofizetesek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 13 + $startcol : StgErintettNepessegSzamaTableMap::translateFieldName('OsszesAdofizeto', TableMap::TYPE\_PHPNAME, $indexType)];

$this->osszes\_adofizeto = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 14 + $startcol : StgErintettNepessegSzamaTableMap::translateFieldName('OsszesAdo', TableMap::TYPE\_PHPNAME, $indexType)];

$this->osszes\_ado = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 15; // 15 = StgErintettNepessegSzamaTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgErintettNepessegSzama'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgErintettNepessegSzama The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgErintettNepessegSzamaTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgErintettNepessegSzama The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgErintettNepessegSzamaTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [regisztralt\_bunelkovetok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgErintettNepessegSzama The current object (for fluent API support)

\*/

public function setRegisztraltBunelkovetokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->regisztralt\_bunelkovetok\_szama !== $v) {

$this->regisztralt\_bunelkovetok\_szama = $v;

$this->modifiedColumns[StgErintettNepessegSzamaTableMap::COL\_REGISZTRALT\_BUNELKOVETOK\_SZAMA] = true;

}

return $this;

} // setRegisztraltBunelkovetokSzama()

/\*\*

\* Set the value of [haztartasi\_gazfogyasztok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgErintettNepessegSzama The current object (for fluent API support)

\*/

public function setHaztartasiGazfogyasztokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->haztartasi\_gazfogyasztok\_szama !== $v) {

$this->haztartasi\_gazfogyasztok\_szama = $v;

$this->modifiedColumns[StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_GAZFOGYASZTOK\_SZAMA] = true;

}

return $this;

} // setHaztartasiGazfogyasztokSzama()

/\*\*

\* Set the value of [haztartasi\_villamosenergia\_fogyasztok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgErintettNepessegSzama The current object (for fluent API support)

\*/

public function setHaztartasiVillamosenergiaFogyasztokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->haztartasi\_villamosenergia\_fogyasztok\_szama !== $v) {

$this->haztartasi\_villamosenergia\_fogyasztok\_szama = $v;

$this->modifiedColumns[StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA] = true;

}

return $this;

} // setHaztartasiVillamosenergiaFogyasztokSzama()

/\*\*

\* Set the value of [elvandorlasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgErintettNepessegSzama The current object (for fluent API support)

\*/

public function setElvandorlasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->elvandorlasok\_szama !== $v) {

$this->elvandorlasok\_szama = $v;

$this->modifiedColumns[StgErintettNepessegSzamaTableMap::COL\_ELVANDORLASOK\_SZAMA] = true;

}

return $this;

} // setElvandorlasokSzama()

/\*\*

\* Set the value of [allando\_nepesseg\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgErintettNepessegSzama The current object (for fluent API support)

\*/

public function setAllandoNepessegSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->allando\_nepesseg\_szama !== $v) {

$this->allando\_nepesseg\_szama = $v;

$this->modifiedColumns[StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA] = true;

}

return $this;

} // setAllandoNepessegSzama()

/\*\*

\* Set the value of [vallalkozasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgErintettNepessegSzama The current object (for fluent API support)

\*/

public function setVallalkozasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->vallalkozasok\_szama !== $v) {

$this->vallalkozasok\_szama = $v;

$this->modifiedColumns[StgErintettNepessegSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA] = true;

}

return $this;

} // setVallalkozasokSzama()

/\*\*

\* Set the value of [nyilvantartott\_allaskeresok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgErintettNepessegSzama The current object (for fluent API support)

\*/

public function setNyilvantartottAllaskeresokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nyilvantartott\_allaskeresok\_szama !== $v) {

$this->nyilvantartott\_allaskeresok\_szama = $v;

$this->modifiedColumns[StgErintettNepessegSzamaTableMap::COL\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA] = true;

}

return $this;

} // setNyilvantartottAllaskeresokSzama()

/\*\*

\* Set the value of [lakasallomany] column.

\*

\* @param int $v new value

\* @return $this|\StgErintettNepessegSzama The current object (for fluent API support)

\*/

public function setLakasallomany($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->lakasallomany !== $v) {

$this->lakasallomany = $v;

$this->modifiedColumns[StgErintettNepessegSzamaTableMap::COL\_LAKASALLOMANY] = true;

}

return $this;

} // setLakasallomany()

/\*\*

\* Set the value of [bolcsodebe\_beirt\_gyermekek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgErintettNepessegSzama The current object (for fluent API support)

\*/

public function setBolcsodebeBeirtGyermekekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->bolcsodebe\_beirt\_gyermekek\_szama !== $v) {

$this->bolcsodebe\_beirt\_gyermekek\_szama = $v;

$this->modifiedColumns[StgErintettNepessegSzamaTableMap::COL\_BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA] = true;

}

return $this;

} // setBolcsodebeBeirtGyermekekSzama()

/\*\*

\* Set the value of [allando\_nepessegbol\_18\_59\_evesek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgErintettNepessegSzama The current object (for fluent API support)

\*/

public function setAllandoNepessegbol1859EvesekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->allando\_nepessegbol\_18\_59\_evesek\_szama !== $v) {

$this->allando\_nepessegbol\_18\_59\_evesek\_szama = $v;

$this->modifiedColumns[StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA] = true;

}

return $this;

} // setAllandoNepessegbol1859EvesekSzama()

/\*\*

\* Set the value of [mobilelofizetesek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgErintettNepessegSzama The current object (for fluent API support)

\*/

public function setMobilelofizetesekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->mobilelofizetesek\_szama !== $v) {

$this->mobilelofizetesek\_szama = $v;

$this->modifiedColumns[StgErintettNepessegSzamaTableMap::COL\_MOBILELOFIZETESEK\_SZAMA] = true;

}

return $this;

} // setMobilelofizetesekSzama()

/\*\*

\* Set the value of [osszes\_adofizeto] column.

\*

\* @param int $v new value

\* @return $this|\StgErintettNepessegSzama The current object (for fluent API support)

\*/

public function setOsszesAdofizeto($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->osszes\_adofizeto !== $v) {

$this->osszes\_adofizeto = $v;

$this->modifiedColumns[StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADOFIZETO] = true;

}

return $this;

} // setOsszesAdofizeto()

/\*\*

\* Set the value of [osszes\_ado] column.

\*

\* @param int $v new value

\* @return $this|\StgErintettNepessegSzama The current object (for fluent API support)

\*/

public function setOsszesAdo($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->osszes\_ado !== $v) {

$this->osszes\_ado = $v;

$this->modifiedColumns[StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADO] = true;

}

return $this;

} // setOsszesAdo()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgErintettNepessegSzamaTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgErintettNepessegSzamaQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgErintettNepessegSzama::setDeleted()

\* @see StgErintettNepessegSzama::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgErintettNepessegSzamaTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgErintettNepessegSzamaQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgErintettNepessegSzamaTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgErintettNepessegSzamaTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_REGISZTRALT\_BUNELKOVETOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'REGISZTRALT\_BUNELKOVETOK\_SZAMA';

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_GAZFOGYASZTOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'HAZTARTASI\_GAZFOGYASZTOK\_SZAMA';

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA';

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_ELVANDORLASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'ELVANDORLASOK\_SZAMA';

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'ALLANDO\_NEPESSEG\_SZAMA';

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'VALLALKOZASOK\_SZAMA';

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'NYILVANTARTOTT\_ALLASKERESOK\_SZAMA';

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_LAKASALLOMANY)) {

$modifiedColumns[':p' . $index++] = 'LAKASALLOMANY';

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA';

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA';

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_MOBILELOFIZETESEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'MOBILELOFIZETESEK\_SZAMA';

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADOFIZETO)) {

$modifiedColumns[':p' . $index++] = 'OSSZES\_ADOFIZETO';

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADO)) {

$modifiedColumns[':p' . $index++] = 'OSSZES\_ADO';

}

$sql = sprintf(

'INSERT INTO stg\_erintett\_nepesseg\_szama (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'REGISZTRALT\_BUNELKOVETOK\_SZAMA':

$stmt->bindValue($identifier, $this->regisztralt\_bunelkovetok\_szama, PDO::PARAM\_INT);

break;

case 'HAZTARTASI\_GAZFOGYASZTOK\_SZAMA':

$stmt->bindValue($identifier, $this->haztartasi\_gazfogyasztok\_szama, PDO::PARAM\_INT);

break;

case 'HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA':

$stmt->bindValue($identifier, $this->haztartasi\_villamosenergia\_fogyasztok\_szama, PDO::PARAM\_INT);

break;

case 'ELVANDORLASOK\_SZAMA':

$stmt->bindValue($identifier, $this->elvandorlasok\_szama, PDO::PARAM\_INT);

break;

case 'ALLANDO\_NEPESSEG\_SZAMA':

$stmt->bindValue($identifier, $this->allando\_nepesseg\_szama, PDO::PARAM\_INT);

break;

case 'VALLALKOZASOK\_SZAMA':

$stmt->bindValue($identifier, $this->vallalkozasok\_szama, PDO::PARAM\_INT);

break;

case 'NYILVANTARTOTT\_ALLASKERESOK\_SZAMA':

$stmt->bindValue($identifier, $this->nyilvantartott\_allaskeresok\_szama, PDO::PARAM\_INT);

break;

case 'LAKASALLOMANY':

$stmt->bindValue($identifier, $this->lakasallomany, PDO::PARAM\_INT);

break;

case 'BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA':

$stmt->bindValue($identifier, $this->bolcsodebe\_beirt\_gyermekek\_szama, PDO::PARAM\_INT);

break;

case 'ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA':

$stmt->bindValue($identifier, $this->allando\_nepessegbol\_18\_59\_evesek\_szama, PDO::PARAM\_INT);

break;

case 'MOBILELOFIZETESEK\_SZAMA':

$stmt->bindValue($identifier, $this->mobilelofizetesek\_szama, PDO::PARAM\_INT);

break;

case 'OSSZES\_ADOFIZETO':

$stmt->bindValue($identifier, $this->osszes\_adofizeto, PDO::PARAM\_INT);

break;

case 'OSSZES\_ADO':

$stmt->bindValue($identifier, $this->osszes\_ado, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgErintettNepessegSzamaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getRegisztraltBunelkovetokSzama();

break;

case 3:

return $this->getHaztartasiGazfogyasztokSzama();

break;

case 4:

return $this->getHaztartasiVillamosenergiaFogyasztokSzama();

break;

case 5:

return $this->getElvandorlasokSzama();

break;

case 6:

return $this->getAllandoNepessegSzama();

break;

case 7:

return $this->getVallalkozasokSzama();

break;

case 8:

return $this->getNyilvantartottAllaskeresokSzama();

break;

case 9:

return $this->getLakasallomany();

break;

case 10:

return $this->getBolcsodebeBeirtGyermekekSzama();

break;

case 11:

return $this->getAllandoNepessegbol1859EvesekSzama();

break;

case 12:

return $this->getMobilelofizetesekSzama();

break;

case 13:

return $this->getOsszesAdofizeto();

break;

case 14:

return $this->getOsszesAdo();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgErintettNepessegSzama'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgErintettNepessegSzama'][$this->getPrimaryKey()] = true;

$keys = StgErintettNepessegSzamaTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getRegisztraltBunelkovetokSzama(),

$keys[3] => $this->getHaztartasiGazfogyasztokSzama(),

$keys[4] => $this->getHaztartasiVillamosenergiaFogyasztokSzama(),

$keys[5] => $this->getElvandorlasokSzama(),

$keys[6] => $this->getAllandoNepessegSzama(),

$keys[7] => $this->getVallalkozasokSzama(),

$keys[8] => $this->getNyilvantartottAllaskeresokSzama(),

$keys[9] => $this->getLakasallomany(),

$keys[10] => $this->getBolcsodebeBeirtGyermekekSzama(),

$keys[11] => $this->getAllandoNepessegbol1859EvesekSzama(),

$keys[12] => $this->getMobilelofizetesekSzama(),

$keys[13] => $this->getOsszesAdofizeto(),

$keys[14] => $this->getOsszesAdo(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgErintettNepessegSzama

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgErintettNepessegSzamaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgErintettNepessegSzama

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setRegisztraltBunelkovetokSzama($value);

break;

case 3:

$this->setHaztartasiGazfogyasztokSzama($value);

break;

case 4:

$this->setHaztartasiVillamosenergiaFogyasztokSzama($value);

break;

case 5:

$this->setElvandorlasokSzama($value);

break;

case 6:

$this->setAllandoNepessegSzama($value);

break;

case 7:

$this->setVallalkozasokSzama($value);

break;

case 8:

$this->setNyilvantartottAllaskeresokSzama($value);

break;

case 9:

$this->setLakasallomany($value);

break;

case 10:

$this->setBolcsodebeBeirtGyermekekSzama($value);

break;

case 11:

$this->setAllandoNepessegbol1859EvesekSzama($value);

break;

case 12:

$this->setMobilelofizetesekSzama($value);

break;

case 13:

$this->setOsszesAdofizeto($value);

break;

case 14:

$this->setOsszesAdo($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgErintettNepessegSzamaTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setRegisztraltBunelkovetokSzama($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setHaztartasiGazfogyasztokSzama($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setHaztartasiVillamosenergiaFogyasztokSzama($arr[$keys[4]]);

}

if (array\_key\_exists($keys[5], $arr)) {

$this->setElvandorlasokSzama($arr[$keys[5]]);

}

if (array\_key\_exists($keys[6], $arr)) {

$this->setAllandoNepessegSzama($arr[$keys[6]]);

}

if (array\_key\_exists($keys[7], $arr)) {

$this->setVallalkozasokSzama($arr[$keys[7]]);

}

if (array\_key\_exists($keys[8], $arr)) {

$this->setNyilvantartottAllaskeresokSzama($arr[$keys[8]]);

}

if (array\_key\_exists($keys[9], $arr)) {

$this->setLakasallomany($arr[$keys[9]]);

}

if (array\_key\_exists($keys[10], $arr)) {

$this->setBolcsodebeBeirtGyermekekSzama($arr[$keys[10]]);

}

if (array\_key\_exists($keys[11], $arr)) {

$this->setAllandoNepessegbol1859EvesekSzama($arr[$keys[11]]);

}

if (array\_key\_exists($keys[12], $arr)) {

$this->setMobilelofizetesekSzama($arr[$keys[12]]);

}

if (array\_key\_exists($keys[13], $arr)) {

$this->setOsszesAdofizeto($arr[$keys[13]]);

}

if (array\_key\_exists($keys[14], $arr)) {

$this->setOsszesAdo($arr[$keys[14]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgErintettNepessegSzama The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgErintettNepessegSzamaTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgErintettNepessegSzamaTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgErintettNepessegSzamaTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_REGISZTRALT\_BUNELKOVETOK\_SZAMA)) {

$criteria->add(StgErintettNepessegSzamaTableMap::COL\_REGISZTRALT\_BUNELKOVETOK\_SZAMA, $this->regisztralt\_bunelkovetok\_szama);

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_GAZFOGYASZTOK\_SZAMA)) {

$criteria->add(StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_GAZFOGYASZTOK\_SZAMA, $this->haztartasi\_gazfogyasztok\_szama);

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA)) {

$criteria->add(StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA, $this->haztartasi\_villamosenergia\_fogyasztok\_szama);

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_ELVANDORLASOK\_SZAMA)) {

$criteria->add(StgErintettNepessegSzamaTableMap::COL\_ELVANDORLASOK\_SZAMA, $this->elvandorlasok\_szama);

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA)) {

$criteria->add(StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA, $this->allando\_nepesseg\_szama);

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA)) {

$criteria->add(StgErintettNepessegSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA, $this->vallalkozasok\_szama);

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA)) {

$criteria->add(StgErintettNepessegSzamaTableMap::COL\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA, $this->nyilvantartott\_allaskeresok\_szama);

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_LAKASALLOMANY)) {

$criteria->add(StgErintettNepessegSzamaTableMap::COL\_LAKASALLOMANY, $this->lakasallomany);

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA)) {

$criteria->add(StgErintettNepessegSzamaTableMap::COL\_BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA, $this->bolcsodebe\_beirt\_gyermekek\_szama);

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA)) {

$criteria->add(StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA, $this->allando\_nepessegbol\_18\_59\_evesek\_szama);

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_MOBILELOFIZETESEK\_SZAMA)) {

$criteria->add(StgErintettNepessegSzamaTableMap::COL\_MOBILELOFIZETESEK\_SZAMA, $this->mobilelofizetesek\_szama);

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADOFIZETO)) {

$criteria->add(StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADOFIZETO, $this->osszes\_adofizeto);

}

if ($this->isColumnModified(StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADO)) {

$criteria->add(StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADO, $this->osszes\_ado);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgErintettNepessegSzama object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgErintettNepessegSzama (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setRegisztraltBunelkovetokSzama($this->getRegisztraltBunelkovetokSzama());

$copyObj->setHaztartasiGazfogyasztokSzama($this->getHaztartasiGazfogyasztokSzama());

$copyObj->setHaztartasiVillamosenergiaFogyasztokSzama($this->getHaztartasiVillamosenergiaFogyasztokSzama());

$copyObj->setElvandorlasokSzama($this->getElvandorlasokSzama());

$copyObj->setAllandoNepessegSzama($this->getAllandoNepessegSzama());

$copyObj->setVallalkozasokSzama($this->getVallalkozasokSzama());

$copyObj->setNyilvantartottAllaskeresokSzama($this->getNyilvantartottAllaskeresokSzama());

$copyObj->setLakasallomany($this->getLakasallomany());

$copyObj->setBolcsodebeBeirtGyermekekSzama($this->getBolcsodebeBeirtGyermekekSzama());

$copyObj->setAllandoNepessegbol1859EvesekSzama($this->getAllandoNepessegbol1859EvesekSzama());

$copyObj->setMobilelofizetesekSzama($this->getMobilelofizetesekSzama());

$copyObj->setOsszesAdofizeto($this->getOsszesAdofizeto());

$copyObj->setOsszesAdo($this->getOsszesAdo());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgErintettNepessegSzama Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->regisztralt\_bunelkovetok\_szama = null;

$this->haztartasi\_gazfogyasztok\_szama = null;

$this->haztartasi\_villamosenergia\_fogyasztok\_szama = null;

$this->elvandorlasok\_szama = null;

$this->allando\_nepesseg\_szama = null;

$this->vallalkozasok\_szama = null;

$this->nyilvantartott\_allaskeresok\_szama = null;

$this->lakasallomany = null;

$this->bolcsodebe\_beirt\_gyermekek\_szama = null;

$this->allando\_nepessegbol\_18\_59\_evesek\_szama = null;

$this->mobilelofizetesek\_szama = null;

$this->osszes\_adofizeto = null;

$this->osszes\_ado = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgErintettNepessegSzamaTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgErintettNepessegSzamaQuery.php

<?php

namespace Base;

use \StgErintettNepessegSzama as ChildStgErintettNepessegSzama;

use \StgErintettNepessegSzamaQuery as ChildStgErintettNepessegSzamaQuery;

use \Exception;

use Map\StgErintettNepessegSzamaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_erintett\_nepesseg\_szama' table.

\*

\*

\*

\* @method ChildStgErintettNepessegSzamaQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgErintettNepessegSzamaQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgErintettNepessegSzamaQuery orderByRegisztraltBunelkovetokSzama($order = Criteria::ASC) Order by the regisztralt\_bunelkovetok\_szama column

\* @method ChildStgErintettNepessegSzamaQuery orderByHaztartasiGazfogyasztokSzama($order = Criteria::ASC) Order by the haztartasi\_gazfogyasztok\_szama column

\* @method ChildStgErintettNepessegSzamaQuery orderByHaztartasiVillamosenergiaFogyasztokSzama($order = Criteria::ASC) Order by the haztartasi\_villamosenergia\_fogyasztok\_szama column

\* @method ChildStgErintettNepessegSzamaQuery orderByElvandorlasokSzama($order = Criteria::ASC) Order by the elvandorlasok\_szama column

\* @method ChildStgErintettNepessegSzamaQuery orderByAllandoNepessegSzama($order = Criteria::ASC) Order by the allando\_nepesseg\_szama column

\* @method ChildStgErintettNepessegSzamaQuery orderByVallalkozasokSzama($order = Criteria::ASC) Order by the vallalkozasok\_szama column

\* @method ChildStgErintettNepessegSzamaQuery orderByNyilvantartottAllaskeresokSzama($order = Criteria::ASC) Order by the nyilvantartott\_allaskeresok\_szama column

\* @method ChildStgErintettNepessegSzamaQuery orderByLakasallomany($order = Criteria::ASC) Order by the lakasallomany column

\* @method ChildStgErintettNepessegSzamaQuery orderByBolcsodebeBeirtGyermekekSzama($order = Criteria::ASC) Order by the bolcsodebe\_beirt\_gyermekek\_szama column

\* @method ChildStgErintettNepessegSzamaQuery orderByAllandoNepessegbol1859EvesekSzama($order = Criteria::ASC) Order by the allando\_nepessegbol\_18\_59\_evesek\_szama column

\* @method ChildStgErintettNepessegSzamaQuery orderByMobilelofizetesekSzama($order = Criteria::ASC) Order by the mobilelofizetesek\_szama column

\* @method ChildStgErintettNepessegSzamaQuery orderByOsszesAdofizeto($order = Criteria::ASC) Order by the osszes\_adofizeto column

\* @method ChildStgErintettNepessegSzamaQuery orderByOsszesAdo($order = Criteria::ASC) Order by the osszes\_ado column

\*

\* @method ChildStgErintettNepessegSzamaQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgErintettNepessegSzamaQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgErintettNepessegSzamaQuery groupByRegisztraltBunelkovetokSzama() Group by the regisztralt\_bunelkovetok\_szama column

\* @method ChildStgErintettNepessegSzamaQuery groupByHaztartasiGazfogyasztokSzama() Group by the haztartasi\_gazfogyasztok\_szama column

\* @method ChildStgErintettNepessegSzamaQuery groupByHaztartasiVillamosenergiaFogyasztokSzama() Group by the haztartasi\_villamosenergia\_fogyasztok\_szama column

\* @method ChildStgErintettNepessegSzamaQuery groupByElvandorlasokSzama() Group by the elvandorlasok\_szama column

\* @method ChildStgErintettNepessegSzamaQuery groupByAllandoNepessegSzama() Group by the allando\_nepesseg\_szama column

\* @method ChildStgErintettNepessegSzamaQuery groupByVallalkozasokSzama() Group by the vallalkozasok\_szama column

\* @method ChildStgErintettNepessegSzamaQuery groupByNyilvantartottAllaskeresokSzama() Group by the nyilvantartott\_allaskeresok\_szama column

\* @method ChildStgErintettNepessegSzamaQuery groupByLakasallomany() Group by the lakasallomany column

\* @method ChildStgErintettNepessegSzamaQuery groupByBolcsodebeBeirtGyermekekSzama() Group by the bolcsodebe\_beirt\_gyermekek\_szama column

\* @method ChildStgErintettNepessegSzamaQuery groupByAllandoNepessegbol1859EvesekSzama() Group by the allando\_nepessegbol\_18\_59\_evesek\_szama column

\* @method ChildStgErintettNepessegSzamaQuery groupByMobilelofizetesekSzama() Group by the mobilelofizetesek\_szama column

\* @method ChildStgErintettNepessegSzamaQuery groupByOsszesAdofizeto() Group by the osszes\_adofizeto column

\* @method ChildStgErintettNepessegSzamaQuery groupByOsszesAdo() Group by the osszes\_ado column

\*

\* @method ChildStgErintettNepessegSzamaQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgErintettNepessegSzamaQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgErintettNepessegSzamaQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgErintettNepessegSzama findOne(ConnectionInterface $con = null) Return the first ChildStgErintettNepessegSzama matching the query

\* @method ChildStgErintettNepessegSzama findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgErintettNepessegSzama matching the query, or a new ChildStgErintettNepessegSzama object populated from the query conditions when no match is found

\*

\* @method ChildStgErintettNepessegSzama findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgErintettNepessegSzama filtered by the telepules\_nev column

\* @method ChildStgErintettNepessegSzama findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgErintettNepessegSzama filtered by the telepules\_KSHKOD column

\* @method ChildStgErintettNepessegSzama findOneByRegisztraltBunelkovetokSzama(int $regisztralt\_bunelkovetok\_szama) Return the first ChildStgErintettNepessegSzama filtered by the regisztralt\_bunelkovetok\_szama column

\* @method ChildStgErintettNepessegSzama findOneByHaztartasiGazfogyasztokSzama(int $haztartasi\_gazfogyasztok\_szama) Return the first ChildStgErintettNepessegSzama filtered by the haztartasi\_gazfogyasztok\_szama column

\* @method ChildStgErintettNepessegSzama findOneByHaztartasiVillamosenergiaFogyasztokSzama(int $haztartasi\_villamosenergia\_fogyasztok\_szama) Return the first ChildStgErintettNepessegSzama filtered by the haztartasi\_villamosenergia\_fogyasztok\_szama column

\* @method ChildStgErintettNepessegSzama findOneByElvandorlasokSzama(int $elvandorlasok\_szama) Return the first ChildStgErintettNepessegSzama filtered by the elvandorlasok\_szama column

\* @method ChildStgErintettNepessegSzama findOneByAllandoNepessegSzama(int $allando\_nepesseg\_szama) Return the first ChildStgErintettNepessegSzama filtered by the allando\_nepesseg\_szama column

\* @method ChildStgErintettNepessegSzama findOneByVallalkozasokSzama(int $vallalkozasok\_szama) Return the first ChildStgErintettNepessegSzama filtered by the vallalkozasok\_szama column

\* @method ChildStgErintettNepessegSzama findOneByNyilvantartottAllaskeresokSzama(int $nyilvantartott\_allaskeresok\_szama) Return the first ChildStgErintettNepessegSzama filtered by the nyilvantartott\_allaskeresok\_szama column

\* @method ChildStgErintettNepessegSzama findOneByLakasallomany(int $lakasallomany) Return the first ChildStgErintettNepessegSzama filtered by the lakasallomany column

\* @method ChildStgErintettNepessegSzama findOneByBolcsodebeBeirtGyermekekSzama(int $bolcsodebe\_beirt\_gyermekek\_szama) Return the first ChildStgErintettNepessegSzama filtered by the bolcsodebe\_beirt\_gyermekek\_szama column

\* @method ChildStgErintettNepessegSzama findOneByAllandoNepessegbol1859EvesekSzama(int $allando\_nepessegbol\_18\_59\_evesek\_szama) Return the first ChildStgErintettNepessegSzama filtered by the allando\_nepessegbol\_18\_59\_evesek\_szama column

\* @method ChildStgErintettNepessegSzama findOneByMobilelofizetesekSzama(int $mobilelofizetesek\_szama) Return the first ChildStgErintettNepessegSzama filtered by the mobilelofizetesek\_szama column

\* @method ChildStgErintettNepessegSzama findOneByOsszesAdofizeto(int $osszes\_adofizeto) Return the first ChildStgErintettNepessegSzama filtered by the osszes\_adofizeto column

\* @method ChildStgErintettNepessegSzama findOneByOsszesAdo(int $osszes\_ado) Return the first ChildStgErintettNepessegSzama filtered by the osszes\_ado column

\*

\* @method ChildStgErintettNepessegSzama[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgErintettNepessegSzama objects based on current ModelCriteria

\* @method ChildStgErintettNepessegSzama[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgErintettNepessegSzama objects filtered by the telepules\_nev column

\* @method ChildStgErintettNepessegSzama[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgErintettNepessegSzama objects filtered by the telepules\_KSHKOD column

\* @method ChildStgErintettNepessegSzama[]|ObjectCollection findByRegisztraltBunelkovetokSzama(int $regisztralt\_bunelkovetok\_szama) Return ChildStgErintettNepessegSzama objects filtered by the regisztralt\_bunelkovetok\_szama column

\* @method ChildStgErintettNepessegSzama[]|ObjectCollection findByHaztartasiGazfogyasztokSzama(int $haztartasi\_gazfogyasztok\_szama) Return ChildStgErintettNepessegSzama objects filtered by the haztartasi\_gazfogyasztok\_szama column

\* @method ChildStgErintettNepessegSzama[]|ObjectCollection findByHaztartasiVillamosenergiaFogyasztokSzama(int $haztartasi\_villamosenergia\_fogyasztok\_szama) Return ChildStgErintettNepessegSzama objects filtered by the haztartasi\_villamosenergia\_fogyasztok\_szama column

\* @method ChildStgErintettNepessegSzama[]|ObjectCollection findByElvandorlasokSzama(int $elvandorlasok\_szama) Return ChildStgErintettNepessegSzama objects filtered by the elvandorlasok\_szama column

\* @method ChildStgErintettNepessegSzama[]|ObjectCollection findByAllandoNepessegSzama(int $allando\_nepesseg\_szama) Return ChildStgErintettNepessegSzama objects filtered by the allando\_nepesseg\_szama column

\* @method ChildStgErintettNepessegSzama[]|ObjectCollection findByVallalkozasokSzama(int $vallalkozasok\_szama) Return ChildStgErintettNepessegSzama objects filtered by the vallalkozasok\_szama column

\* @method ChildStgErintettNepessegSzama[]|ObjectCollection findByNyilvantartottAllaskeresokSzama(int $nyilvantartott\_allaskeresok\_szama) Return ChildStgErintettNepessegSzama objects filtered by the nyilvantartott\_allaskeresok\_szama column

\* @method ChildStgErintettNepessegSzama[]|ObjectCollection findByLakasallomany(int $lakasallomany) Return ChildStgErintettNepessegSzama objects filtered by the lakasallomany column

\* @method ChildStgErintettNepessegSzama[]|ObjectCollection findByBolcsodebeBeirtGyermekekSzama(int $bolcsodebe\_beirt\_gyermekek\_szama) Return ChildStgErintettNepessegSzama objects filtered by the bolcsodebe\_beirt\_gyermekek\_szama column

\* @method ChildStgErintettNepessegSzama[]|ObjectCollection findByAllandoNepessegbol1859EvesekSzama(int $allando\_nepessegbol\_18\_59\_evesek\_szama) Return ChildStgErintettNepessegSzama objects filtered by the allando\_nepessegbol\_18\_59\_evesek\_szama column

\* @method ChildStgErintettNepessegSzama[]|ObjectCollection findByMobilelofizetesekSzama(int $mobilelofizetesek\_szama) Return ChildStgErintettNepessegSzama objects filtered by the mobilelofizetesek\_szama column

\* @method ChildStgErintettNepessegSzama[]|ObjectCollection findByOsszesAdofizeto(int $osszes\_adofizeto) Return ChildStgErintettNepessegSzama objects filtered by the osszes\_adofizeto column

\* @method ChildStgErintettNepessegSzama[]|ObjectCollection findByOsszesAdo(int $osszes\_ado) Return ChildStgErintettNepessegSzama objects filtered by the osszes\_ado column

\* @method ChildStgErintettNepessegSzama[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgErintettNepessegSzamaQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgErintettNepessegSzamaQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgErintettNepessegSzama', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgErintettNepessegSzamaQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgErintettNepessegSzamaQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgErintettNepessegSzamaQuery) {

return $criteria;

}

$query = new ChildStgErintettNepessegSzamaQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgErintettNepessegSzama|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgErintettNepessegSzama object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgErintettNepessegSzama object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgErintettNepessegSzama object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgErintettNepessegSzama object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the regisztralt\_bunelkovetok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByRegisztraltBunelkovetokSzama(1234); // WHERE regisztralt\_bunelkovetok\_szama = 1234

\* $query->filterByRegisztraltBunelkovetokSzama(array(12, 34)); // WHERE regisztralt\_bunelkovetok\_szama IN (12, 34)

\* $query->filterByRegisztraltBunelkovetokSzama(array('min' => 12)); // WHERE regisztralt\_bunelkovetok\_szama > 12

\* </code>

\*

\* @param mixed $regisztraltBunelkovetokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function filterByRegisztraltBunelkovetokSzama($regisztraltBunelkovetokSzama = null, $comparison = null)

{

if (is\_array($regisztraltBunelkovetokSzama)) {

$useMinMax = false;

if (isset($regisztraltBunelkovetokSzama['min'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_REGISZTRALT\_BUNELKOVETOK\_SZAMA, $regisztraltBunelkovetokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($regisztraltBunelkovetokSzama['max'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_REGISZTRALT\_BUNELKOVETOK\_SZAMA, $regisztraltBunelkovetokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_REGISZTRALT\_BUNELKOVETOK\_SZAMA, $regisztraltBunelkovetokSzama, $comparison);

}

/\*\*

\* Filter the query on the haztartasi\_gazfogyasztok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByHaztartasiGazfogyasztokSzama(1234); // WHERE haztartasi\_gazfogyasztok\_szama = 1234

\* $query->filterByHaztartasiGazfogyasztokSzama(array(12, 34)); // WHERE haztartasi\_gazfogyasztok\_szama IN (12, 34)

\* $query->filterByHaztartasiGazfogyasztokSzama(array('min' => 12)); // WHERE haztartasi\_gazfogyasztok\_szama > 12

\* </code>

\*

\* @param mixed $haztartasiGazfogyasztokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function filterByHaztartasiGazfogyasztokSzama($haztartasiGazfogyasztokSzama = null, $comparison = null)

{

if (is\_array($haztartasiGazfogyasztokSzama)) {

$useMinMax = false;

if (isset($haztartasiGazfogyasztokSzama['min'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_GAZFOGYASZTOK\_SZAMA, $haztartasiGazfogyasztokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($haztartasiGazfogyasztokSzama['max'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_GAZFOGYASZTOK\_SZAMA, $haztartasiGazfogyasztokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_GAZFOGYASZTOK\_SZAMA, $haztartasiGazfogyasztokSzama, $comparison);

}

/\*\*

\* Filter the query on the haztartasi\_villamosenergia\_fogyasztok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByHaztartasiVillamosenergiaFogyasztokSzama(1234); // WHERE haztartasi\_villamosenergia\_fogyasztok\_szama = 1234

\* $query->filterByHaztartasiVillamosenergiaFogyasztokSzama(array(12, 34)); // WHERE haztartasi\_villamosenergia\_fogyasztok\_szama IN (12, 34)

\* $query->filterByHaztartasiVillamosenergiaFogyasztokSzama(array('min' => 12)); // WHERE haztartasi\_villamosenergia\_fogyasztok\_szama > 12

\* </code>

\*

\* @param mixed $haztartasiVillamosenergiaFogyasztokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function filterByHaztartasiVillamosenergiaFogyasztokSzama($haztartasiVillamosenergiaFogyasztokSzama = null, $comparison = null)

{

if (is\_array($haztartasiVillamosenergiaFogyasztokSzama)) {

$useMinMax = false;

if (isset($haztartasiVillamosenergiaFogyasztokSzama['min'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA, $haztartasiVillamosenergiaFogyasztokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($haztartasiVillamosenergiaFogyasztokSzama['max'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA, $haztartasiVillamosenergiaFogyasztokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA, $haztartasiVillamosenergiaFogyasztokSzama, $comparison);

}

/\*\*

\* Filter the query on the elvandorlasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByElvandorlasokSzama(1234); // WHERE elvandorlasok\_szama = 1234

\* $query->filterByElvandorlasokSzama(array(12, 34)); // WHERE elvandorlasok\_szama IN (12, 34)

\* $query->filterByElvandorlasokSzama(array('min' => 12)); // WHERE elvandorlasok\_szama > 12

\* </code>

\*

\* @param mixed $elvandorlasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function filterByElvandorlasokSzama($elvandorlasokSzama = null, $comparison = null)

{

if (is\_array($elvandorlasokSzama)) {

$useMinMax = false;

if (isset($elvandorlasokSzama['min'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_ELVANDORLASOK\_SZAMA, $elvandorlasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($elvandorlasokSzama['max'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_ELVANDORLASOK\_SZAMA, $elvandorlasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_ELVANDORLASOK\_SZAMA, $elvandorlasokSzama, $comparison);

}

/\*\*

\* Filter the query on the allando\_nepesseg\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByAllandoNepessegSzama(1234); // WHERE allando\_nepesseg\_szama = 1234

\* $query->filterByAllandoNepessegSzama(array(12, 34)); // WHERE allando\_nepesseg\_szama IN (12, 34)

\* $query->filterByAllandoNepessegSzama(array('min' => 12)); // WHERE allando\_nepesseg\_szama > 12

\* </code>

\*

\* @param mixed $allandoNepessegSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function filterByAllandoNepessegSzama($allandoNepessegSzama = null, $comparison = null)

{

if (is\_array($allandoNepessegSzama)) {

$useMinMax = false;

if (isset($allandoNepessegSzama['min'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA, $allandoNepessegSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($allandoNepessegSzama['max'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA, $allandoNepessegSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA, $allandoNepessegSzama, $comparison);

}

/\*\*

\* Filter the query on the vallalkozasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByVallalkozasokSzama(1234); // WHERE vallalkozasok\_szama = 1234

\* $query->filterByVallalkozasokSzama(array(12, 34)); // WHERE vallalkozasok\_szama IN (12, 34)

\* $query->filterByVallalkozasokSzama(array('min' => 12)); // WHERE vallalkozasok\_szama > 12

\* </code>

\*

\* @param mixed $vallalkozasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function filterByVallalkozasokSzama($vallalkozasokSzama = null, $comparison = null)

{

if (is\_array($vallalkozasokSzama)) {

$useMinMax = false;

if (isset($vallalkozasokSzama['min'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA, $vallalkozasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($vallalkozasokSzama['max'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA, $vallalkozasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA, $vallalkozasokSzama, $comparison);

}

/\*\*

\* Filter the query on the nyilvantartott\_allaskeresok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByNyilvantartottAllaskeresokSzama(1234); // WHERE nyilvantartott\_allaskeresok\_szama = 1234

\* $query->filterByNyilvantartottAllaskeresokSzama(array(12, 34)); // WHERE nyilvantartott\_allaskeresok\_szama IN (12, 34)

\* $query->filterByNyilvantartottAllaskeresokSzama(array('min' => 12)); // WHERE nyilvantartott\_allaskeresok\_szama > 12

\* </code>

\*

\* @param mixed $nyilvantartottAllaskeresokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function filterByNyilvantartottAllaskeresokSzama($nyilvantartottAllaskeresokSzama = null, $comparison = null)

{

if (is\_array($nyilvantartottAllaskeresokSzama)) {

$useMinMax = false;

if (isset($nyilvantartottAllaskeresokSzama['min'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA, $nyilvantartottAllaskeresokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nyilvantartottAllaskeresokSzama['max'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA, $nyilvantartottAllaskeresokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA, $nyilvantartottAllaskeresokSzama, $comparison);

}

/\*\*

\* Filter the query on the lakasallomany column

\*

\* Example usage:

\* <code>

\* $query->filterByLakasallomany(1234); // WHERE lakasallomany = 1234

\* $query->filterByLakasallomany(array(12, 34)); // WHERE lakasallomany IN (12, 34)

\* $query->filterByLakasallomany(array('min' => 12)); // WHERE lakasallomany > 12

\* </code>

\*

\* @param mixed $lakasallomany The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function filterByLakasallomany($lakasallomany = null, $comparison = null)

{

if (is\_array($lakasallomany)) {

$useMinMax = false;

if (isset($lakasallomany['min'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_LAKASALLOMANY, $lakasallomany['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($lakasallomany['max'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_LAKASALLOMANY, $lakasallomany['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_LAKASALLOMANY, $lakasallomany, $comparison);

}

/\*\*

\* Filter the query on the bolcsodebe\_beirt\_gyermekek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByBolcsodebeBeirtGyermekekSzama(1234); // WHERE bolcsodebe\_beirt\_gyermekek\_szama = 1234

\* $query->filterByBolcsodebeBeirtGyermekekSzama(array(12, 34)); // WHERE bolcsodebe\_beirt\_gyermekek\_szama IN (12, 34)

\* $query->filterByBolcsodebeBeirtGyermekekSzama(array('min' => 12)); // WHERE bolcsodebe\_beirt\_gyermekek\_szama > 12

\* </code>

\*

\* @param mixed $bolcsodebeBeirtGyermekekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function filterByBolcsodebeBeirtGyermekekSzama($bolcsodebeBeirtGyermekekSzama = null, $comparison = null)

{

if (is\_array($bolcsodebeBeirtGyermekekSzama)) {

$useMinMax = false;

if (isset($bolcsodebeBeirtGyermekekSzama['min'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA, $bolcsodebeBeirtGyermekekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($bolcsodebeBeirtGyermekekSzama['max'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA, $bolcsodebeBeirtGyermekekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA, $bolcsodebeBeirtGyermekekSzama, $comparison);

}

/\*\*

\* Filter the query on the allando\_nepessegbol\_18\_59\_evesek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByAllandoNepessegbol1859EvesekSzama(1234); // WHERE allando\_nepessegbol\_18\_59\_evesek\_szama = 1234

\* $query->filterByAllandoNepessegbol1859EvesekSzama(array(12, 34)); // WHERE allando\_nepessegbol\_18\_59\_evesek\_szama IN (12, 34)

\* $query->filterByAllandoNepessegbol1859EvesekSzama(array('min' => 12)); // WHERE allando\_nepessegbol\_18\_59\_evesek\_szama > 12

\* </code>

\*

\* @param mixed $allandoNepessegbol1859EvesekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function filterByAllandoNepessegbol1859EvesekSzama($allandoNepessegbol1859EvesekSzama = null, $comparison = null)

{

if (is\_array($allandoNepessegbol1859EvesekSzama)) {

$useMinMax = false;

if (isset($allandoNepessegbol1859EvesekSzama['min'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA, $allandoNepessegbol1859EvesekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($allandoNepessegbol1859EvesekSzama['max'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA, $allandoNepessegbol1859EvesekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA, $allandoNepessegbol1859EvesekSzama, $comparison);

}

/\*\*

\* Filter the query on the mobilelofizetesek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByMobilelofizetesekSzama(1234); // WHERE mobilelofizetesek\_szama = 1234

\* $query->filterByMobilelofizetesekSzama(array(12, 34)); // WHERE mobilelofizetesek\_szama IN (12, 34)

\* $query->filterByMobilelofizetesekSzama(array('min' => 12)); // WHERE mobilelofizetesek\_szama > 12

\* </code>

\*

\* @param mixed $mobilelofizetesekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function filterByMobilelofizetesekSzama($mobilelofizetesekSzama = null, $comparison = null)

{

if (is\_array($mobilelofizetesekSzama)) {

$useMinMax = false;

if (isset($mobilelofizetesekSzama['min'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_MOBILELOFIZETESEK\_SZAMA, $mobilelofizetesekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($mobilelofizetesekSzama['max'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_MOBILELOFIZETESEK\_SZAMA, $mobilelofizetesekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_MOBILELOFIZETESEK\_SZAMA, $mobilelofizetesekSzama, $comparison);

}

/\*\*

\* Filter the query on the osszes\_adofizeto column

\*

\* Example usage:

\* <code>

\* $query->filterByOsszesAdofizeto(1234); // WHERE osszes\_adofizeto = 1234

\* $query->filterByOsszesAdofizeto(array(12, 34)); // WHERE osszes\_adofizeto IN (12, 34)

\* $query->filterByOsszesAdofizeto(array('min' => 12)); // WHERE osszes\_adofizeto > 12

\* </code>

\*

\* @param mixed $osszesAdofizeto The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function filterByOsszesAdofizeto($osszesAdofizeto = null, $comparison = null)

{

if (is\_array($osszesAdofizeto)) {

$useMinMax = false;

if (isset($osszesAdofizeto['min'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADOFIZETO, $osszesAdofizeto['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($osszesAdofizeto['max'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADOFIZETO, $osszesAdofizeto['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADOFIZETO, $osszesAdofizeto, $comparison);

}

/\*\*

\* Filter the query on the osszes\_ado column

\*

\* Example usage:

\* <code>

\* $query->filterByOsszesAdo(1234); // WHERE osszes\_ado = 1234

\* $query->filterByOsszesAdo(array(12, 34)); // WHERE osszes\_ado IN (12, 34)

\* $query->filterByOsszesAdo(array('min' => 12)); // WHERE osszes\_ado > 12

\* </code>

\*

\* @param mixed $osszesAdo The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function filterByOsszesAdo($osszesAdo = null, $comparison = null)

{

if (is\_array($osszesAdo)) {

$useMinMax = false;

if (isset($osszesAdo['min'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADO, $osszesAdo['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($osszesAdo['max'])) {

$this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADO, $osszesAdo['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADO, $osszesAdo, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgErintettNepessegSzama $stgErintettNepessegSzama Object to remove from the list of results

\*

\* @return $this|ChildStgErintettNepessegSzamaQuery The current query, for fluid interface

\*/

public function prune($stgErintettNepessegSzama = null)

{

if ($stgErintettNepessegSzama) {

throw new LogicException('StgErintettNepessegSzama object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_erintett\_nepesseg\_szama table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgErintettNepessegSzamaTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgErintettNepessegSzamaTableMap::clearInstancePool();

StgErintettNepessegSzamaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgErintettNepessegSzamaTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgErintettNepessegSzamaTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgErintettNepessegSzamaTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgErintettNepessegSzamaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgErintettNepessegSzamaQuery

#### StgErtekesitesNettoArbevetele.php

<?php

namespace Base;

use \StgErtekesitesNettoArbeveteleQuery as ChildStgErtekesitesNettoArbeveteleQuery;

use \Exception;

use \PDO;

use Map\StgErtekesitesNettoArbeveteleTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgErtekesitesNettoArbevetele implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgErtekesitesNettoArbeveteleTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the ertekesites\_netto\_arbevetele field.

\* @var int

\*/

protected $ertekesites\_netto\_arbevetele;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgErtekesitesNettoArbevetele object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgErtekesitesNettoArbevetele</code> instance. If

\* <code>obj</code> is an instance of <code>StgErtekesitesNettoArbevetele</code>, delegates to

\* <code>equals(StgErtekesitesNettoArbevetele)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgErtekesitesNettoArbevetele The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [ertekesites\_netto\_arbevetele] column value.

\*

\* @return int

\*/

public function getErtekesitesNettoArbevetele()

{

return $this->ertekesites\_netto\_arbevetele;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgErtekesitesNettoArbeveteleTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgErtekesitesNettoArbeveteleTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgErtekesitesNettoArbeveteleTableMap::translateFieldName('ErtekesitesNettoArbevetele', TableMap::TYPE\_PHPNAME, $indexType)];

$this->ertekesites\_netto\_arbevetele = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 3; // 3 = StgErtekesitesNettoArbeveteleTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgErtekesitesNettoArbevetele'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgErtekesitesNettoArbevetele The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgErtekesitesNettoArbeveteleTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgErtekesitesNettoArbevetele The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgErtekesitesNettoArbeveteleTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [ertekesites\_netto\_arbevetele] column.

\*

\* @param int $v new value

\* @return $this|\StgErtekesitesNettoArbevetele The current object (for fluent API support)

\*/

public function setErtekesitesNettoArbevetele($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->ertekesites\_netto\_arbevetele !== $v) {

$this->ertekesites\_netto\_arbevetele = $v;

$this->modifiedColumns[StgErtekesitesNettoArbeveteleTableMap::COL\_ERTEKESITES\_NETTO\_ARBEVETELE] = true;

}

return $this;

} // setErtekesitesNettoArbevetele()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgErtekesitesNettoArbeveteleTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgErtekesitesNettoArbeveteleQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgErtekesitesNettoArbevetele::setDeleted()

\* @see StgErtekesitesNettoArbevetele::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgErtekesitesNettoArbeveteleTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgErtekesitesNettoArbeveteleQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgErtekesitesNettoArbeveteleTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgErtekesitesNettoArbeveteleTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgErtekesitesNettoArbeveteleTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgErtekesitesNettoArbeveteleTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgErtekesitesNettoArbeveteleTableMap::COL\_ERTEKESITES\_NETTO\_ARBEVETELE)) {

$modifiedColumns[':p' . $index++] = 'ERTEKESITES\_NETTO\_ARBEVETELE';

}

$sql = sprintf(

'INSERT INTO stg\_ertekesites\_netto\_arbevetele (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'ERTEKESITES\_NETTO\_ARBEVETELE':

$stmt->bindValue($identifier, $this->ertekesites\_netto\_arbevetele, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgErtekesitesNettoArbeveteleTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getErtekesitesNettoArbevetele();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgErtekesitesNettoArbevetele'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgErtekesitesNettoArbevetele'][$this->getPrimaryKey()] = true;

$keys = StgErtekesitesNettoArbeveteleTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getErtekesitesNettoArbevetele(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgErtekesitesNettoArbevetele

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgErtekesitesNettoArbeveteleTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgErtekesitesNettoArbevetele

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setErtekesitesNettoArbevetele($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgErtekesitesNettoArbeveteleTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setErtekesitesNettoArbevetele($arr[$keys[2]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgErtekesitesNettoArbevetele The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgErtekesitesNettoArbeveteleTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgErtekesitesNettoArbeveteleTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgErtekesitesNettoArbeveteleTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgErtekesitesNettoArbeveteleTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgErtekesitesNettoArbeveteleTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgErtekesitesNettoArbeveteleTableMap::COL\_ERTEKESITES\_NETTO\_ARBEVETELE)) {

$criteria->add(StgErtekesitesNettoArbeveteleTableMap::COL\_ERTEKESITES\_NETTO\_ARBEVETELE, $this->ertekesites\_netto\_arbevetele);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgErtekesitesNettoArbevetele object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgErtekesitesNettoArbevetele (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setErtekesitesNettoArbevetele($this->getErtekesitesNettoArbevetele());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgErtekesitesNettoArbevetele Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->ertekesites\_netto\_arbevetele = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgErtekesitesNettoArbeveteleTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgErtekesitesNettoArbeveteleQuery.php

<?php

namespace Base;

use \StgErtekesitesNettoArbevetele as ChildStgErtekesitesNettoArbevetele;

use \StgErtekesitesNettoArbeveteleQuery as ChildStgErtekesitesNettoArbeveteleQuery;

use \Exception;

use Map\StgErtekesitesNettoArbeveteleTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_ertekesites\_netto\_arbevetele' table.

\*

\*

\*

\* @method ChildStgErtekesitesNettoArbeveteleQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgErtekesitesNettoArbeveteleQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgErtekesitesNettoArbeveteleQuery orderByErtekesitesNettoArbevetele($order = Criteria::ASC) Order by the ertekesites\_netto\_arbevetele column

\*

\* @method ChildStgErtekesitesNettoArbeveteleQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgErtekesitesNettoArbeveteleQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgErtekesitesNettoArbeveteleQuery groupByErtekesitesNettoArbevetele() Group by the ertekesites\_netto\_arbevetele column

\*

\* @method ChildStgErtekesitesNettoArbeveteleQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgErtekesitesNettoArbeveteleQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgErtekesitesNettoArbeveteleQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgErtekesitesNettoArbevetele findOne(ConnectionInterface $con = null) Return the first ChildStgErtekesitesNettoArbevetele matching the query

\* @method ChildStgErtekesitesNettoArbevetele findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgErtekesitesNettoArbevetele matching the query, or a new ChildStgErtekesitesNettoArbevetele object populated from the query conditions when no match is found

\*

\* @method ChildStgErtekesitesNettoArbevetele findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgErtekesitesNettoArbevetele filtered by the telepules\_nev column

\* @method ChildStgErtekesitesNettoArbevetele findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgErtekesitesNettoArbevetele filtered by the telepules\_KSHKOD column

\* @method ChildStgErtekesitesNettoArbevetele findOneByErtekesitesNettoArbevetele(int $ertekesites\_netto\_arbevetele) Return the first ChildStgErtekesitesNettoArbevetele filtered by the ertekesites\_netto\_arbevetele column

\*

\* @method ChildStgErtekesitesNettoArbevetele[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgErtekesitesNettoArbevetele objects based on current ModelCriteria

\* @method ChildStgErtekesitesNettoArbevetele[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgErtekesitesNettoArbevetele objects filtered by the telepules\_nev column

\* @method ChildStgErtekesitesNettoArbevetele[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgErtekesitesNettoArbevetele objects filtered by the telepules\_KSHKOD column

\* @method ChildStgErtekesitesNettoArbevetele[]|ObjectCollection findByErtekesitesNettoArbevetele(int $ertekesites\_netto\_arbevetele) Return ChildStgErtekesitesNettoArbevetele objects filtered by the ertekesites\_netto\_arbevetele column

\* @method ChildStgErtekesitesNettoArbevetele[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgErtekesitesNettoArbeveteleQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgErtekesitesNettoArbeveteleQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgErtekesitesNettoArbevetele', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgErtekesitesNettoArbeveteleQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgErtekesitesNettoArbeveteleQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgErtekesitesNettoArbeveteleQuery) {

return $criteria;

}

$query = new ChildStgErtekesitesNettoArbeveteleQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgErtekesitesNettoArbevetele|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgErtekesitesNettoArbevetele object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgErtekesitesNettoArbevetele object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgErtekesitesNettoArbeveteleQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgErtekesitesNettoArbevetele object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgErtekesitesNettoArbeveteleQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgErtekesitesNettoArbevetele object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErtekesitesNettoArbeveteleQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgErtekesitesNettoArbeveteleTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErtekesitesNettoArbeveteleQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgErtekesitesNettoArbeveteleTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the ertekesites\_netto\_arbevetele column

\*

\* Example usage:

\* <code>

\* $query->filterByErtekesitesNettoArbevetele(1234); // WHERE ertekesites\_netto\_arbevetele = 1234

\* $query->filterByErtekesitesNettoArbevetele(array(12, 34)); // WHERE ertekesites\_netto\_arbevetele IN (12, 34)

\* $query->filterByErtekesitesNettoArbevetele(array('min' => 12)); // WHERE ertekesites\_netto\_arbevetele > 12

\* </code>

\*

\* @param mixed $ertekesitesNettoArbevetele The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgErtekesitesNettoArbeveteleQuery The current query, for fluid interface

\*/

public function filterByErtekesitesNettoArbevetele($ertekesitesNettoArbevetele = null, $comparison = null)

{

if (is\_array($ertekesitesNettoArbevetele)) {

$useMinMax = false;

if (isset($ertekesitesNettoArbevetele['min'])) {

$this->addUsingAlias(StgErtekesitesNettoArbeveteleTableMap::COL\_ERTEKESITES\_NETTO\_ARBEVETELE, $ertekesitesNettoArbevetele['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($ertekesitesNettoArbevetele['max'])) {

$this->addUsingAlias(StgErtekesitesNettoArbeveteleTableMap::COL\_ERTEKESITES\_NETTO\_ARBEVETELE, $ertekesitesNettoArbevetele['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgErtekesitesNettoArbeveteleTableMap::COL\_ERTEKESITES\_NETTO\_ARBEVETELE, $ertekesitesNettoArbevetele, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgErtekesitesNettoArbevetele $stgErtekesitesNettoArbevetele Object to remove from the list of results

\*

\* @return $this|ChildStgErtekesitesNettoArbeveteleQuery The current query, for fluid interface

\*/

public function prune($stgErtekesitesNettoArbevetele = null)

{

if ($stgErtekesitesNettoArbevetele) {

throw new LogicException('StgErtekesitesNettoArbevetele object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_ertekesites\_netto\_arbevetele table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgErtekesitesNettoArbeveteleTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgErtekesitesNettoArbeveteleTableMap::clearInstancePool();

StgErtekesitesNettoArbeveteleTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgErtekesitesNettoArbeveteleTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgErtekesitesNettoArbeveteleTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgErtekesitesNettoArbeveteleTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgErtekesitesNettoArbeveteleTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgErtekesitesNettoArbeveteleQuery

#### StgFoglalkoztatottakSzama.php

<?php

namespace Base;

use \StgFoglalkoztatottakSzamaQuery as ChildStgFoglalkoztatottakSzamaQuery;

use \Exception;

use \PDO;

use Map\StgFoglalkoztatottakSzamaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgFoglalkoztatottakSzama implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgFoglalkoztatottakSzamaTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the kisterseg\_nev field.

\* @var string

\*/

protected $kisterseg\_nev;

/\*\*

\* The value for the kisterseg\_kshkod field.

\* @var string

\*/

protected $kisterseg\_kshkod;

/\*\*

\* The value for the foglalkoztatottak\_szama field.

\* @var int

\*/

protected $foglalkoztatottak\_szama;

/\*\*

\* The value for the kistersegi\_foglalkoztatasi\_rata field.

\* @var double

\*/

protected $kistersegi\_foglalkoztatasi\_rata;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgFoglalkoztatottakSzama object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgFoglalkoztatottakSzama</code> instance. If

\* <code>obj</code> is an instance of <code>StgFoglalkoztatottakSzama</code>, delegates to

\* <code>equals(StgFoglalkoztatottakSzama)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgFoglalkoztatottakSzama The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [kisterseg\_nev] column value.

\*

\* @return string

\*/

public function getKistersegNev()

{

return $this->kisterseg\_nev;

}

/\*\*

\* Get the [kisterseg\_kshkod] column value.

\*

\* @return string

\*/

public function getKistersegKshkod()

{

return $this->kisterseg\_kshkod;

}

/\*\*

\* Get the [foglalkoztatottak\_szama] column value.

\*

\* @return int

\*/

public function getFoglalkoztatottakSzama()

{

return $this->foglalkoztatottak\_szama;

}

/\*\*

\* Get the [kistersegi\_foglalkoztatasi\_rata] column value.

\*

\* @return double

\*/

public function getKistersegiFoglalkoztatasiRata()

{

return $this->kistersegi\_foglalkoztatasi\_rata;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgFoglalkoztatottakSzamaTableMap::translateFieldName('KistersegNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kisterseg\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgFoglalkoztatottakSzamaTableMap::translateFieldName('KistersegKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kisterseg\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgFoglalkoztatottakSzamaTableMap::translateFieldName('FoglalkoztatottakSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->foglalkoztatottak\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgFoglalkoztatottakSzamaTableMap::translateFieldName('KistersegiFoglalkoztatasiRata', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kistersegi\_foglalkoztatasi\_rata = (null !== $col) ? (double) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 4; // 4 = StgFoglalkoztatottakSzamaTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgFoglalkoztatottakSzama'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [kisterseg\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgFoglalkoztatottakSzama The current object (for fluent API support)

\*/

public function setKistersegNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->kisterseg\_nev !== $v) {

$this->kisterseg\_nev = $v;

$this->modifiedColumns[StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEG\_NEV] = true;

}

return $this;

} // setKistersegNev()

/\*\*

\* Set the value of [kisterseg\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgFoglalkoztatottakSzama The current object (for fluent API support)

\*/

public function setKistersegKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->kisterseg\_kshkod !== $v) {

$this->kisterseg\_kshkod = $v;

$this->modifiedColumns[StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEG\_KSHKOD] = true;

}

return $this;

} // setKistersegKshkod()

/\*\*

\* Set the value of [foglalkoztatottak\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgFoglalkoztatottakSzama The current object (for fluent API support)

\*/

public function setFoglalkoztatottakSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->foglalkoztatottak\_szama !== $v) {

$this->foglalkoztatottak\_szama = $v;

$this->modifiedColumns[StgFoglalkoztatottakSzamaTableMap::COL\_FOGLALKOZTATOTTAK\_SZAMA] = true;

}

return $this;

} // setFoglalkoztatottakSzama()

/\*\*

\* Set the value of [kistersegi\_foglalkoztatasi\_rata] column.

\*

\* @param double $v new value

\* @return $this|\StgFoglalkoztatottakSzama The current object (for fluent API support)

\*/

public function setKistersegiFoglalkoztatasiRata($v)

{

if ($v !== null) {

$v = (double) $v;

}

if ($this->kistersegi\_foglalkoztatasi\_rata !== $v) {

$this->kistersegi\_foglalkoztatasi\_rata = $v;

$this->modifiedColumns[StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEGI\_FOGLALKOZTATASI\_RATA] = true;

}

return $this;

} // setKistersegiFoglalkoztatasiRata()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgFoglalkoztatottakSzamaTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgFoglalkoztatottakSzamaQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgFoglalkoztatottakSzama::setDeleted()

\* @see StgFoglalkoztatottakSzama::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgFoglalkoztatottakSzamaTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgFoglalkoztatottakSzamaQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgFoglalkoztatottakSzamaTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgFoglalkoztatottakSzamaTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEG\_NEV)) {

$modifiedColumns[':p' . $index++] = 'KISTERSEG\_NEV';

}

if ($this->isColumnModified(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEG\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'KISTERSEG\_KSHKOD';

}

if ($this->isColumnModified(StgFoglalkoztatottakSzamaTableMap::COL\_FOGLALKOZTATOTTAK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'FOGLALKOZTATOTTAK\_SZAMA';

}

if ($this->isColumnModified(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEGI\_FOGLALKOZTATASI\_RATA)) {

$modifiedColumns[':p' . $index++] = 'KISTERSEGI\_FOGLALKOZTATASI\_RATA';

}

$sql = sprintf(

'INSERT INTO stg\_foglalkoztatottak\_szama (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'KISTERSEG\_NEV':

$stmt->bindValue($identifier, $this->kisterseg\_nev, PDO::PARAM\_STR);

break;

case 'KISTERSEG\_KSHKOD':

$stmt->bindValue($identifier, $this->kisterseg\_kshkod, PDO::PARAM\_STR);

break;

case 'FOGLALKOZTATOTTAK\_SZAMA':

$stmt->bindValue($identifier, $this->foglalkoztatottak\_szama, PDO::PARAM\_INT);

break;

case 'KISTERSEGI\_FOGLALKOZTATASI\_RATA':

$stmt->bindValue($identifier, $this->kistersegi\_foglalkoztatasi\_rata, PDO::PARAM\_STR);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgFoglalkoztatottakSzamaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getKistersegNev();

break;

case 1:

return $this->getKistersegKshkod();

break;

case 2:

return $this->getFoglalkoztatottakSzama();

break;

case 3:

return $this->getKistersegiFoglalkoztatasiRata();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgFoglalkoztatottakSzama'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgFoglalkoztatottakSzama'][$this->getPrimaryKey()] = true;

$keys = StgFoglalkoztatottakSzamaTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getKistersegNev(),

$keys[1] => $this->getKistersegKshkod(),

$keys[2] => $this->getFoglalkoztatottakSzama(),

$keys[3] => $this->getKistersegiFoglalkoztatasiRata(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgFoglalkoztatottakSzama

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgFoglalkoztatottakSzamaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgFoglalkoztatottakSzama

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setKistersegNev($value);

break;

case 1:

$this->setKistersegKshkod($value);

break;

case 2:

$this->setFoglalkoztatottakSzama($value);

break;

case 3:

$this->setKistersegiFoglalkoztatasiRata($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgFoglalkoztatottakSzamaTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setKistersegNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setKistersegKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setFoglalkoztatottakSzama($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setKistersegiFoglalkoztatasiRata($arr[$keys[3]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgFoglalkoztatottakSzama The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgFoglalkoztatottakSzamaTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEG\_NEV)) {

$criteria->add(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEG\_NEV, $this->kisterseg\_nev);

}

if ($this->isColumnModified(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEG\_KSHKOD)) {

$criteria->add(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEG\_KSHKOD, $this->kisterseg\_kshkod);

}

if ($this->isColumnModified(StgFoglalkoztatottakSzamaTableMap::COL\_FOGLALKOZTATOTTAK\_SZAMA)) {

$criteria->add(StgFoglalkoztatottakSzamaTableMap::COL\_FOGLALKOZTATOTTAK\_SZAMA, $this->foglalkoztatottak\_szama);

}

if ($this->isColumnModified(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEGI\_FOGLALKOZTATASI\_RATA)) {

$criteria->add(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEGI\_FOGLALKOZTATASI\_RATA, $this->kistersegi\_foglalkoztatasi\_rata);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgFoglalkoztatottakSzama object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgFoglalkoztatottakSzama (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setKistersegNev($this->getKistersegNev());

$copyObj->setKistersegKshkod($this->getKistersegKshkod());

$copyObj->setFoglalkoztatottakSzama($this->getFoglalkoztatottakSzama());

$copyObj->setKistersegiFoglalkoztatasiRata($this->getKistersegiFoglalkoztatasiRata());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgFoglalkoztatottakSzama Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->kisterseg\_nev = null;

$this->kisterseg\_kshkod = null;

$this->foglalkoztatottak\_szama = null;

$this->kistersegi\_foglalkoztatasi\_rata = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgFoglalkoztatottakSzamaTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgFoglalkoztatottakSzamaQuery.php

<?php

namespace Base;

use \StgFoglalkoztatottakSzama as ChildStgFoglalkoztatottakSzama;

use \StgFoglalkoztatottakSzamaQuery as ChildStgFoglalkoztatottakSzamaQuery;

use \Exception;

use Map\StgFoglalkoztatottakSzamaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_foglalkoztatottak\_szama' table.

\*

\*

\*

\* @method ChildStgFoglalkoztatottakSzamaQuery orderByKistersegNev($order = Criteria::ASC) Order by the kisterseg\_nev column

\* @method ChildStgFoglalkoztatottakSzamaQuery orderByKistersegKshkod($order = Criteria::ASC) Order by the kisterseg\_KSHKOD column

\* @method ChildStgFoglalkoztatottakSzamaQuery orderByFoglalkoztatottakSzama($order = Criteria::ASC) Order by the foglalkoztatottak\_szama column

\* @method ChildStgFoglalkoztatottakSzamaQuery orderByKistersegiFoglalkoztatasiRata($order = Criteria::ASC) Order by the kistersegi\_foglalkoztatasi\_rata column

\*

\* @method ChildStgFoglalkoztatottakSzamaQuery groupByKistersegNev() Group by the kisterseg\_nev column

\* @method ChildStgFoglalkoztatottakSzamaQuery groupByKistersegKshkod() Group by the kisterseg\_KSHKOD column

\* @method ChildStgFoglalkoztatottakSzamaQuery groupByFoglalkoztatottakSzama() Group by the foglalkoztatottak\_szama column

\* @method ChildStgFoglalkoztatottakSzamaQuery groupByKistersegiFoglalkoztatasiRata() Group by the kistersegi\_foglalkoztatasi\_rata column

\*

\* @method ChildStgFoglalkoztatottakSzamaQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgFoglalkoztatottakSzamaQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgFoglalkoztatottakSzamaQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgFoglalkoztatottakSzama findOne(ConnectionInterface $con = null) Return the first ChildStgFoglalkoztatottakSzama matching the query

\* @method ChildStgFoglalkoztatottakSzama findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgFoglalkoztatottakSzama matching the query, or a new ChildStgFoglalkoztatottakSzama object populated from the query conditions when no match is found

\*

\* @method ChildStgFoglalkoztatottakSzama findOneByKistersegNev(string $kisterseg\_nev) Return the first ChildStgFoglalkoztatottakSzama filtered by the kisterseg\_nev column

\* @method ChildStgFoglalkoztatottakSzama findOneByKistersegKshkod(string $kisterseg\_KSHKOD) Return the first ChildStgFoglalkoztatottakSzama filtered by the kisterseg\_KSHKOD column

\* @method ChildStgFoglalkoztatottakSzama findOneByFoglalkoztatottakSzama(int $foglalkoztatottak\_szama) Return the first ChildStgFoglalkoztatottakSzama filtered by the foglalkoztatottak\_szama column

\* @method ChildStgFoglalkoztatottakSzama findOneByKistersegiFoglalkoztatasiRata(double $kistersegi\_foglalkoztatasi\_rata) Return the first ChildStgFoglalkoztatottakSzama filtered by the kistersegi\_foglalkoztatasi\_rata column

\*

\* @method ChildStgFoglalkoztatottakSzama[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgFoglalkoztatottakSzama objects based on current ModelCriteria

\* @method ChildStgFoglalkoztatottakSzama[]|ObjectCollection findByKistersegNev(string $kisterseg\_nev) Return ChildStgFoglalkoztatottakSzama objects filtered by the kisterseg\_nev column

\* @method ChildStgFoglalkoztatottakSzama[]|ObjectCollection findByKistersegKshkod(string $kisterseg\_KSHKOD) Return ChildStgFoglalkoztatottakSzama objects filtered by the kisterseg\_KSHKOD column

\* @method ChildStgFoglalkoztatottakSzama[]|ObjectCollection findByFoglalkoztatottakSzama(int $foglalkoztatottak\_szama) Return ChildStgFoglalkoztatottakSzama objects filtered by the foglalkoztatottak\_szama column

\* @method ChildStgFoglalkoztatottakSzama[]|ObjectCollection findByKistersegiFoglalkoztatasiRata(double $kistersegi\_foglalkoztatasi\_rata) Return ChildStgFoglalkoztatottakSzama objects filtered by the kistersegi\_foglalkoztatasi\_rata column

\* @method ChildStgFoglalkoztatottakSzama[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgFoglalkoztatottakSzamaQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgFoglalkoztatottakSzamaQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgFoglalkoztatottakSzama', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgFoglalkoztatottakSzamaQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgFoglalkoztatottakSzamaQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgFoglalkoztatottakSzamaQuery) {

return $criteria;

}

$query = new ChildStgFoglalkoztatottakSzamaQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgFoglalkoztatottakSzama|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgFoglalkoztatottakSzama object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgFoglalkoztatottakSzama object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgFoglalkoztatottakSzamaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgFoglalkoztatottakSzama object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgFoglalkoztatottakSzamaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgFoglalkoztatottakSzama object has no primary key');

}

/\*\*

\* Filter the query on the kisterseg\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByKistersegNev('fooValue'); // WHERE kisterseg\_nev = 'fooValue'

\* $query->filterByKistersegNev('%fooValue%'); // WHERE kisterseg\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $kistersegNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgFoglalkoztatottakSzamaQuery The current query, for fluid interface

\*/

public function filterByKistersegNev($kistersegNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($kistersegNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $kistersegNev)) {

$kistersegNev = str\_replace('\*', '%', $kistersegNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEG\_NEV, $kistersegNev, $comparison);

}

/\*\*

\* Filter the query on the kisterseg\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByKistersegKshkod('fooValue'); // WHERE kisterseg\_KSHKOD = 'fooValue'

\* $query->filterByKistersegKshkod('%fooValue%'); // WHERE kisterseg\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $kistersegKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgFoglalkoztatottakSzamaQuery The current query, for fluid interface

\*/

public function filterByKistersegKshkod($kistersegKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($kistersegKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $kistersegKshkod)) {

$kistersegKshkod = str\_replace('\*', '%', $kistersegKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEG\_KSHKOD, $kistersegKshkod, $comparison);

}

/\*\*

\* Filter the query on the foglalkoztatottak\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByFoglalkoztatottakSzama(1234); // WHERE foglalkoztatottak\_szama = 1234

\* $query->filterByFoglalkoztatottakSzama(array(12, 34)); // WHERE foglalkoztatottak\_szama IN (12, 34)

\* $query->filterByFoglalkoztatottakSzama(array('min' => 12)); // WHERE foglalkoztatottak\_szama > 12

\* </code>

\*

\* @param mixed $foglalkoztatottakSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgFoglalkoztatottakSzamaQuery The current query, for fluid interface

\*/

public function filterByFoglalkoztatottakSzama($foglalkoztatottakSzama = null, $comparison = null)

{

if (is\_array($foglalkoztatottakSzama)) {

$useMinMax = false;

if (isset($foglalkoztatottakSzama['min'])) {

$this->addUsingAlias(StgFoglalkoztatottakSzamaTableMap::COL\_FOGLALKOZTATOTTAK\_SZAMA, $foglalkoztatottakSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($foglalkoztatottakSzama['max'])) {

$this->addUsingAlias(StgFoglalkoztatottakSzamaTableMap::COL\_FOGLALKOZTATOTTAK\_SZAMA, $foglalkoztatottakSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgFoglalkoztatottakSzamaTableMap::COL\_FOGLALKOZTATOTTAK\_SZAMA, $foglalkoztatottakSzama, $comparison);

}

/\*\*

\* Filter the query on the kistersegi\_foglalkoztatasi\_rata column

\*

\* Example usage:

\* <code>

\* $query->filterByKistersegiFoglalkoztatasiRata(1234); // WHERE kistersegi\_foglalkoztatasi\_rata = 1234

\* $query->filterByKistersegiFoglalkoztatasiRata(array(12, 34)); // WHERE kistersegi\_foglalkoztatasi\_rata IN (12, 34)

\* $query->filterByKistersegiFoglalkoztatasiRata(array('min' => 12)); // WHERE kistersegi\_foglalkoztatasi\_rata > 12

\* </code>

\*

\* @param mixed $kistersegiFoglalkoztatasiRata The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgFoglalkoztatottakSzamaQuery The current query, for fluid interface

\*/

public function filterByKistersegiFoglalkoztatasiRata($kistersegiFoglalkoztatasiRata = null, $comparison = null)

{

if (is\_array($kistersegiFoglalkoztatasiRata)) {

$useMinMax = false;

if (isset($kistersegiFoglalkoztatasiRata['min'])) {

$this->addUsingAlias(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEGI\_FOGLALKOZTATASI\_RATA, $kistersegiFoglalkoztatasiRata['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($kistersegiFoglalkoztatasiRata['max'])) {

$this->addUsingAlias(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEGI\_FOGLALKOZTATASI\_RATA, $kistersegiFoglalkoztatasiRata['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEGI\_FOGLALKOZTATASI\_RATA, $kistersegiFoglalkoztatasiRata, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgFoglalkoztatottakSzama $stgFoglalkoztatottakSzama Object to remove from the list of results

\*

\* @return $this|ChildStgFoglalkoztatottakSzamaQuery The current query, for fluid interface

\*/

public function prune($stgFoglalkoztatottakSzama = null)

{

if ($stgFoglalkoztatottakSzama) {

throw new LogicException('StgFoglalkoztatottakSzama object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_foglalkoztatottak\_szama table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgFoglalkoztatottakSzamaTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgFoglalkoztatottakSzamaTableMap::clearInstancePool();

StgFoglalkoztatottakSzamaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgFoglalkoztatottakSzamaTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgFoglalkoztatottakSzamaTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgFoglalkoztatottakSzamaTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgFoglalkoztatottakSzamaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgFoglalkoztatottakSzamaQuery

#### StgJovedelemEsAllaskeresokSzama.php

<?php

namespace Base;

use \StgJovedelemEsAllaskeresokSzamaQuery as ChildStgJovedelemEsAllaskeresokSzamaQuery;

use \Exception;

use \PDO;

use Map\StgJovedelemEsAllaskeresokSzamaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgJovedelemEsAllaskeresokSzama implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgJovedelemEsAllaskeresokSzamaTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the osszes\_jovedelem\_foallasbol field.

\* @var int

\*/

protected $osszes\_jovedelem\_foallasbol;

/\*\*

\* The value for the allando\_nepesseg\_szama field.

\* @var int

\*/

protected $allando\_nepesseg\_szama;

/\*\*

\* The value for the p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama field.

\* @var int

\*/

protected $p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgJovedelemEsAllaskeresokSzama object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgJovedelemEsAllaskeresokSzama</code> instance. If

\* <code>obj</code> is an instance of <code>StgJovedelemEsAllaskeresokSzama</code>, delegates to

\* <code>equals(StgJovedelemEsAllaskeresokSzama)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgJovedelemEsAllaskeresokSzama The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [osszes\_jovedelem\_foallasbol] column value.

\*

\* @return int

\*/

public function getOsszesJovedelemFoallasbol()

{

return $this->osszes\_jovedelem\_foallasbol;

}

/\*\*

\* Get the [allando\_nepesseg\_szama] column value.

\*

\* @return int

\*/

public function getAllandoNepessegSzama()

{

return $this->allando\_nepesseg\_szama;

}

/\*\*

\* Get the [p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama] column value.

\*

\* @return int

\*/

public function getP180NaponTuliNyilvantartottAllaskeresokSzama()

{

return $this->p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgJovedelemEsAllaskeresokSzamaTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgJovedelemEsAllaskeresokSzamaTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgJovedelemEsAllaskeresokSzamaTableMap::translateFieldName('OsszesJovedelemFoallasbol', TableMap::TYPE\_PHPNAME, $indexType)];

$this->osszes\_jovedelem\_foallasbol = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgJovedelemEsAllaskeresokSzamaTableMap::translateFieldName('AllandoNepessegSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->allando\_nepesseg\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgJovedelemEsAllaskeresokSzamaTableMap::translateFieldName('P180NaponTuliNyilvantartottAllaskeresokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 5; // 5 = StgJovedelemEsAllaskeresokSzamaTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgJovedelemEsAllaskeresokSzama'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgJovedelemEsAllaskeresokSzama The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgJovedelemEsAllaskeresokSzamaTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgJovedelemEsAllaskeresokSzama The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgJovedelemEsAllaskeresokSzamaTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [osszes\_jovedelem\_foallasbol] column.

\*

\* @param int $v new value

\* @return $this|\StgJovedelemEsAllaskeresokSzama The current object (for fluent API support)

\*/

public function setOsszesJovedelemFoallasbol($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->osszes\_jovedelem\_foallasbol !== $v) {

$this->osszes\_jovedelem\_foallasbol = $v;

$this->modifiedColumns[StgJovedelemEsAllaskeresokSzamaTableMap::COL\_OSSZES\_JOVEDELEM\_FOALLASBOL] = true;

}

return $this;

} // setOsszesJovedelemFoallasbol()

/\*\*

\* Set the value of [allando\_nepesseg\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgJovedelemEsAllaskeresokSzama The current object (for fluent API support)

\*/

public function setAllandoNepessegSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->allando\_nepesseg\_szama !== $v) {

$this->allando\_nepesseg\_szama = $v;

$this->modifiedColumns[StgJovedelemEsAllaskeresokSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA] = true;

}

return $this;

} // setAllandoNepessegSzama()

/\*\*

\* Set the value of [p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgJovedelemEsAllaskeresokSzama The current object (for fluent API support)

\*/

public function setP180NaponTuliNyilvantartottAllaskeresokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama !== $v) {

$this->p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama = $v;

$this->modifiedColumns[StgJovedelemEsAllaskeresokSzamaTableMap::COL\_P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA] = true;

}

return $this;

} // setP180NaponTuliNyilvantartottAllaskeresokSzama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgJovedelemEsAllaskeresokSzamaTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgJovedelemEsAllaskeresokSzamaQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgJovedelemEsAllaskeresokSzama::setDeleted()

\* @see StgJovedelemEsAllaskeresokSzama::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgJovedelemEsAllaskeresokSzamaTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgJovedelemEsAllaskeresokSzamaQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgJovedelemEsAllaskeresokSzamaTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgJovedelemEsAllaskeresokSzamaTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_OSSZES\_JOVEDELEM\_FOALLASBOL)) {

$modifiedColumns[':p' . $index++] = 'OSSZES\_JOVEDELEM\_FOALLASBOL';

}

if ($this->isColumnModified(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'ALLANDO\_NEPESSEG\_SZAMA';

}

if ($this->isColumnModified(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_jovedelem\_es\_allaskeresok\_szama (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'OSSZES\_JOVEDELEM\_FOALLASBOL':

$stmt->bindValue($identifier, $this->osszes\_jovedelem\_foallasbol, PDO::PARAM\_INT);

break;

case 'ALLANDO\_NEPESSEG\_SZAMA':

$stmt->bindValue($identifier, $this->allando\_nepesseg\_szama, PDO::PARAM\_INT);

break;

case 'P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA':

$stmt->bindValue($identifier, $this->p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgJovedelemEsAllaskeresokSzamaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getOsszesJovedelemFoallasbol();

break;

case 3:

return $this->getAllandoNepessegSzama();

break;

case 4:

return $this->getP180NaponTuliNyilvantartottAllaskeresokSzama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgJovedelemEsAllaskeresokSzama'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgJovedelemEsAllaskeresokSzama'][$this->getPrimaryKey()] = true;

$keys = StgJovedelemEsAllaskeresokSzamaTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getOsszesJovedelemFoallasbol(),

$keys[3] => $this->getAllandoNepessegSzama(),

$keys[4] => $this->getP180NaponTuliNyilvantartottAllaskeresokSzama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgJovedelemEsAllaskeresokSzama

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgJovedelemEsAllaskeresokSzamaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgJovedelemEsAllaskeresokSzama

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setOsszesJovedelemFoallasbol($value);

break;

case 3:

$this->setAllandoNepessegSzama($value);

break;

case 4:

$this->setP180NaponTuliNyilvantartottAllaskeresokSzama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgJovedelemEsAllaskeresokSzamaTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setOsszesJovedelemFoallasbol($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setAllandoNepessegSzama($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setP180NaponTuliNyilvantartottAllaskeresokSzama($arr[$keys[4]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgJovedelemEsAllaskeresokSzama The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgJovedelemEsAllaskeresokSzamaTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_OSSZES\_JOVEDELEM\_FOALLASBOL)) {

$criteria->add(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_OSSZES\_JOVEDELEM\_FOALLASBOL, $this->osszes\_jovedelem\_foallasbol);

}

if ($this->isColumnModified(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA)) {

$criteria->add(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA, $this->allando\_nepesseg\_szama);

}

if ($this->isColumnModified(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA)) {

$criteria->add(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA, $this->p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgJovedelemEsAllaskeresokSzama object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgJovedelemEsAllaskeresokSzama (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setOsszesJovedelemFoallasbol($this->getOsszesJovedelemFoallasbol());

$copyObj->setAllandoNepessegSzama($this->getAllandoNepessegSzama());

$copyObj->setP180NaponTuliNyilvantartottAllaskeresokSzama($this->getP180NaponTuliNyilvantartottAllaskeresokSzama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgJovedelemEsAllaskeresokSzama Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->osszes\_jovedelem\_foallasbol = null;

$this->allando\_nepesseg\_szama = null;

$this->p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgJovedelemEsAllaskeresokSzamaTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgJovedelemEsAllaskeresokSzamaQuery.php

<?php

namespace Base;

use \StgJovedelemEsAllaskeresokSzama as ChildStgJovedelemEsAllaskeresokSzama;

use \StgJovedelemEsAllaskeresokSzamaQuery as ChildStgJovedelemEsAllaskeresokSzamaQuery;

use \Exception;

use Map\StgJovedelemEsAllaskeresokSzamaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_jovedelem\_es\_allaskeresok\_szama' table.

\*

\*

\*

\* @method ChildStgJovedelemEsAllaskeresokSzamaQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgJovedelemEsAllaskeresokSzamaQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgJovedelemEsAllaskeresokSzamaQuery orderByOsszesJovedelemFoallasbol($order = Criteria::ASC) Order by the osszes\_jovedelem\_foallasbol column

\* @method ChildStgJovedelemEsAllaskeresokSzamaQuery orderByAllandoNepessegSzama($order = Criteria::ASC) Order by the allando\_nepesseg\_szama column

\* @method ChildStgJovedelemEsAllaskeresokSzamaQuery orderByP180NaponTuliNyilvantartottAllaskeresokSzama($order = Criteria::ASC) Order by the p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama column

\*

\* @method ChildStgJovedelemEsAllaskeresokSzamaQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgJovedelemEsAllaskeresokSzamaQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgJovedelemEsAllaskeresokSzamaQuery groupByOsszesJovedelemFoallasbol() Group by the osszes\_jovedelem\_foallasbol column

\* @method ChildStgJovedelemEsAllaskeresokSzamaQuery groupByAllandoNepessegSzama() Group by the allando\_nepesseg\_szama column

\* @method ChildStgJovedelemEsAllaskeresokSzamaQuery groupByP180NaponTuliNyilvantartottAllaskeresokSzama() Group by the p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama column

\*

\* @method ChildStgJovedelemEsAllaskeresokSzamaQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgJovedelemEsAllaskeresokSzamaQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgJovedelemEsAllaskeresokSzamaQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgJovedelemEsAllaskeresokSzama findOne(ConnectionInterface $con = null) Return the first ChildStgJovedelemEsAllaskeresokSzama matching the query

\* @method ChildStgJovedelemEsAllaskeresokSzama findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgJovedelemEsAllaskeresokSzama matching the query, or a new ChildStgJovedelemEsAllaskeresokSzama object populated from the query conditions when no match is found

\*

\* @method ChildStgJovedelemEsAllaskeresokSzama findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgJovedelemEsAllaskeresokSzama filtered by the telepules\_nev column

\* @method ChildStgJovedelemEsAllaskeresokSzama findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgJovedelemEsAllaskeresokSzama filtered by the telepules\_KSHKOD column

\* @method ChildStgJovedelemEsAllaskeresokSzama findOneByOsszesJovedelemFoallasbol(int $osszes\_jovedelem\_foallasbol) Return the first ChildStgJovedelemEsAllaskeresokSzama filtered by the osszes\_jovedelem\_foallasbol column

\* @method ChildStgJovedelemEsAllaskeresokSzama findOneByAllandoNepessegSzama(int $allando\_nepesseg\_szama) Return the first ChildStgJovedelemEsAllaskeresokSzama filtered by the allando\_nepesseg\_szama column

\* @method ChildStgJovedelemEsAllaskeresokSzama findOneByP180NaponTuliNyilvantartottAllaskeresokSzama(int $p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama) Return the first ChildStgJovedelemEsAllaskeresokSzama filtered by the p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama column

\*

\* @method ChildStgJovedelemEsAllaskeresokSzama[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgJovedelemEsAllaskeresokSzama objects based on current ModelCriteria

\* @method ChildStgJovedelemEsAllaskeresokSzama[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgJovedelemEsAllaskeresokSzama objects filtered by the telepules\_nev column

\* @method ChildStgJovedelemEsAllaskeresokSzama[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgJovedelemEsAllaskeresokSzama objects filtered by the telepules\_KSHKOD column

\* @method ChildStgJovedelemEsAllaskeresokSzama[]|ObjectCollection findByOsszesJovedelemFoallasbol(int $osszes\_jovedelem\_foallasbol) Return ChildStgJovedelemEsAllaskeresokSzama objects filtered by the osszes\_jovedelem\_foallasbol column

\* @method ChildStgJovedelemEsAllaskeresokSzama[]|ObjectCollection findByAllandoNepessegSzama(int $allando\_nepesseg\_szama) Return ChildStgJovedelemEsAllaskeresokSzama objects filtered by the allando\_nepesseg\_szama column

\* @method ChildStgJovedelemEsAllaskeresokSzama[]|ObjectCollection findByP180NaponTuliNyilvantartottAllaskeresokSzama(int $p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama) Return ChildStgJovedelemEsAllaskeresokSzama objects filtered by the p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama column

\* @method ChildStgJovedelemEsAllaskeresokSzama[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgJovedelemEsAllaskeresokSzamaQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgJovedelemEsAllaskeresokSzamaQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgJovedelemEsAllaskeresokSzama', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgJovedelemEsAllaskeresokSzamaQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgJovedelemEsAllaskeresokSzamaQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgJovedelemEsAllaskeresokSzamaQuery) {

return $criteria;

}

$query = new ChildStgJovedelemEsAllaskeresokSzamaQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgJovedelemEsAllaskeresokSzama|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgJovedelemEsAllaskeresokSzama object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgJovedelemEsAllaskeresokSzama object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgJovedelemEsAllaskeresokSzamaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgJovedelemEsAllaskeresokSzama object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgJovedelemEsAllaskeresokSzamaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgJovedelemEsAllaskeresokSzama object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgJovedelemEsAllaskeresokSzamaQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgJovedelemEsAllaskeresokSzamaQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the osszes\_jovedelem\_foallasbol column

\*

\* Example usage:

\* <code>

\* $query->filterByOsszesJovedelemFoallasbol(1234); // WHERE osszes\_jovedelem\_foallasbol = 1234

\* $query->filterByOsszesJovedelemFoallasbol(array(12, 34)); // WHERE osszes\_jovedelem\_foallasbol IN (12, 34)

\* $query->filterByOsszesJovedelemFoallasbol(array('min' => 12)); // WHERE osszes\_jovedelem\_foallasbol > 12

\* </code>

\*

\* @param mixed $osszesJovedelemFoallasbol The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgJovedelemEsAllaskeresokSzamaQuery The current query, for fluid interface

\*/

public function filterByOsszesJovedelemFoallasbol($osszesJovedelemFoallasbol = null, $comparison = null)

{

if (is\_array($osszesJovedelemFoallasbol)) {

$useMinMax = false;

if (isset($osszesJovedelemFoallasbol['min'])) {

$this->addUsingAlias(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_OSSZES\_JOVEDELEM\_FOALLASBOL, $osszesJovedelemFoallasbol['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($osszesJovedelemFoallasbol['max'])) {

$this->addUsingAlias(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_OSSZES\_JOVEDELEM\_FOALLASBOL, $osszesJovedelemFoallasbol['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_OSSZES\_JOVEDELEM\_FOALLASBOL, $osszesJovedelemFoallasbol, $comparison);

}

/\*\*

\* Filter the query on the allando\_nepesseg\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByAllandoNepessegSzama(1234); // WHERE allando\_nepesseg\_szama = 1234

\* $query->filterByAllandoNepessegSzama(array(12, 34)); // WHERE allando\_nepesseg\_szama IN (12, 34)

\* $query->filterByAllandoNepessegSzama(array('min' => 12)); // WHERE allando\_nepesseg\_szama > 12

\* </code>

\*

\* @param mixed $allandoNepessegSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgJovedelemEsAllaskeresokSzamaQuery The current query, for fluid interface

\*/

public function filterByAllandoNepessegSzama($allandoNepessegSzama = null, $comparison = null)

{

if (is\_array($allandoNepessegSzama)) {

$useMinMax = false;

if (isset($allandoNepessegSzama['min'])) {

$this->addUsingAlias(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA, $allandoNepessegSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($allandoNepessegSzama['max'])) {

$this->addUsingAlias(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA, $allandoNepessegSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA, $allandoNepessegSzama, $comparison);

}

/\*\*

\* Filter the query on the p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByP180NaponTuliNyilvantartottAllaskeresokSzama(1234); // WHERE p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama = 1234

\* $query->filterByP180NaponTuliNyilvantartottAllaskeresokSzama(array(12, 34)); // WHERE p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama IN (12, 34)

\* $query->filterByP180NaponTuliNyilvantartottAllaskeresokSzama(array('min' => 12)); // WHERE p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama > 12

\* </code>

\*

\* @param mixed $p180NaponTuliNyilvantartottAllaskeresokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgJovedelemEsAllaskeresokSzamaQuery The current query, for fluid interface

\*/

public function filterByP180NaponTuliNyilvantartottAllaskeresokSzama($p180NaponTuliNyilvantartottAllaskeresokSzama = null, $comparison = null)

{

if (is\_array($p180NaponTuliNyilvantartottAllaskeresokSzama)) {

$useMinMax = false;

if (isset($p180NaponTuliNyilvantartottAllaskeresokSzama['min'])) {

$this->addUsingAlias(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA, $p180NaponTuliNyilvantartottAllaskeresokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($p180NaponTuliNyilvantartottAllaskeresokSzama['max'])) {

$this->addUsingAlias(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA, $p180NaponTuliNyilvantartottAllaskeresokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA, $p180NaponTuliNyilvantartottAllaskeresokSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgJovedelemEsAllaskeresokSzama $stgJovedelemEsAllaskeresokSzama Object to remove from the list of results

\*

\* @return $this|ChildStgJovedelemEsAllaskeresokSzamaQuery The current query, for fluid interface

\*/

public function prune($stgJovedelemEsAllaskeresokSzama = null)

{

if ($stgJovedelemEsAllaskeresokSzama) {

throw new LogicException('StgJovedelemEsAllaskeresokSzama object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_jovedelem\_es\_allaskeresok\_szama table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgJovedelemEsAllaskeresokSzamaTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgJovedelemEsAllaskeresokSzamaTableMap::clearInstancePool();

StgJovedelemEsAllaskeresokSzamaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgJovedelemEsAllaskeresokSzamaTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgJovedelemEsAllaskeresokSzamaTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgJovedelemEsAllaskeresokSzamaTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgJovedelemEsAllaskeresokSzamaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgJovedelemEsAllaskeresokSzamaQuery

#### StgJovedelmiHelyzet.php

<?php

namespace Base;

use \StgJovedelmiHelyzetQuery as ChildStgJovedelmiHelyzetQuery;

use \Exception;

use \PDO;

use Map\StgJovedelmiHelyzetTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgJovedelmiHelyzet implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgJovedelmiHelyzetTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the osszes\_ado field.

\* @var int

\*/

protected $osszes\_ado;

/\*\*

\* The value for the adofizetok\_szama field.

\* @var int

\*/

protected $adofizetok\_szama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgJovedelmiHelyzet object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgJovedelmiHelyzet</code> instance. If

\* <code>obj</code> is an instance of <code>StgJovedelmiHelyzet</code>, delegates to

\* <code>equals(StgJovedelmiHelyzet)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgJovedelmiHelyzet The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [osszes\_ado] column value.

\*

\* @return int

\*/

public function getOsszesAdo()

{

return $this->osszes\_ado;

}

/\*\*

\* Get the [adofizetok\_szama] column value.

\*

\* @return int

\*/

public function getAdofizetokSzama()

{

return $this->adofizetok\_szama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgJovedelmiHelyzetTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgJovedelmiHelyzetTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgJovedelmiHelyzetTableMap::translateFieldName('OsszesAdo', TableMap::TYPE\_PHPNAME, $indexType)];

$this->osszes\_ado = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgJovedelmiHelyzetTableMap::translateFieldName('AdofizetokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->adofizetok\_szama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 4; // 4 = StgJovedelmiHelyzetTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgJovedelmiHelyzet'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgJovedelmiHelyzet The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgJovedelmiHelyzetTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgJovedelmiHelyzet The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgJovedelmiHelyzetTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [osszes\_ado] column.

\*

\* @param int $v new value

\* @return $this|\StgJovedelmiHelyzet The current object (for fluent API support)

\*/

public function setOsszesAdo($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->osszes\_ado !== $v) {

$this->osszes\_ado = $v;

$this->modifiedColumns[StgJovedelmiHelyzetTableMap::COL\_OSSZES\_ADO] = true;

}

return $this;

} // setOsszesAdo()

/\*\*

\* Set the value of [adofizetok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgJovedelmiHelyzet The current object (for fluent API support)

\*/

public function setAdofizetokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->adofizetok\_szama !== $v) {

$this->adofizetok\_szama = $v;

$this->modifiedColumns[StgJovedelmiHelyzetTableMap::COL\_ADOFIZETOK\_SZAMA] = true;

}

return $this;

} // setAdofizetokSzama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgJovedelmiHelyzetTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgJovedelmiHelyzetQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgJovedelmiHelyzet::setDeleted()

\* @see StgJovedelmiHelyzet::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgJovedelmiHelyzetTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgJovedelmiHelyzetQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgJovedelmiHelyzetTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgJovedelmiHelyzetTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgJovedelmiHelyzetTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgJovedelmiHelyzetTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgJovedelmiHelyzetTableMap::COL\_OSSZES\_ADO)) {

$modifiedColumns[':p' . $index++] = 'OSSZES\_ADO';

}

if ($this->isColumnModified(StgJovedelmiHelyzetTableMap::COL\_ADOFIZETOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'ADOFIZETOK\_SZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_jovedelmi\_helyzet (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'OSSZES\_ADO':

$stmt->bindValue($identifier, $this->osszes\_ado, PDO::PARAM\_INT);

break;

case 'ADOFIZETOK\_SZAMA':

$stmt->bindValue($identifier, $this->adofizetok\_szama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgJovedelmiHelyzetTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getOsszesAdo();

break;

case 3:

return $this->getAdofizetokSzama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgJovedelmiHelyzet'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgJovedelmiHelyzet'][$this->getPrimaryKey()] = true;

$keys = StgJovedelmiHelyzetTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getOsszesAdo(),

$keys[3] => $this->getAdofizetokSzama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgJovedelmiHelyzet

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgJovedelmiHelyzetTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgJovedelmiHelyzet

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setOsszesAdo($value);

break;

case 3:

$this->setAdofizetokSzama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgJovedelmiHelyzetTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setOsszesAdo($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setAdofizetokSzama($arr[$keys[3]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgJovedelmiHelyzet The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgJovedelmiHelyzetTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgJovedelmiHelyzetTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgJovedelmiHelyzetTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgJovedelmiHelyzetTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgJovedelmiHelyzetTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgJovedelmiHelyzetTableMap::COL\_OSSZES\_ADO)) {

$criteria->add(StgJovedelmiHelyzetTableMap::COL\_OSSZES\_ADO, $this->osszes\_ado);

}

if ($this->isColumnModified(StgJovedelmiHelyzetTableMap::COL\_ADOFIZETOK\_SZAMA)) {

$criteria->add(StgJovedelmiHelyzetTableMap::COL\_ADOFIZETOK\_SZAMA, $this->adofizetok\_szama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgJovedelmiHelyzet object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgJovedelmiHelyzet (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setOsszesAdo($this->getOsszesAdo());

$copyObj->setAdofizetokSzama($this->getAdofizetokSzama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgJovedelmiHelyzet Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->osszes\_ado = null;

$this->adofizetok\_szama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgJovedelmiHelyzetTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgJovedelmiHelyzetQuery.php

<?php

namespace Base;

use \StgJovedelmiHelyzet as ChildStgJovedelmiHelyzet;

use \StgJovedelmiHelyzetQuery as ChildStgJovedelmiHelyzetQuery;

use \Exception;

use Map\StgJovedelmiHelyzetTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_jovedelmi\_helyzet' table.

\*

\*

\*

\* @method ChildStgJovedelmiHelyzetQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgJovedelmiHelyzetQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgJovedelmiHelyzetQuery orderByOsszesAdo($order = Criteria::ASC) Order by the osszes\_ado column

\* @method ChildStgJovedelmiHelyzetQuery orderByAdofizetokSzama($order = Criteria::ASC) Order by the adofizetok\_szama column

\*

\* @method ChildStgJovedelmiHelyzetQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgJovedelmiHelyzetQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgJovedelmiHelyzetQuery groupByOsszesAdo() Group by the osszes\_ado column

\* @method ChildStgJovedelmiHelyzetQuery groupByAdofizetokSzama() Group by the adofizetok\_szama column

\*

\* @method ChildStgJovedelmiHelyzetQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgJovedelmiHelyzetQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgJovedelmiHelyzetQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgJovedelmiHelyzet findOne(ConnectionInterface $con = null) Return the first ChildStgJovedelmiHelyzet matching the query

\* @method ChildStgJovedelmiHelyzet findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgJovedelmiHelyzet matching the query, or a new ChildStgJovedelmiHelyzet object populated from the query conditions when no match is found

\*

\* @method ChildStgJovedelmiHelyzet findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgJovedelmiHelyzet filtered by the telepules\_nev column

\* @method ChildStgJovedelmiHelyzet findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgJovedelmiHelyzet filtered by the telepules\_KSHKOD column

\* @method ChildStgJovedelmiHelyzet findOneByOsszesAdo(int $osszes\_ado) Return the first ChildStgJovedelmiHelyzet filtered by the osszes\_ado column

\* @method ChildStgJovedelmiHelyzet findOneByAdofizetokSzama(int $adofizetok\_szama) Return the first ChildStgJovedelmiHelyzet filtered by the adofizetok\_szama column

\*

\* @method ChildStgJovedelmiHelyzet[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgJovedelmiHelyzet objects based on current ModelCriteria

\* @method ChildStgJovedelmiHelyzet[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgJovedelmiHelyzet objects filtered by the telepules\_nev column

\* @method ChildStgJovedelmiHelyzet[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgJovedelmiHelyzet objects filtered by the telepules\_KSHKOD column

\* @method ChildStgJovedelmiHelyzet[]|ObjectCollection findByOsszesAdo(int $osszes\_ado) Return ChildStgJovedelmiHelyzet objects filtered by the osszes\_ado column

\* @method ChildStgJovedelmiHelyzet[]|ObjectCollection findByAdofizetokSzama(int $adofizetok\_szama) Return ChildStgJovedelmiHelyzet objects filtered by the adofizetok\_szama column

\* @method ChildStgJovedelmiHelyzet[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgJovedelmiHelyzetQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgJovedelmiHelyzetQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgJovedelmiHelyzet', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgJovedelmiHelyzetQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgJovedelmiHelyzetQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgJovedelmiHelyzetQuery) {

return $criteria;

}

$query = new ChildStgJovedelmiHelyzetQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgJovedelmiHelyzet|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgJovedelmiHelyzet object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgJovedelmiHelyzet object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgJovedelmiHelyzetQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgJovedelmiHelyzet object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgJovedelmiHelyzetQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgJovedelmiHelyzet object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgJovedelmiHelyzetQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgJovedelmiHelyzetTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgJovedelmiHelyzetQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgJovedelmiHelyzetTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the osszes\_ado column

\*

\* Example usage:

\* <code>

\* $query->filterByOsszesAdo(1234); // WHERE osszes\_ado = 1234

\* $query->filterByOsszesAdo(array(12, 34)); // WHERE osszes\_ado IN (12, 34)

\* $query->filterByOsszesAdo(array('min' => 12)); // WHERE osszes\_ado > 12

\* </code>

\*

\* @param mixed $osszesAdo The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgJovedelmiHelyzetQuery The current query, for fluid interface

\*/

public function filterByOsszesAdo($osszesAdo = null, $comparison = null)

{

if (is\_array($osszesAdo)) {

$useMinMax = false;

if (isset($osszesAdo['min'])) {

$this->addUsingAlias(StgJovedelmiHelyzetTableMap::COL\_OSSZES\_ADO, $osszesAdo['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($osszesAdo['max'])) {

$this->addUsingAlias(StgJovedelmiHelyzetTableMap::COL\_OSSZES\_ADO, $osszesAdo['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgJovedelmiHelyzetTableMap::COL\_OSSZES\_ADO, $osszesAdo, $comparison);

}

/\*\*

\* Filter the query on the adofizetok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByAdofizetokSzama(1234); // WHERE adofizetok\_szama = 1234

\* $query->filterByAdofizetokSzama(array(12, 34)); // WHERE adofizetok\_szama IN (12, 34)

\* $query->filterByAdofizetokSzama(array('min' => 12)); // WHERE adofizetok\_szama > 12

\* </code>

\*

\* @param mixed $adofizetokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgJovedelmiHelyzetQuery The current query, for fluid interface

\*/

public function filterByAdofizetokSzama($adofizetokSzama = null, $comparison = null)

{

if (is\_array($adofizetokSzama)) {

$useMinMax = false;

if (isset($adofizetokSzama['min'])) {

$this->addUsingAlias(StgJovedelmiHelyzetTableMap::COL\_ADOFIZETOK\_SZAMA, $adofizetokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($adofizetokSzama['max'])) {

$this->addUsingAlias(StgJovedelmiHelyzetTableMap::COL\_ADOFIZETOK\_SZAMA, $adofizetokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgJovedelmiHelyzetTableMap::COL\_ADOFIZETOK\_SZAMA, $adofizetokSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgJovedelmiHelyzet $stgJovedelmiHelyzet Object to remove from the list of results

\*

\* @return $this|ChildStgJovedelmiHelyzetQuery The current query, for fluid interface

\*/

public function prune($stgJovedelmiHelyzet = null)

{

if ($stgJovedelmiHelyzet) {

throw new LogicException('StgJovedelmiHelyzet object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_jovedelmi\_helyzet table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgJovedelmiHelyzetTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgJovedelmiHelyzetTableMap::clearInstancePool();

StgJovedelmiHelyzetTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgJovedelmiHelyzetTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgJovedelmiHelyzetTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgJovedelmiHelyzetTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgJovedelmiHelyzetTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgJovedelmiHelyzetQuery

#### StgKeresztmetszetiForgalom.php

<?php

namespace Base;

use \StgKeresztmetszetiForgalomQuery as ChildStgKeresztmetszetiForgalomQuery;

use \Exception;

use \PDO;

use Map\StgKeresztmetszetiForgalomTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgKeresztmetszetiForgalom implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgKeresztmetszetiForgalomTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the shapenr field.

\* @var int

\*/

protected $shapenr;

/\*\*

\* The value for the partnr field.

\* @var int

\*/

protected $partnr;

/\*\*

\* The value for the nrparts field.

\* @var int

\*/

protected $nrparts;

/\*\*

\* The value for the pointnr field.

\* @var int

\*/

protected $pointnr;

/\*\*

\* The value for the nrpoints field.

\* @var int

\*/

protected $nrpoints;

/\*\*

\* The value for the x field.

\* @var double

\*/

protected $x;

/\*\*

\* The value for the y field.

\* @var double

\*/

protected $y;

/\*\*

\* The value for the measure field.

\* @var double

\*/

protected $measure;

/\*\*

\* The value for the kszam field.

\* @var string

\*/

protected $kszam;

/\*\*

\* The value for the pkod field.

\* @var double

\*/

protected $pkod;

/\*\*

\* The value for the kkod field.

\* @var string

\*/

protected $kkod;

/\*\*

\* The value for the vvkod field.

\* @var string

\*/

protected $vvkod;

/\*\*

\* The value for the kszelv field.

\* @var string

\*/

protected $kszelv;

/\*\*

\* The value for the vszelv field.

\* @var string

\*/

protected $vszelv;

/\*\*

\* The value for the rshossz field.

\* @var double

\*/

protected $rshossz;

/\*\*

\* The value for the anf field.

\* @var int

\*/

protected $anf;

/\*\*

\* The value for the anet field.

\* @var int

\*/

protected $anet;

/\*\*

\* The value for the mof field.

\* @var int

\*/

protected $mof;

/\*\*

\* The value for the ongj field.

\* @var int

\*/

protected $ongj;

/\*\*

\* The value for the oj field.

\* @var int

\*/

protected $oj;

/\*\*

\* The value for the omot field.

\* @var int

\*/

protected $omot;

/\*\*

\* The value for the ev field.

\* @var int

\*/

protected $ev;

/\*\*

\* The value for the asz field.

\* @var int

\*/

protected $asz;

/\*\*

\* The value for the buszcs field.

\* @var int

\*/

protected $buszcs;

/\*\*

\* The value for the busze field.

\* @var int

\*/

protected $busze;

/\*\*

\* The value for the obusz field.

\* @var string

\*/

protected $obusz;

/\*\*

\* The value for the nyszer field.

\* @var string

\*/

protected $nyszer;

/\*\*

\* The value for the potktgk field.

\* @var string

\*/

protected $potktgk;

/\*\*

\* The value for the ktgk field.

\* @var string

\*/

protected $ktgk;

/\*\*

\* The value for the ntgk field.

\* @var string

\*/

protected $ntgk;

/\*\*

\* The value for the kntgk field.

\* @var string

\*/

protected $kntgk;

/\*\*

\* The value for the otgk field.

\* @var string

\*/

protected $otgk;

/\*\*

\* The value for the szgk field.

\* @var string

\*/

protected $szgk;

/\*\*

\* The value for the oszgk field.

\* @var string

\*/

protected $oszgk;

/\*\*

\* The value for the mkp field.

\* @var string

\*/

protected $mkp;

/\*\*

\* The value for the kpf field.

\* @var string

\*/

protected $kpf;

/\*\*

\* The value for the lassu field.

\* @var string

\*/

protected $lassu;

/\*\*

\* The value for the spec field.

\* @var string

\*/

protected $spec;

/\*\*

\* The value for the fmegb field.

\* @var string

\*/

protected $fmegb;

/\*\*

\* The value for the adatforr field.

\* @var string

\*/

protected $adatforr;

/\*\*

\* The value for the szamlnap field.

\* @var string

\*/

protected $szamlnap;

/\*\*

\* The value for the jelleg1 field.

\* @var string

\*/

protected $jelleg1;

/\*\*

\* The value for the jelleg2 field.

\* @var string

\*/

protected $jelleg2;

/\*\*

\* The value for the fmegj field.

\* @var string

\*/

protected $fmegj;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgKeresztmetszetiForgalom object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgKeresztmetszetiForgalom</code> instance. If

\* <code>obj</code> is an instance of <code>StgKeresztmetszetiForgalom</code>, delegates to

\* <code>equals(StgKeresztmetszetiForgalom)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgKeresztmetszetiForgalom The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [shapenr] column value.

\*

\* @return int

\*/

public function getShapenr()

{

return $this->shapenr;

}

/\*\*

\* Get the [partnr] column value.

\*

\* @return int

\*/

public function getPartnr()

{

return $this->partnr;

}

/\*\*

\* Get the [nrparts] column value.

\*

\* @return int

\*/

public function getNrparts()

{

return $this->nrparts;

}

/\*\*

\* Get the [pointnr] column value.

\*

\* @return int

\*/

public function getPointnr()

{

return $this->pointnr;

}

/\*\*

\* Get the [nrpoints] column value.

\*

\* @return int

\*/

public function getNrpoints()

{

return $this->nrpoints;

}

/\*\*

\* Get the [x] column value.

\*

\* @return double

\*/

public function getX()

{

return $this->x;

}

/\*\*

\* Get the [y] column value.

\*

\* @return double

\*/

public function getY()

{

return $this->y;

}

/\*\*

\* Get the [measure] column value.

\*

\* @return double

\*/

public function getMeasure()

{

return $this->measure;

}

/\*\*

\* Get the [kszam] column value.

\*

\* @return string

\*/

public function getKszam()

{

return $this->kszam;

}

/\*\*

\* Get the [pkod] column value.

\*

\* @return double

\*/

public function getPkod()

{

return $this->pkod;

}

/\*\*

\* Get the [kkod] column value.

\*

\* @return string

\*/

public function getKkod()

{

return $this->kkod;

}

/\*\*

\* Get the [vvkod] column value.

\*

\* @return string

\*/

public function getVvkod()

{

return $this->vvkod;

}

/\*\*

\* Get the [kszelv] column value.

\*

\* @return string

\*/

public function getKszelv()

{

return $this->kszelv;

}

/\*\*

\* Get the [vszelv] column value.

\*

\* @return string

\*/

public function getVszelv()

{

return $this->vszelv;

}

/\*\*

\* Get the [rshossz] column value.

\*

\* @return double

\*/

public function getRshossz()

{

return $this->rshossz;

}

/\*\*

\* Get the [anf] column value.

\*

\* @return int

\*/

public function getAnf()

{

return $this->anf;

}

/\*\*

\* Get the [anet] column value.

\*

\* @return int

\*/

public function getAnet()

{

return $this->anet;

}

/\*\*

\* Get the [mof] column value.

\*

\* @return int

\*/

public function getMof()

{

return $this->mof;

}

/\*\*

\* Get the [ongj] column value.

\*

\* @return int

\*/

public function getOngj()

{

return $this->ongj;

}

/\*\*

\* Get the [oj] column value.

\*

\* @return int

\*/

public function getOj()

{

return $this->oj;

}

/\*\*

\* Get the [omot] column value.

\*

\* @return int

\*/

public function getOmot()

{

return $this->omot;

}

/\*\*

\* Get the [ev] column value.

\*

\* @return int

\*/

public function getEv()

{

return $this->ev;

}

/\*\*

\* Get the [asz] column value.

\*

\* @return int

\*/

public function getAsz()

{

return $this->asz;

}

/\*\*

\* Get the [buszcs] column value.

\*

\* @return int

\*/

public function getBuszcs()

{

return $this->buszcs;

}

/\*\*

\* Get the [busze] column value.

\*

\* @return int

\*/

public function getBusze()

{

return $this->busze;

}

/\*\*

\* Get the [obusz] column value.

\*

\* @return string

\*/

public function getObusz()

{

return $this->obusz;

}

/\*\*

\* Get the [nyszer] column value.

\*

\* @return string

\*/

public function getNyszer()

{

return $this->nyszer;

}

/\*\*

\* Get the [potktgk] column value.

\*

\* @return string

\*/

public function getPotktgk()

{

return $this->potktgk;

}

/\*\*

\* Get the [ktgk] column value.

\*

\* @return string

\*/

public function getKtgk()

{

return $this->ktgk;

}

/\*\*

\* Get the [ntgk] column value.

\*

\* @return string

\*/

public function getNtgk()

{

return $this->ntgk;

}

/\*\*

\* Get the [kntgk] column value.

\*

\* @return string

\*/

public function getKntgk()

{

return $this->kntgk;

}

/\*\*

\* Get the [otgk] column value.

\*

\* @return string

\*/

public function getOtgk()

{

return $this->otgk;

}

/\*\*

\* Get the [szgk] column value.

\*

\* @return string

\*/

public function getSzgk()

{

return $this->szgk;

}

/\*\*

\* Get the [oszgk] column value.

\*

\* @return string

\*/

public function getOszgk()

{

return $this->oszgk;

}

/\*\*

\* Get the [mkp] column value.

\*

\* @return string

\*/

public function getMkp()

{

return $this->mkp;

}

/\*\*

\* Get the [kpf] column value.

\*

\* @return string

\*/

public function getKpf()

{

return $this->kpf;

}

/\*\*

\* Get the [lassu] column value.

\*

\* @return string

\*/

public function getLassu()

{

return $this->lassu;

}

/\*\*

\* Get the [spec] column value.

\*

\* @return string

\*/

public function getSpec()

{

return $this->spec;

}

/\*\*

\* Get the [fmegb] column value.

\*

\* @return string

\*/

public function getFmegb()

{

return $this->fmegb;

}

/\*\*

\* Get the [adatforr] column value.

\*

\* @return string

\*/

public function getAdatforr()

{

return $this->adatforr;

}

/\*\*

\* Get the [szamlnap] column value.

\*

\* @return string

\*/

public function getSzamlnap()

{

return $this->szamlnap;

}

/\*\*

\* Get the [jelleg1] column value.

\*

\* @return string

\*/

public function getJelleg1()

{

return $this->jelleg1;

}

/\*\*

\* Get the [jelleg2] column value.

\*

\* @return string

\*/

public function getJelleg2()

{

return $this->jelleg2;

}

/\*\*

\* Get the [fmegj] column value.

\*

\* @return string

\*/

public function getFmegj()

{

return $this->fmegj;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Shapenr', TableMap::TYPE\_PHPNAME, $indexType)];

$this->shapenr = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Partnr', TableMap::TYPE\_PHPNAME, $indexType)];

$this->partnr = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Nrparts', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nrparts = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Pointnr', TableMap::TYPE\_PHPNAME, $indexType)];

$this->pointnr = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Nrpoints', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nrpoints = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 5 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('X', TableMap::TYPE\_PHPNAME, $indexType)];

$this->x = (null !== $col) ? (double) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 6 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Y', TableMap::TYPE\_PHPNAME, $indexType)];

$this->y = (null !== $col) ? (double) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 7 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Measure', TableMap::TYPE\_PHPNAME, $indexType)];

$this->measure = (null !== $col) ? (double) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 8 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Kszam', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kszam = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 9 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Pkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->pkod = (null !== $col) ? (double) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 10 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Kkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 11 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Vvkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->vvkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 12 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Kszelv', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kszelv = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 13 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Vszelv', TableMap::TYPE\_PHPNAME, $indexType)];

$this->vszelv = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 14 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Rshossz', TableMap::TYPE\_PHPNAME, $indexType)];

$this->rshossz = (null !== $col) ? (double) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 15 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Anf', TableMap::TYPE\_PHPNAME, $indexType)];

$this->anf = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 16 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Anet', TableMap::TYPE\_PHPNAME, $indexType)];

$this->anet = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 17 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Mof', TableMap::TYPE\_PHPNAME, $indexType)];

$this->mof = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 18 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Ongj', TableMap::TYPE\_PHPNAME, $indexType)];

$this->ongj = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 19 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Oj', TableMap::TYPE\_PHPNAME, $indexType)];

$this->oj = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 20 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Omot', TableMap::TYPE\_PHPNAME, $indexType)];

$this->omot = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 21 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Ev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->ev = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 22 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Asz', TableMap::TYPE\_PHPNAME, $indexType)];

$this->asz = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 23 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Buszcs', TableMap::TYPE\_PHPNAME, $indexType)];

$this->buszcs = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 24 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Busze', TableMap::TYPE\_PHPNAME, $indexType)];

$this->busze = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 25 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Obusz', TableMap::TYPE\_PHPNAME, $indexType)];

$this->obusz = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 26 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Nyszer', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nyszer = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 27 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Potktgk', TableMap::TYPE\_PHPNAME, $indexType)];

$this->potktgk = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 28 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Ktgk', TableMap::TYPE\_PHPNAME, $indexType)];

$this->ktgk = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 29 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Ntgk', TableMap::TYPE\_PHPNAME, $indexType)];

$this->ntgk = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 30 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Kntgk', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kntgk = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 31 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Otgk', TableMap::TYPE\_PHPNAME, $indexType)];

$this->otgk = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 32 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Szgk', TableMap::TYPE\_PHPNAME, $indexType)];

$this->szgk = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 33 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Oszgk', TableMap::TYPE\_PHPNAME, $indexType)];

$this->oszgk = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 34 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Mkp', TableMap::TYPE\_PHPNAME, $indexType)];

$this->mkp = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 35 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Kpf', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kpf = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 36 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Lassu', TableMap::TYPE\_PHPNAME, $indexType)];

$this->lassu = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 37 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Spec', TableMap::TYPE\_PHPNAME, $indexType)];

$this->spec = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 38 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Fmegb', TableMap::TYPE\_PHPNAME, $indexType)];

$this->fmegb = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 39 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Adatforr', TableMap::TYPE\_PHPNAME, $indexType)];

$this->adatforr = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 40 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Szamlnap', TableMap::TYPE\_PHPNAME, $indexType)];

$this->szamlnap = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 41 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Jelleg1', TableMap::TYPE\_PHPNAME, $indexType)];

$this->jelleg1 = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 42 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Jelleg2', TableMap::TYPE\_PHPNAME, $indexType)];

$this->jelleg2 = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 43 + $startcol : StgKeresztmetszetiForgalomTableMap::translateFieldName('Fmegj', TableMap::TYPE\_PHPNAME, $indexType)];

$this->fmegj = (null !== $col) ? (string) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 44; // 44 = StgKeresztmetszetiForgalomTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgKeresztmetszetiForgalom'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [shapenr] column.

\*

\* @param int $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setShapenr($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->shapenr !== $v) {

$this->shapenr = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_SHAPENR] = true;

}

return $this;

} // setShapenr()

/\*\*

\* Set the value of [partnr] column.

\*

\* @param int $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setPartnr($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->partnr !== $v) {

$this->partnr = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_PARTNR] = true;

}

return $this;

} // setPartnr()

/\*\*

\* Set the value of [nrparts] column.

\*

\* @param int $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setNrparts($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nrparts !== $v) {

$this->nrparts = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_NRPARTS] = true;

}

return $this;

} // setNrparts()

/\*\*

\* Set the value of [pointnr] column.

\*

\* @param int $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setPointnr($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->pointnr !== $v) {

$this->pointnr = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_POINTNR] = true;

}

return $this;

} // setPointnr()

/\*\*

\* Set the value of [nrpoints] column.

\*

\* @param int $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setNrpoints($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nrpoints !== $v) {

$this->nrpoints = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_NRPOINTS] = true;

}

return $this;

} // setNrpoints()

/\*\*

\* Set the value of [x] column.

\*

\* @param double $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setX($v)

{

if ($v !== null) {

$v = (double) $v;

}

if ($this->x !== $v) {

$this->x = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_X] = true;

}

return $this;

} // setX()

/\*\*

\* Set the value of [y] column.

\*

\* @param double $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setY($v)

{

if ($v !== null) {

$v = (double) $v;

}

if ($this->y !== $v) {

$this->y = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_Y] = true;

}

return $this;

} // setY()

/\*\*

\* Set the value of [measure] column.

\*

\* @param double $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setMeasure($v)

{

if ($v !== null) {

$v = (double) $v;

}

if ($this->measure !== $v) {

$this->measure = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_MEASURE] = true;

}

return $this;

} // setMeasure()

/\*\*

\* Set the value of [kszam] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setKszam($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->kszam !== $v) {

$this->kszam = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_KSZAM] = true;

}

return $this;

} // setKszam()

/\*\*

\* Set the value of [pkod] column.

\*

\* @param double $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setPkod($v)

{

if ($v !== null) {

$v = (double) $v;

}

if ($this->pkod !== $v) {

$this->pkod = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_PKOD] = true;

}

return $this;

} // setPkod()

/\*\*

\* Set the value of [kkod] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setKkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->kkod !== $v) {

$this->kkod = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_KKOD] = true;

}

return $this;

} // setKkod()

/\*\*

\* Set the value of [vvkod] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setVvkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->vvkod !== $v) {

$this->vvkod = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_VVKOD] = true;

}

return $this;

} // setVvkod()

/\*\*

\* Set the value of [kszelv] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setKszelv($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->kszelv !== $v) {

$this->kszelv = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_KSZELV] = true;

}

return $this;

} // setKszelv()

/\*\*

\* Set the value of [vszelv] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setVszelv($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->vszelv !== $v) {

$this->vszelv = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_VSZELV] = true;

}

return $this;

} // setVszelv()

/\*\*

\* Set the value of [rshossz] column.

\*

\* @param double $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setRshossz($v)

{

if ($v !== null) {

$v = (double) $v;

}

if ($this->rshossz !== $v) {

$this->rshossz = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_RSHOSSZ] = true;

}

return $this;

} // setRshossz()

/\*\*

\* Set the value of [anf] column.

\*

\* @param int $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setAnf($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->anf !== $v) {

$this->anf = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_ANF] = true;

}

return $this;

} // setAnf()

/\*\*

\* Set the value of [anet] column.

\*

\* @param int $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setAnet($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->anet !== $v) {

$this->anet = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_ANET] = true;

}

return $this;

} // setAnet()

/\*\*

\* Set the value of [mof] column.

\*

\* @param int $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setMof($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->mof !== $v) {

$this->mof = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_MOF] = true;

}

return $this;

} // setMof()

/\*\*

\* Set the value of [ongj] column.

\*

\* @param int $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setOngj($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->ongj !== $v) {

$this->ongj = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_ONGJ] = true;

}

return $this;

} // setOngj()

/\*\*

\* Set the value of [oj] column.

\*

\* @param int $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setOj($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->oj !== $v) {

$this->oj = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_OJ] = true;

}

return $this;

} // setOj()

/\*\*

\* Set the value of [omot] column.

\*

\* @param int $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setOmot($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->omot !== $v) {

$this->omot = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_OMOT] = true;

}

return $this;

} // setOmot()

/\*\*

\* Set the value of [ev] column.

\*

\* @param int $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setEv($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->ev !== $v) {

$this->ev = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_EV] = true;

}

return $this;

} // setEv()

/\*\*

\* Set the value of [asz] column.

\*

\* @param int $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setAsz($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->asz !== $v) {

$this->asz = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_ASZ] = true;

}

return $this;

} // setAsz()

/\*\*

\* Set the value of [buszcs] column.

\*

\* @param int $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setBuszcs($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->buszcs !== $v) {

$this->buszcs = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_BUSZCS] = true;

}

return $this;

} // setBuszcs()

/\*\*

\* Set the value of [busze] column.

\*

\* @param int $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setBusze($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->busze !== $v) {

$this->busze = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_BUSZE] = true;

}

return $this;

} // setBusze()

/\*\*

\* Set the value of [obusz] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setObusz($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->obusz !== $v) {

$this->obusz = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_OBUSZ] = true;

}

return $this;

} // setObusz()

/\*\*

\* Set the value of [nyszer] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setNyszer($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->nyszer !== $v) {

$this->nyszer = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_NYSZER] = true;

}

return $this;

} // setNyszer()

/\*\*

\* Set the value of [potktgk] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setPotktgk($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->potktgk !== $v) {

$this->potktgk = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_POTKTGK] = true;

}

return $this;

} // setPotktgk()

/\*\*

\* Set the value of [ktgk] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setKtgk($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->ktgk !== $v) {

$this->ktgk = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_KTGK] = true;

}

return $this;

} // setKtgk()

/\*\*

\* Set the value of [ntgk] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setNtgk($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->ntgk !== $v) {

$this->ntgk = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_NTGK] = true;

}

return $this;

} // setNtgk()

/\*\*

\* Set the value of [kntgk] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setKntgk($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->kntgk !== $v) {

$this->kntgk = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_KNTGK] = true;

}

return $this;

} // setKntgk()

/\*\*

\* Set the value of [otgk] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setOtgk($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->otgk !== $v) {

$this->otgk = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_OTGK] = true;

}

return $this;

} // setOtgk()

/\*\*

\* Set the value of [szgk] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setSzgk($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->szgk !== $v) {

$this->szgk = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_SZGK] = true;

}

return $this;

} // setSzgk()

/\*\*

\* Set the value of [oszgk] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setOszgk($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->oszgk !== $v) {

$this->oszgk = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_OSZGK] = true;

}

return $this;

} // setOszgk()

/\*\*

\* Set the value of [mkp] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setMkp($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->mkp !== $v) {

$this->mkp = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_MKP] = true;

}

return $this;

} // setMkp()

/\*\*

\* Set the value of [kpf] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setKpf($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->kpf !== $v) {

$this->kpf = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_KPF] = true;

}

return $this;

} // setKpf()

/\*\*

\* Set the value of [lassu] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setLassu($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->lassu !== $v) {

$this->lassu = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_LASSU] = true;

}

return $this;

} // setLassu()

/\*\*

\* Set the value of [spec] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setSpec($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->spec !== $v) {

$this->spec = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_SPEC] = true;

}

return $this;

} // setSpec()

/\*\*

\* Set the value of [fmegb] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setFmegb($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->fmegb !== $v) {

$this->fmegb = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_FMEGB] = true;

}

return $this;

} // setFmegb()

/\*\*

\* Set the value of [adatforr] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setAdatforr($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->adatforr !== $v) {

$this->adatforr = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_ADATFORR] = true;

}

return $this;

} // setAdatforr()

/\*\*

\* Set the value of [szamlnap] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setSzamlnap($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->szamlnap !== $v) {

$this->szamlnap = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_SZAMLNAP] = true;

}

return $this;

} // setSzamlnap()

/\*\*

\* Set the value of [jelleg1] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setJelleg1($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->jelleg1 !== $v) {

$this->jelleg1 = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_JELLEG1] = true;

}

return $this;

} // setJelleg1()

/\*\*

\* Set the value of [jelleg2] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setJelleg2($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->jelleg2 !== $v) {

$this->jelleg2 = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_JELLEG2] = true;

}

return $this;

} // setJelleg2()

/\*\*

\* Set the value of [fmegj] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalom The current object (for fluent API support)

\*/

public function setFmegj($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->fmegj !== $v) {

$this->fmegj = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomTableMap::COL\_FMEGJ] = true;

}

return $this;

} // setFmegj()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgKeresztmetszetiForgalomTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgKeresztmetszetiForgalomQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgKeresztmetszetiForgalom::setDeleted()

\* @see StgKeresztmetszetiForgalom::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKeresztmetszetiForgalomTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgKeresztmetszetiForgalomQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKeresztmetszetiForgalomTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgKeresztmetszetiForgalomTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_SHAPENR)) {

$modifiedColumns[':p' . $index++] = 'SHAPENR';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_PARTNR)) {

$modifiedColumns[':p' . $index++] = 'PARTNR';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_NRPARTS)) {

$modifiedColumns[':p' . $index++] = 'NRPARTS';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_POINTNR)) {

$modifiedColumns[':p' . $index++] = 'POINTNR';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_NRPOINTS)) {

$modifiedColumns[':p' . $index++] = 'NRPOINTS';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_X)) {

$modifiedColumns[':p' . $index++] = 'X';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_Y)) {

$modifiedColumns[':p' . $index++] = 'Y';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_MEASURE)) {

$modifiedColumns[':p' . $index++] = 'MEASURE';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_KSZAM)) {

$modifiedColumns[':p' . $index++] = 'KSZAM';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_PKOD)) {

$modifiedColumns[':p' . $index++] = 'PKOD';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_KKOD)) {

$modifiedColumns[':p' . $index++] = 'KKOD';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_VVKOD)) {

$modifiedColumns[':p' . $index++] = 'VVKOD';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_KSZELV)) {

$modifiedColumns[':p' . $index++] = 'KSZELV';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_VSZELV)) {

$modifiedColumns[':p' . $index++] = 'VSZELV';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_RSHOSSZ)) {

$modifiedColumns[':p' . $index++] = 'RSHOSSZ';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_ANF)) {

$modifiedColumns[':p' . $index++] = 'ANF';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_ANET)) {

$modifiedColumns[':p' . $index++] = 'ANET';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_MOF)) {

$modifiedColumns[':p' . $index++] = 'MOF';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_ONGJ)) {

$modifiedColumns[':p' . $index++] = 'ONGJ';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_OJ)) {

$modifiedColumns[':p' . $index++] = 'OJ';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_OMOT)) {

$modifiedColumns[':p' . $index++] = 'OMOT';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_EV)) {

$modifiedColumns[':p' . $index++] = 'EV';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_ASZ)) {

$modifiedColumns[':p' . $index++] = 'ASZ';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_BUSZCS)) {

$modifiedColumns[':p' . $index++] = 'BUSZCS';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_BUSZE)) {

$modifiedColumns[':p' . $index++] = 'BUSZE';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_OBUSZ)) {

$modifiedColumns[':p' . $index++] = 'OBUSZ';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_NYSZER)) {

$modifiedColumns[':p' . $index++] = 'NYSZER';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_POTKTGK)) {

$modifiedColumns[':p' . $index++] = 'POTKTGK';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_KTGK)) {

$modifiedColumns[':p' . $index++] = 'KTGK';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_NTGK)) {

$modifiedColumns[':p' . $index++] = 'NTGK';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_KNTGK)) {

$modifiedColumns[':p' . $index++] = 'KNTGK';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_OTGK)) {

$modifiedColumns[':p' . $index++] = 'OTGK';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_SZGK)) {

$modifiedColumns[':p' . $index++] = 'SZGK';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_OSZGK)) {

$modifiedColumns[':p' . $index++] = 'OSZGK';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_MKP)) {

$modifiedColumns[':p' . $index++] = 'MKP';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_KPF)) {

$modifiedColumns[':p' . $index++] = 'KPF';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_LASSU)) {

$modifiedColumns[':p' . $index++] = 'LASSU';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_SPEC)) {

$modifiedColumns[':p' . $index++] = 'SPEC';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_FMEGB)) {

$modifiedColumns[':p' . $index++] = 'FMEGB';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_ADATFORR)) {

$modifiedColumns[':p' . $index++] = 'ADATFORR';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_SZAMLNAP)) {

$modifiedColumns[':p' . $index++] = 'SZAMLNAP';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_JELLEG1)) {

$modifiedColumns[':p' . $index++] = 'JELLEG1';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_JELLEG2)) {

$modifiedColumns[':p' . $index++] = 'JELLEG2';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_FMEGJ)) {

$modifiedColumns[':p' . $index++] = 'FMEGJ';

}

$sql = sprintf(

'INSERT INTO stg\_keresztmetszeti\_forgalom (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'SHAPENR':

$stmt->bindValue($identifier, $this->shapenr, PDO::PARAM\_INT);

break;

case 'PARTNR':

$stmt->bindValue($identifier, $this->partnr, PDO::PARAM\_INT);

break;

case 'NRPARTS':

$stmt->bindValue($identifier, $this->nrparts, PDO::PARAM\_INT);

break;

case 'POINTNR':

$stmt->bindValue($identifier, $this->pointnr, PDO::PARAM\_INT);

break;

case 'NRPOINTS':

$stmt->bindValue($identifier, $this->nrpoints, PDO::PARAM\_INT);

break;

case 'X':

$stmt->bindValue($identifier, $this->x, PDO::PARAM\_STR);

break;

case 'Y':

$stmt->bindValue($identifier, $this->y, PDO::PARAM\_STR);

break;

case 'MEASURE':

$stmt->bindValue($identifier, $this->measure, PDO::PARAM\_STR);

break;

case 'KSZAM':

$stmt->bindValue($identifier, $this->kszam, PDO::PARAM\_STR);

break;

case 'PKOD':

$stmt->bindValue($identifier, $this->pkod, PDO::PARAM\_STR);

break;

case 'KKOD':

$stmt->bindValue($identifier, $this->kkod, PDO::PARAM\_STR);

break;

case 'VVKOD':

$stmt->bindValue($identifier, $this->vvkod, PDO::PARAM\_STR);

break;

case 'KSZELV':

$stmt->bindValue($identifier, $this->kszelv, PDO::PARAM\_STR);

break;

case 'VSZELV':

$stmt->bindValue($identifier, $this->vszelv, PDO::PARAM\_STR);

break;

case 'RSHOSSZ':

$stmt->bindValue($identifier, $this->rshossz, PDO::PARAM\_STR);

break;

case 'ANF':

$stmt->bindValue($identifier, $this->anf, PDO::PARAM\_INT);

break;

case 'ANET':

$stmt->bindValue($identifier, $this->anet, PDO::PARAM\_INT);

break;

case 'MOF':

$stmt->bindValue($identifier, $this->mof, PDO::PARAM\_INT);

break;

case 'ONGJ':

$stmt->bindValue($identifier, $this->ongj, PDO::PARAM\_INT);

break;

case 'OJ':

$stmt->bindValue($identifier, $this->oj, PDO::PARAM\_INT);

break;

case 'OMOT':

$stmt->bindValue($identifier, $this->omot, PDO::PARAM\_INT);

break;

case 'EV':

$stmt->bindValue($identifier, $this->ev, PDO::PARAM\_INT);

break;

case 'ASZ':

$stmt->bindValue($identifier, $this->asz, PDO::PARAM\_INT);

break;

case 'BUSZCS':

$stmt->bindValue($identifier, $this->buszcs, PDO::PARAM\_INT);

break;

case 'BUSZE':

$stmt->bindValue($identifier, $this->busze, PDO::PARAM\_INT);

break;

case 'OBUSZ':

$stmt->bindValue($identifier, $this->obusz, PDO::PARAM\_STR);

break;

case 'NYSZER':

$stmt->bindValue($identifier, $this->nyszer, PDO::PARAM\_STR);

break;

case 'POTKTGK':

$stmt->bindValue($identifier, $this->potktgk, PDO::PARAM\_STR);

break;

case 'KTGK':

$stmt->bindValue($identifier, $this->ktgk, PDO::PARAM\_STR);

break;

case 'NTGK':

$stmt->bindValue($identifier, $this->ntgk, PDO::PARAM\_STR);

break;

case 'KNTGK':

$stmt->bindValue($identifier, $this->kntgk, PDO::PARAM\_STR);

break;

case 'OTGK':

$stmt->bindValue($identifier, $this->otgk, PDO::PARAM\_STR);

break;

case 'SZGK':

$stmt->bindValue($identifier, $this->szgk, PDO::PARAM\_STR);

break;

case 'OSZGK':

$stmt->bindValue($identifier, $this->oszgk, PDO::PARAM\_STR);

break;

case 'MKP':

$stmt->bindValue($identifier, $this->mkp, PDO::PARAM\_STR);

break;

case 'KPF':

$stmt->bindValue($identifier, $this->kpf, PDO::PARAM\_STR);

break;

case 'LASSU':

$stmt->bindValue($identifier, $this->lassu, PDO::PARAM\_STR);

break;

case 'SPEC':

$stmt->bindValue($identifier, $this->spec, PDO::PARAM\_STR);

break;

case 'FMEGB':

$stmt->bindValue($identifier, $this->fmegb, PDO::PARAM\_STR);

break;

case 'ADATFORR':

$stmt->bindValue($identifier, $this->adatforr, PDO::PARAM\_STR);

break;

case 'SZAMLNAP':

$stmt->bindValue($identifier, $this->szamlnap, PDO::PARAM\_STR);

break;

case 'JELLEG1':

$stmt->bindValue($identifier, $this->jelleg1, PDO::PARAM\_STR);

break;

case 'JELLEG2':

$stmt->bindValue($identifier, $this->jelleg2, PDO::PARAM\_STR);

break;

case 'FMEGJ':

$stmt->bindValue($identifier, $this->fmegj, PDO::PARAM\_STR);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgKeresztmetszetiForgalomTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getShapenr();

break;

case 1:

return $this->getPartnr();

break;

case 2:

return $this->getNrparts();

break;

case 3:

return $this->getPointnr();

break;

case 4:

return $this->getNrpoints();

break;

case 5:

return $this->getX();

break;

case 6:

return $this->getY();

break;

case 7:

return $this->getMeasure();

break;

case 8:

return $this->getKszam();

break;

case 9:

return $this->getPkod();

break;

case 10:

return $this->getKkod();

break;

case 11:

return $this->getVvkod();

break;

case 12:

return $this->getKszelv();

break;

case 13:

return $this->getVszelv();

break;

case 14:

return $this->getRshossz();

break;

case 15:

return $this->getAnf();

break;

case 16:

return $this->getAnet();

break;

case 17:

return $this->getMof();

break;

case 18:

return $this->getOngj();

break;

case 19:

return $this->getOj();

break;

case 20:

return $this->getOmot();

break;

case 21:

return $this->getEv();

break;

case 22:

return $this->getAsz();

break;

case 23:

return $this->getBuszcs();

break;

case 24:

return $this->getBusze();

break;

case 25:

return $this->getObusz();

break;

case 26:

return $this->getNyszer();

break;

case 27:

return $this->getPotktgk();

break;

case 28:

return $this->getKtgk();

break;

case 29:

return $this->getNtgk();

break;

case 30:

return $this->getKntgk();

break;

case 31:

return $this->getOtgk();

break;

case 32:

return $this->getSzgk();

break;

case 33:

return $this->getOszgk();

break;

case 34:

return $this->getMkp();

break;

case 35:

return $this->getKpf();

break;

case 36:

return $this->getLassu();

break;

case 37:

return $this->getSpec();

break;

case 38:

return $this->getFmegb();

break;

case 39:

return $this->getAdatforr();

break;

case 40:

return $this->getSzamlnap();

break;

case 41:

return $this->getJelleg1();

break;

case 42:

return $this->getJelleg2();

break;

case 43:

return $this->getFmegj();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgKeresztmetszetiForgalom'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgKeresztmetszetiForgalom'][$this->getPrimaryKey()] = true;

$keys = StgKeresztmetszetiForgalomTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getShapenr(),

$keys[1] => $this->getPartnr(),

$keys[2] => $this->getNrparts(),

$keys[3] => $this->getPointnr(),

$keys[4] => $this->getNrpoints(),

$keys[5] => $this->getX(),

$keys[6] => $this->getY(),

$keys[7] => $this->getMeasure(),

$keys[8] => $this->getKszam(),

$keys[9] => $this->getPkod(),

$keys[10] => $this->getKkod(),

$keys[11] => $this->getVvkod(),

$keys[12] => $this->getKszelv(),

$keys[13] => $this->getVszelv(),

$keys[14] => $this->getRshossz(),

$keys[15] => $this->getAnf(),

$keys[16] => $this->getAnet(),

$keys[17] => $this->getMof(),

$keys[18] => $this->getOngj(),

$keys[19] => $this->getOj(),

$keys[20] => $this->getOmot(),

$keys[21] => $this->getEv(),

$keys[22] => $this->getAsz(),

$keys[23] => $this->getBuszcs(),

$keys[24] => $this->getBusze(),

$keys[25] => $this->getObusz(),

$keys[26] => $this->getNyszer(),

$keys[27] => $this->getPotktgk(),

$keys[28] => $this->getKtgk(),

$keys[29] => $this->getNtgk(),

$keys[30] => $this->getKntgk(),

$keys[31] => $this->getOtgk(),

$keys[32] => $this->getSzgk(),

$keys[33] => $this->getOszgk(),

$keys[34] => $this->getMkp(),

$keys[35] => $this->getKpf(),

$keys[36] => $this->getLassu(),

$keys[37] => $this->getSpec(),

$keys[38] => $this->getFmegb(),

$keys[39] => $this->getAdatforr(),

$keys[40] => $this->getSzamlnap(),

$keys[41] => $this->getJelleg1(),

$keys[42] => $this->getJelleg2(),

$keys[43] => $this->getFmegj(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgKeresztmetszetiForgalom

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgKeresztmetszetiForgalomTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgKeresztmetszetiForgalom

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setShapenr($value);

break;

case 1:

$this->setPartnr($value);

break;

case 2:

$this->setNrparts($value);

break;

case 3:

$this->setPointnr($value);

break;

case 4:

$this->setNrpoints($value);

break;

case 5:

$this->setX($value);

break;

case 6:

$this->setY($value);

break;

case 7:

$this->setMeasure($value);

break;

case 8:

$this->setKszam($value);

break;

case 9:

$this->setPkod($value);

break;

case 10:

$this->setKkod($value);

break;

case 11:

$this->setVvkod($value);

break;

case 12:

$this->setKszelv($value);

break;

case 13:

$this->setVszelv($value);

break;

case 14:

$this->setRshossz($value);

break;

case 15:

$this->setAnf($value);

break;

case 16:

$this->setAnet($value);

break;

case 17:

$this->setMof($value);

break;

case 18:

$this->setOngj($value);

break;

case 19:

$this->setOj($value);

break;

case 20:

$this->setOmot($value);

break;

case 21:

$this->setEv($value);

break;

case 22:

$this->setAsz($value);

break;

case 23:

$this->setBuszcs($value);

break;

case 24:

$this->setBusze($value);

break;

case 25:

$this->setObusz($value);

break;

case 26:

$this->setNyszer($value);

break;

case 27:

$this->setPotktgk($value);

break;

case 28:

$this->setKtgk($value);

break;

case 29:

$this->setNtgk($value);

break;

case 30:

$this->setKntgk($value);

break;

case 31:

$this->setOtgk($value);

break;

case 32:

$this->setSzgk($value);

break;

case 33:

$this->setOszgk($value);

break;

case 34:

$this->setMkp($value);

break;

case 35:

$this->setKpf($value);

break;

case 36:

$this->setLassu($value);

break;

case 37:

$this->setSpec($value);

break;

case 38:

$this->setFmegb($value);

break;

case 39:

$this->setAdatforr($value);

break;

case 40:

$this->setSzamlnap($value);

break;

case 41:

$this->setJelleg1($value);

break;

case 42:

$this->setJelleg2($value);

break;

case 43:

$this->setFmegj($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgKeresztmetszetiForgalomTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setShapenr($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setPartnr($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setNrparts($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setPointnr($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setNrpoints($arr[$keys[4]]);

}

if (array\_key\_exists($keys[5], $arr)) {

$this->setX($arr[$keys[5]]);

}

if (array\_key\_exists($keys[6], $arr)) {

$this->setY($arr[$keys[6]]);

}

if (array\_key\_exists($keys[7], $arr)) {

$this->setMeasure($arr[$keys[7]]);

}

if (array\_key\_exists($keys[8], $arr)) {

$this->setKszam($arr[$keys[8]]);

}

if (array\_key\_exists($keys[9], $arr)) {

$this->setPkod($arr[$keys[9]]);

}

if (array\_key\_exists($keys[10], $arr)) {

$this->setKkod($arr[$keys[10]]);

}

if (array\_key\_exists($keys[11], $arr)) {

$this->setVvkod($arr[$keys[11]]);

}

if (array\_key\_exists($keys[12], $arr)) {

$this->setKszelv($arr[$keys[12]]);

}

if (array\_key\_exists($keys[13], $arr)) {

$this->setVszelv($arr[$keys[13]]);

}

if (array\_key\_exists($keys[14], $arr)) {

$this->setRshossz($arr[$keys[14]]);

}

if (array\_key\_exists($keys[15], $arr)) {

$this->setAnf($arr[$keys[15]]);

}

if (array\_key\_exists($keys[16], $arr)) {

$this->setAnet($arr[$keys[16]]);

}

if (array\_key\_exists($keys[17], $arr)) {

$this->setMof($arr[$keys[17]]);

}

if (array\_key\_exists($keys[18], $arr)) {

$this->setOngj($arr[$keys[18]]);

}

if (array\_key\_exists($keys[19], $arr)) {

$this->setOj($arr[$keys[19]]);

}

if (array\_key\_exists($keys[20], $arr)) {

$this->setOmot($arr[$keys[20]]);

}

if (array\_key\_exists($keys[21], $arr)) {

$this->setEv($arr[$keys[21]]);

}

if (array\_key\_exists($keys[22], $arr)) {

$this->setAsz($arr[$keys[22]]);

}

if (array\_key\_exists($keys[23], $arr)) {

$this->setBuszcs($arr[$keys[23]]);

}

if (array\_key\_exists($keys[24], $arr)) {

$this->setBusze($arr[$keys[24]]);

}

if (array\_key\_exists($keys[25], $arr)) {

$this->setObusz($arr[$keys[25]]);

}

if (array\_key\_exists($keys[26], $arr)) {

$this->setNyszer($arr[$keys[26]]);

}

if (array\_key\_exists($keys[27], $arr)) {

$this->setPotktgk($arr[$keys[27]]);

}

if (array\_key\_exists($keys[28], $arr)) {

$this->setKtgk($arr[$keys[28]]);

}

if (array\_key\_exists($keys[29], $arr)) {

$this->setNtgk($arr[$keys[29]]);

}

if (array\_key\_exists($keys[30], $arr)) {

$this->setKntgk($arr[$keys[30]]);

}

if (array\_key\_exists($keys[31], $arr)) {

$this->setOtgk($arr[$keys[31]]);

}

if (array\_key\_exists($keys[32], $arr)) {

$this->setSzgk($arr[$keys[32]]);

}

if (array\_key\_exists($keys[33], $arr)) {

$this->setOszgk($arr[$keys[33]]);

}

if (array\_key\_exists($keys[34], $arr)) {

$this->setMkp($arr[$keys[34]]);

}

if (array\_key\_exists($keys[35], $arr)) {

$this->setKpf($arr[$keys[35]]);

}

if (array\_key\_exists($keys[36], $arr)) {

$this->setLassu($arr[$keys[36]]);

}

if (array\_key\_exists($keys[37], $arr)) {

$this->setSpec($arr[$keys[37]]);

}

if (array\_key\_exists($keys[38], $arr)) {

$this->setFmegb($arr[$keys[38]]);

}

if (array\_key\_exists($keys[39], $arr)) {

$this->setAdatforr($arr[$keys[39]]);

}

if (array\_key\_exists($keys[40], $arr)) {

$this->setSzamlnap($arr[$keys[40]]);

}

if (array\_key\_exists($keys[41], $arr)) {

$this->setJelleg1($arr[$keys[41]]);

}

if (array\_key\_exists($keys[42], $arr)) {

$this->setJelleg2($arr[$keys[42]]);

}

if (array\_key\_exists($keys[43], $arr)) {

$this->setFmegj($arr[$keys[43]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgKeresztmetszetiForgalom The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgKeresztmetszetiForgalomTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_SHAPENR)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_SHAPENR, $this->shapenr);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_PARTNR)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_PARTNR, $this->partnr);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_NRPARTS)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_NRPARTS, $this->nrparts);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_POINTNR)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_POINTNR, $this->pointnr);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_NRPOINTS)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_NRPOINTS, $this->nrpoints);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_X)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_X, $this->x);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_Y)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_Y, $this->y);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_MEASURE)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_MEASURE, $this->measure);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_KSZAM)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_KSZAM, $this->kszam);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_PKOD)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_PKOD, $this->pkod);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_KKOD)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_KKOD, $this->kkod);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_VVKOD)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_VVKOD, $this->vvkod);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_KSZELV)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_KSZELV, $this->kszelv);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_VSZELV)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_VSZELV, $this->vszelv);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_RSHOSSZ)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_RSHOSSZ, $this->rshossz);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_ANF)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_ANF, $this->anf);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_ANET)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_ANET, $this->anet);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_MOF)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_MOF, $this->mof);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_ONGJ)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_ONGJ, $this->ongj);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_OJ)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_OJ, $this->oj);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_OMOT)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_OMOT, $this->omot);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_EV)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_EV, $this->ev);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_ASZ)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_ASZ, $this->asz);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_BUSZCS)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_BUSZCS, $this->buszcs);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_BUSZE)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_BUSZE, $this->busze);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_OBUSZ)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_OBUSZ, $this->obusz);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_NYSZER)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_NYSZER, $this->nyszer);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_POTKTGK)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_POTKTGK, $this->potktgk);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_KTGK)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_KTGK, $this->ktgk);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_NTGK)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_NTGK, $this->ntgk);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_KNTGK)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_KNTGK, $this->kntgk);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_OTGK)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_OTGK, $this->otgk);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_SZGK)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_SZGK, $this->szgk);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_OSZGK)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_OSZGK, $this->oszgk);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_MKP)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_MKP, $this->mkp);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_KPF)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_KPF, $this->kpf);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_LASSU)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_LASSU, $this->lassu);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_SPEC)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_SPEC, $this->spec);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_FMEGB)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_FMEGB, $this->fmegb);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_ADATFORR)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_ADATFORR, $this->adatforr);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_SZAMLNAP)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_SZAMLNAP, $this->szamlnap);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_JELLEG1)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_JELLEG1, $this->jelleg1);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_JELLEG2)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_JELLEG2, $this->jelleg2);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomTableMap::COL\_FMEGJ)) {

$criteria->add(StgKeresztmetszetiForgalomTableMap::COL\_FMEGJ, $this->fmegj);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgKeresztmetszetiForgalom object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgKeresztmetszetiForgalom (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setShapenr($this->getShapenr());

$copyObj->setPartnr($this->getPartnr());

$copyObj->setNrparts($this->getNrparts());

$copyObj->setPointnr($this->getPointnr());

$copyObj->setNrpoints($this->getNrpoints());

$copyObj->setX($this->getX());

$copyObj->setY($this->getY());

$copyObj->setMeasure($this->getMeasure());

$copyObj->setKszam($this->getKszam());

$copyObj->setPkod($this->getPkod());

$copyObj->setKkod($this->getKkod());

$copyObj->setVvkod($this->getVvkod());

$copyObj->setKszelv($this->getKszelv());

$copyObj->setVszelv($this->getVszelv());

$copyObj->setRshossz($this->getRshossz());

$copyObj->setAnf($this->getAnf());

$copyObj->setAnet($this->getAnet());

$copyObj->setMof($this->getMof());

$copyObj->setOngj($this->getOngj());

$copyObj->setOj($this->getOj());

$copyObj->setOmot($this->getOmot());

$copyObj->setEv($this->getEv());

$copyObj->setAsz($this->getAsz());

$copyObj->setBuszcs($this->getBuszcs());

$copyObj->setBusze($this->getBusze());

$copyObj->setObusz($this->getObusz());

$copyObj->setNyszer($this->getNyszer());

$copyObj->setPotktgk($this->getPotktgk());

$copyObj->setKtgk($this->getKtgk());

$copyObj->setNtgk($this->getNtgk());

$copyObj->setKntgk($this->getKntgk());

$copyObj->setOtgk($this->getOtgk());

$copyObj->setSzgk($this->getSzgk());

$copyObj->setOszgk($this->getOszgk());

$copyObj->setMkp($this->getMkp());

$copyObj->setKpf($this->getKpf());

$copyObj->setLassu($this->getLassu());

$copyObj->setSpec($this->getSpec());

$copyObj->setFmegb($this->getFmegb());

$copyObj->setAdatforr($this->getAdatforr());

$copyObj->setSzamlnap($this->getSzamlnap());

$copyObj->setJelleg1($this->getJelleg1());

$copyObj->setJelleg2($this->getJelleg2());

$copyObj->setFmegj($this->getFmegj());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgKeresztmetszetiForgalom Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->shapenr = null;

$this->partnr = null;

$this->nrparts = null;

$this->pointnr = null;

$this->nrpoints = null;

$this->x = null;

$this->y = null;

$this->measure = null;

$this->kszam = null;

$this->pkod = null;

$this->kkod = null;

$this->vvkod = null;

$this->kszelv = null;

$this->vszelv = null;

$this->rshossz = null;

$this->anf = null;

$this->anet = null;

$this->mof = null;

$this->ongj = null;

$this->oj = null;

$this->omot = null;

$this->ev = null;

$this->asz = null;

$this->buszcs = null;

$this->busze = null;

$this->obusz = null;

$this->nyszer = null;

$this->potktgk = null;

$this->ktgk = null;

$this->ntgk = null;

$this->kntgk = null;

$this->otgk = null;

$this->szgk = null;

$this->oszgk = null;

$this->mkp = null;

$this->kpf = null;

$this->lassu = null;

$this->spec = null;

$this->fmegb = null;

$this->adatforr = null;

$this->szamlnap = null;

$this->jelleg1 = null;

$this->jelleg2 = null;

$this->fmegj = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgKeresztmetszetiForgalomTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgKeresztmetszetiForgalomQuery.php

<?php

namespace Base;

use \StgKeresztmetszetiForgalom as ChildStgKeresztmetszetiForgalom;

use \StgKeresztmetszetiForgalomQuery as ChildStgKeresztmetszetiForgalomQuery;

use \Exception;

use Map\StgKeresztmetszetiForgalomTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_keresztmetszeti\_forgalom' table.

\*

\*

\*

\* @method ChildStgKeresztmetszetiForgalomQuery orderByShapenr($order = Criteria::ASC) Order by the shapenr column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByPartnr($order = Criteria::ASC) Order by the partnr column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByNrparts($order = Criteria::ASC) Order by the nrparts column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByPointnr($order = Criteria::ASC) Order by the pointnr column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByNrpoints($order = Criteria::ASC) Order by the nrpointS column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByX($order = Criteria::ASC) Order by the x column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByY($order = Criteria::ASC) Order by the y column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByMeasure($order = Criteria::ASC) Order by the measure column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByKszam($order = Criteria::ASC) Order by the kszam column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByPkod($order = Criteria::ASC) Order by the pkod column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByKkod($order = Criteria::ASC) Order by the kkod column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByVvkod($order = Criteria::ASC) Order by the vvkod column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByKszelv($order = Criteria::ASC) Order by the kszelv column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByVszelv($order = Criteria::ASC) Order by the vszelv column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByRshossz($order = Criteria::ASC) Order by the RSHOSSZ column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByAnf($order = Criteria::ASC) Order by the ANF column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByAnet($order = Criteria::ASC) Order by the ANET column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByMof($order = Criteria::ASC) Order by the MOF column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByOngj($order = Criteria::ASC) Order by the ONGJ column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByOj($order = Criteria::ASC) Order by the OJ column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByOmot($order = Criteria::ASC) Order by the OMOT column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByEv($order = Criteria::ASC) Order by the EV column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByAsz($order = Criteria::ASC) Order by the ASZ column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByBuszcs($order = Criteria::ASC) Order by the BUSZCS column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByBusze($order = Criteria::ASC) Order by the BUSZE column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByObusz($order = Criteria::ASC) Order by the OBUSZ column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByNyszer($order = Criteria::ASC) Order by the NYSZER column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByPotktgk($order = Criteria::ASC) Order by the POTKTGK column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByKtgk($order = Criteria::ASC) Order by the KTGK column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByNtgk($order = Criteria::ASC) Order by the NTGK column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByKntgk($order = Criteria::ASC) Order by the KNTGK column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByOtgk($order = Criteria::ASC) Order by the OTGK column

\* @method ChildStgKeresztmetszetiForgalomQuery orderBySzgk($order = Criteria::ASC) Order by the SZGK column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByOszgk($order = Criteria::ASC) Order by the OSZGK column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByMkp($order = Criteria::ASC) Order by the MKP column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByKpf($order = Criteria::ASC) Order by the KPF column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByLassu($order = Criteria::ASC) Order by the LASSU column

\* @method ChildStgKeresztmetszetiForgalomQuery orderBySpec($order = Criteria::ASC) Order by the SPEC column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByFmegb($order = Criteria::ASC) Order by the FMEGB column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByAdatforr($order = Criteria::ASC) Order by the ADATFORR column

\* @method ChildStgKeresztmetszetiForgalomQuery orderBySzamlnap($order = Criteria::ASC) Order by the SZAMLNAP column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByJelleg1($order = Criteria::ASC) Order by the JELLEG1 column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByJelleg2($order = Criteria::ASC) Order by the JELLEG2 column

\* @method ChildStgKeresztmetszetiForgalomQuery orderByFmegj($order = Criteria::ASC) Order by the FMEGJ column

\*

\* @method ChildStgKeresztmetszetiForgalomQuery groupByShapenr() Group by the shapenr column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByPartnr() Group by the partnr column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByNrparts() Group by the nrparts column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByPointnr() Group by the pointnr column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByNrpoints() Group by the nrpointS column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByX() Group by the x column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByY() Group by the y column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByMeasure() Group by the measure column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByKszam() Group by the kszam column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByPkod() Group by the pkod column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByKkod() Group by the kkod column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByVvkod() Group by the vvkod column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByKszelv() Group by the kszelv column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByVszelv() Group by the vszelv column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByRshossz() Group by the RSHOSSZ column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByAnf() Group by the ANF column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByAnet() Group by the ANET column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByMof() Group by the MOF column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByOngj() Group by the ONGJ column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByOj() Group by the OJ column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByOmot() Group by the OMOT column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByEv() Group by the EV column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByAsz() Group by the ASZ column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByBuszcs() Group by the BUSZCS column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByBusze() Group by the BUSZE column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByObusz() Group by the OBUSZ column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByNyszer() Group by the NYSZER column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByPotktgk() Group by the POTKTGK column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByKtgk() Group by the KTGK column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByNtgk() Group by the NTGK column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByKntgk() Group by the KNTGK column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByOtgk() Group by the OTGK column

\* @method ChildStgKeresztmetszetiForgalomQuery groupBySzgk() Group by the SZGK column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByOszgk() Group by the OSZGK column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByMkp() Group by the MKP column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByKpf() Group by the KPF column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByLassu() Group by the LASSU column

\* @method ChildStgKeresztmetszetiForgalomQuery groupBySpec() Group by the SPEC column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByFmegb() Group by the FMEGB column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByAdatforr() Group by the ADATFORR column

\* @method ChildStgKeresztmetszetiForgalomQuery groupBySzamlnap() Group by the SZAMLNAP column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByJelleg1() Group by the JELLEG1 column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByJelleg2() Group by the JELLEG2 column

\* @method ChildStgKeresztmetszetiForgalomQuery groupByFmegj() Group by the FMEGJ column

\*

\* @method ChildStgKeresztmetszetiForgalomQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgKeresztmetszetiForgalomQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgKeresztmetszetiForgalomQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgKeresztmetszetiForgalom findOne(ConnectionInterface $con = null) Return the first ChildStgKeresztmetszetiForgalom matching the query

\* @method ChildStgKeresztmetszetiForgalom findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgKeresztmetszetiForgalom matching the query, or a new ChildStgKeresztmetszetiForgalom object populated from the query conditions when no match is found

\*

\* @method ChildStgKeresztmetszetiForgalom findOneByShapenr(int $shapenr) Return the first ChildStgKeresztmetszetiForgalom filtered by the shapenr column

\* @method ChildStgKeresztmetszetiForgalom findOneByPartnr(int $partnr) Return the first ChildStgKeresztmetszetiForgalom filtered by the partnr column

\* @method ChildStgKeresztmetszetiForgalom findOneByNrparts(int $nrparts) Return the first ChildStgKeresztmetszetiForgalom filtered by the nrparts column

\* @method ChildStgKeresztmetszetiForgalom findOneByPointnr(int $pointnr) Return the first ChildStgKeresztmetszetiForgalom filtered by the pointnr column

\* @method ChildStgKeresztmetszetiForgalom findOneByNrpoints(int $nrpointS) Return the first ChildStgKeresztmetszetiForgalom filtered by the nrpointS column

\* @method ChildStgKeresztmetszetiForgalom findOneByX(double $x) Return the first ChildStgKeresztmetszetiForgalom filtered by the x column

\* @method ChildStgKeresztmetszetiForgalom findOneByY(double $y) Return the first ChildStgKeresztmetszetiForgalom filtered by the y column

\* @method ChildStgKeresztmetszetiForgalom findOneByMeasure(double $measure) Return the first ChildStgKeresztmetszetiForgalom filtered by the measure column

\* @method ChildStgKeresztmetszetiForgalom findOneByKszam(string $kszam) Return the first ChildStgKeresztmetszetiForgalom filtered by the kszam column

\* @method ChildStgKeresztmetszetiForgalom findOneByPkod(double $pkod) Return the first ChildStgKeresztmetszetiForgalom filtered by the pkod column

\* @method ChildStgKeresztmetszetiForgalom findOneByKkod(string $kkod) Return the first ChildStgKeresztmetszetiForgalom filtered by the kkod column

\* @method ChildStgKeresztmetszetiForgalom findOneByVvkod(string $vvkod) Return the first ChildStgKeresztmetszetiForgalom filtered by the vvkod column

\* @method ChildStgKeresztmetszetiForgalom findOneByKszelv(string $kszelv) Return the first ChildStgKeresztmetszetiForgalom filtered by the kszelv column

\* @method ChildStgKeresztmetszetiForgalom findOneByVszelv(string $vszelv) Return the first ChildStgKeresztmetszetiForgalom filtered by the vszelv column

\* @method ChildStgKeresztmetszetiForgalom findOneByRshossz(double $RSHOSSZ) Return the first ChildStgKeresztmetszetiForgalom filtered by the RSHOSSZ column

\* @method ChildStgKeresztmetszetiForgalom findOneByAnf(int $ANF) Return the first ChildStgKeresztmetszetiForgalom filtered by the ANF column

\* @method ChildStgKeresztmetszetiForgalom findOneByAnet(int $ANET) Return the first ChildStgKeresztmetszetiForgalom filtered by the ANET column

\* @method ChildStgKeresztmetszetiForgalom findOneByMof(int $MOF) Return the first ChildStgKeresztmetszetiForgalom filtered by the MOF column

\* @method ChildStgKeresztmetszetiForgalom findOneByOngj(int $ONGJ) Return the first ChildStgKeresztmetszetiForgalom filtered by the ONGJ column

\* @method ChildStgKeresztmetszetiForgalom findOneByOj(int $OJ) Return the first ChildStgKeresztmetszetiForgalom filtered by the OJ column

\* @method ChildStgKeresztmetszetiForgalom findOneByOmot(int $OMOT) Return the first ChildStgKeresztmetszetiForgalom filtered by the OMOT column

\* @method ChildStgKeresztmetszetiForgalom findOneByEv(int $EV) Return the first ChildStgKeresztmetszetiForgalom filtered by the EV column

\* @method ChildStgKeresztmetszetiForgalom findOneByAsz(int $ASZ) Return the first ChildStgKeresztmetszetiForgalom filtered by the ASZ column

\* @method ChildStgKeresztmetszetiForgalom findOneByBuszcs(int $BUSZCS) Return the first ChildStgKeresztmetszetiForgalom filtered by the BUSZCS column

\* @method ChildStgKeresztmetszetiForgalom findOneByBusze(int $BUSZE) Return the first ChildStgKeresztmetszetiForgalom filtered by the BUSZE column

\* @method ChildStgKeresztmetszetiForgalom findOneByObusz(string $OBUSZ) Return the first ChildStgKeresztmetszetiForgalom filtered by the OBUSZ column

\* @method ChildStgKeresztmetszetiForgalom findOneByNyszer(string $NYSZER) Return the first ChildStgKeresztmetszetiForgalom filtered by the NYSZER column

\* @method ChildStgKeresztmetszetiForgalom findOneByPotktgk(string $POTKTGK) Return the first ChildStgKeresztmetszetiForgalom filtered by the POTKTGK column

\* @method ChildStgKeresztmetszetiForgalom findOneByKtgk(string $KTGK) Return the first ChildStgKeresztmetszetiForgalom filtered by the KTGK column

\* @method ChildStgKeresztmetszetiForgalom findOneByNtgk(string $NTGK) Return the first ChildStgKeresztmetszetiForgalom filtered by the NTGK column

\* @method ChildStgKeresztmetszetiForgalom findOneByKntgk(string $KNTGK) Return the first ChildStgKeresztmetszetiForgalom filtered by the KNTGK column

\* @method ChildStgKeresztmetszetiForgalom findOneByOtgk(string $OTGK) Return the first ChildStgKeresztmetszetiForgalom filtered by the OTGK column

\* @method ChildStgKeresztmetszetiForgalom findOneBySzgk(string $SZGK) Return the first ChildStgKeresztmetszetiForgalom filtered by the SZGK column

\* @method ChildStgKeresztmetszetiForgalom findOneByOszgk(string $OSZGK) Return the first ChildStgKeresztmetszetiForgalom filtered by the OSZGK column

\* @method ChildStgKeresztmetszetiForgalom findOneByMkp(string $MKP) Return the first ChildStgKeresztmetszetiForgalom filtered by the MKP column

\* @method ChildStgKeresztmetszetiForgalom findOneByKpf(string $KPF) Return the first ChildStgKeresztmetszetiForgalom filtered by the KPF column

\* @method ChildStgKeresztmetszetiForgalom findOneByLassu(string $LASSU) Return the first ChildStgKeresztmetszetiForgalom filtered by the LASSU column

\* @method ChildStgKeresztmetszetiForgalom findOneBySpec(string $SPEC) Return the first ChildStgKeresztmetszetiForgalom filtered by the SPEC column

\* @method ChildStgKeresztmetszetiForgalom findOneByFmegb(string $FMEGB) Return the first ChildStgKeresztmetszetiForgalom filtered by the FMEGB column

\* @method ChildStgKeresztmetszetiForgalom findOneByAdatforr(string $ADATFORR) Return the first ChildStgKeresztmetszetiForgalom filtered by the ADATFORR column

\* @method ChildStgKeresztmetszetiForgalom findOneBySzamlnap(string $SZAMLNAP) Return the first ChildStgKeresztmetszetiForgalom filtered by the SZAMLNAP column

\* @method ChildStgKeresztmetszetiForgalom findOneByJelleg1(string $JELLEG1) Return the first ChildStgKeresztmetszetiForgalom filtered by the JELLEG1 column

\* @method ChildStgKeresztmetszetiForgalom findOneByJelleg2(string $JELLEG2) Return the first ChildStgKeresztmetszetiForgalom filtered by the JELLEG2 column

\* @method ChildStgKeresztmetszetiForgalom findOneByFmegj(string $FMEGJ) Return the first ChildStgKeresztmetszetiForgalom filtered by the FMEGJ column

\*

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgKeresztmetszetiForgalom objects based on current ModelCriteria

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByShapenr(int $shapenr) Return ChildStgKeresztmetszetiForgalom objects filtered by the shapenr column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByPartnr(int $partnr) Return ChildStgKeresztmetszetiForgalom objects filtered by the partnr column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByNrparts(int $nrparts) Return ChildStgKeresztmetszetiForgalom objects filtered by the nrparts column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByPointnr(int $pointnr) Return ChildStgKeresztmetszetiForgalom objects filtered by the pointnr column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByNrpoints(int $nrpointS) Return ChildStgKeresztmetszetiForgalom objects filtered by the nrpointS column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByX(double $x) Return ChildStgKeresztmetszetiForgalom objects filtered by the x column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByY(double $y) Return ChildStgKeresztmetszetiForgalom objects filtered by the y column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByMeasure(double $measure) Return ChildStgKeresztmetszetiForgalom objects filtered by the measure column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByKszam(string $kszam) Return ChildStgKeresztmetszetiForgalom objects filtered by the kszam column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByPkod(double $pkod) Return ChildStgKeresztmetszetiForgalom objects filtered by the pkod column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByKkod(string $kkod) Return ChildStgKeresztmetszetiForgalom objects filtered by the kkod column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByVvkod(string $vvkod) Return ChildStgKeresztmetszetiForgalom objects filtered by the vvkod column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByKszelv(string $kszelv) Return ChildStgKeresztmetszetiForgalom objects filtered by the kszelv column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByVszelv(string $vszelv) Return ChildStgKeresztmetszetiForgalom objects filtered by the vszelv column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByRshossz(double $RSHOSSZ) Return ChildStgKeresztmetszetiForgalom objects filtered by the RSHOSSZ column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByAnf(int $ANF) Return ChildStgKeresztmetszetiForgalom objects filtered by the ANF column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByAnet(int $ANET) Return ChildStgKeresztmetszetiForgalom objects filtered by the ANET column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByMof(int $MOF) Return ChildStgKeresztmetszetiForgalom objects filtered by the MOF column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByOngj(int $ONGJ) Return ChildStgKeresztmetszetiForgalom objects filtered by the ONGJ column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByOj(int $OJ) Return ChildStgKeresztmetszetiForgalom objects filtered by the OJ column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByOmot(int $OMOT) Return ChildStgKeresztmetszetiForgalom objects filtered by the OMOT column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByEv(int $EV) Return ChildStgKeresztmetszetiForgalom objects filtered by the EV column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByAsz(int $ASZ) Return ChildStgKeresztmetszetiForgalom objects filtered by the ASZ column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByBuszcs(int $BUSZCS) Return ChildStgKeresztmetszetiForgalom objects filtered by the BUSZCS column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByBusze(int $BUSZE) Return ChildStgKeresztmetszetiForgalom objects filtered by the BUSZE column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByObusz(string $OBUSZ) Return ChildStgKeresztmetszetiForgalom objects filtered by the OBUSZ column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByNyszer(string $NYSZER) Return ChildStgKeresztmetszetiForgalom objects filtered by the NYSZER column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByPotktgk(string $POTKTGK) Return ChildStgKeresztmetszetiForgalom objects filtered by the POTKTGK column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByKtgk(string $KTGK) Return ChildStgKeresztmetszetiForgalom objects filtered by the KTGK column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByNtgk(string $NTGK) Return ChildStgKeresztmetszetiForgalom objects filtered by the NTGK column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByKntgk(string $KNTGK) Return ChildStgKeresztmetszetiForgalom objects filtered by the KNTGK column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByOtgk(string $OTGK) Return ChildStgKeresztmetszetiForgalom objects filtered by the OTGK column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findBySzgk(string $SZGK) Return ChildStgKeresztmetszetiForgalom objects filtered by the SZGK column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByOszgk(string $OSZGK) Return ChildStgKeresztmetszetiForgalom objects filtered by the OSZGK column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByMkp(string $MKP) Return ChildStgKeresztmetszetiForgalom objects filtered by the MKP column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByKpf(string $KPF) Return ChildStgKeresztmetszetiForgalom objects filtered by the KPF column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByLassu(string $LASSU) Return ChildStgKeresztmetszetiForgalom objects filtered by the LASSU column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findBySpec(string $SPEC) Return ChildStgKeresztmetszetiForgalom objects filtered by the SPEC column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByFmegb(string $FMEGB) Return ChildStgKeresztmetszetiForgalom objects filtered by the FMEGB column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByAdatforr(string $ADATFORR) Return ChildStgKeresztmetszetiForgalom objects filtered by the ADATFORR column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findBySzamlnap(string $SZAMLNAP) Return ChildStgKeresztmetszetiForgalom objects filtered by the SZAMLNAP column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByJelleg1(string $JELLEG1) Return ChildStgKeresztmetszetiForgalom objects filtered by the JELLEG1 column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByJelleg2(string $JELLEG2) Return ChildStgKeresztmetszetiForgalom objects filtered by the JELLEG2 column

\* @method ChildStgKeresztmetszetiForgalom[]|ObjectCollection findByFmegj(string $FMEGJ) Return ChildStgKeresztmetszetiForgalom objects filtered by the FMEGJ column

\* @method ChildStgKeresztmetszetiForgalom[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgKeresztmetszetiForgalomQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgKeresztmetszetiForgalomQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgKeresztmetszetiForgalom', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgKeresztmetszetiForgalomQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgKeresztmetszetiForgalomQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgKeresztmetszetiForgalomQuery) {

return $criteria;

}

$query = new ChildStgKeresztmetszetiForgalomQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgKeresztmetszetiForgalom|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgKeresztmetszetiForgalom object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgKeresztmetszetiForgalom object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgKeresztmetszetiForgalom object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgKeresztmetszetiForgalom object has no primary key');

}

/\*\*

\* Filter the query on the shapenr column

\*

\* Example usage:

\* <code>

\* $query->filterByShapenr(1234); // WHERE shapenr = 1234

\* $query->filterByShapenr(array(12, 34)); // WHERE shapenr IN (12, 34)

\* $query->filterByShapenr(array('min' => 12)); // WHERE shapenr > 12

\* </code>

\*

\* @param mixed $shapenr The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByShapenr($shapenr = null, $comparison = null)

{

if (is\_array($shapenr)) {

$useMinMax = false;

if (isset($shapenr['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_SHAPENR, $shapenr['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($shapenr['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_SHAPENR, $shapenr['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_SHAPENR, $shapenr, $comparison);

}

/\*\*

\* Filter the query on the partnr column

\*

\* Example usage:

\* <code>

\* $query->filterByPartnr(1234); // WHERE partnr = 1234

\* $query->filterByPartnr(array(12, 34)); // WHERE partnr IN (12, 34)

\* $query->filterByPartnr(array('min' => 12)); // WHERE partnr > 12

\* </code>

\*

\* @param mixed $partnr The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByPartnr($partnr = null, $comparison = null)

{

if (is\_array($partnr)) {

$useMinMax = false;

if (isset($partnr['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_PARTNR, $partnr['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($partnr['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_PARTNR, $partnr['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_PARTNR, $partnr, $comparison);

}

/\*\*

\* Filter the query on the nrparts column

\*

\* Example usage:

\* <code>

\* $query->filterByNrparts(1234); // WHERE nrparts = 1234

\* $query->filterByNrparts(array(12, 34)); // WHERE nrparts IN (12, 34)

\* $query->filterByNrparts(array('min' => 12)); // WHERE nrparts > 12

\* </code>

\*

\* @param mixed $nrparts The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByNrparts($nrparts = null, $comparison = null)

{

if (is\_array($nrparts)) {

$useMinMax = false;

if (isset($nrparts['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_NRPARTS, $nrparts['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nrparts['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_NRPARTS, $nrparts['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_NRPARTS, $nrparts, $comparison);

}

/\*\*

\* Filter the query on the pointnr column

\*

\* Example usage:

\* <code>

\* $query->filterByPointnr(1234); // WHERE pointnr = 1234

\* $query->filterByPointnr(array(12, 34)); // WHERE pointnr IN (12, 34)

\* $query->filterByPointnr(array('min' => 12)); // WHERE pointnr > 12

\* </code>

\*

\* @param mixed $pointnr The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByPointnr($pointnr = null, $comparison = null)

{

if (is\_array($pointnr)) {

$useMinMax = false;

if (isset($pointnr['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_POINTNR, $pointnr['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($pointnr['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_POINTNR, $pointnr['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_POINTNR, $pointnr, $comparison);

}

/\*\*

\* Filter the query on the nrpointS column

\*

\* Example usage:

\* <code>

\* $query->filterByNrpoints(1234); // WHERE nrpointS = 1234

\* $query->filterByNrpoints(array(12, 34)); // WHERE nrpointS IN (12, 34)

\* $query->filterByNrpoints(array('min' => 12)); // WHERE nrpointS > 12

\* </code>

\*

\* @param mixed $nrpoints The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByNrpoints($nrpoints = null, $comparison = null)

{

if (is\_array($nrpoints)) {

$useMinMax = false;

if (isset($nrpoints['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_NRPOINTS, $nrpoints['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nrpoints['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_NRPOINTS, $nrpoints['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_NRPOINTS, $nrpoints, $comparison);

}

/\*\*

\* Filter the query on the x column

\*

\* Example usage:

\* <code>

\* $query->filterByX(1234); // WHERE x = 1234

\* $query->filterByX(array(12, 34)); // WHERE x IN (12, 34)

\* $query->filterByX(array('min' => 12)); // WHERE x > 12

\* </code>

\*

\* @param mixed $x The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByX($x = null, $comparison = null)

{

if (is\_array($x)) {

$useMinMax = false;

if (isset($x['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_X, $x['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($x['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_X, $x['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_X, $x, $comparison);

}

/\*\*

\* Filter the query on the y column

\*

\* Example usage:

\* <code>

\* $query->filterByY(1234); // WHERE y = 1234

\* $query->filterByY(array(12, 34)); // WHERE y IN (12, 34)

\* $query->filterByY(array('min' => 12)); // WHERE y > 12

\* </code>

\*

\* @param mixed $y The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByY($y = null, $comparison = null)

{

if (is\_array($y)) {

$useMinMax = false;

if (isset($y['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_Y, $y['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($y['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_Y, $y['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_Y, $y, $comparison);

}

/\*\*

\* Filter the query on the measure column

\*

\* Example usage:

\* <code>

\* $query->filterByMeasure(1234); // WHERE measure = 1234

\* $query->filterByMeasure(array(12, 34)); // WHERE measure IN (12, 34)

\* $query->filterByMeasure(array('min' => 12)); // WHERE measure > 12

\* </code>

\*

\* @param mixed $measure The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByMeasure($measure = null, $comparison = null)

{

if (is\_array($measure)) {

$useMinMax = false;

if (isset($measure['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_MEASURE, $measure['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($measure['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_MEASURE, $measure['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_MEASURE, $measure, $comparison);

}

/\*\*

\* Filter the query on the kszam column

\*

\* Example usage:

\* <code>

\* $query->filterByKszam('fooValue'); // WHERE kszam = 'fooValue'

\* $query->filterByKszam('%fooValue%'); // WHERE kszam LIKE '%fooValue%'

\* </code>

\*

\* @param string $kszam The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByKszam($kszam = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($kszam)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $kszam)) {

$kszam = str\_replace('\*', '%', $kszam);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_KSZAM, $kszam, $comparison);

}

/\*\*

\* Filter the query on the pkod column

\*

\* Example usage:

\* <code>

\* $query->filterByPkod(1234); // WHERE pkod = 1234

\* $query->filterByPkod(array(12, 34)); // WHERE pkod IN (12, 34)

\* $query->filterByPkod(array('min' => 12)); // WHERE pkod > 12

\* </code>

\*

\* @param mixed $pkod The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByPkod($pkod = null, $comparison = null)

{

if (is\_array($pkod)) {

$useMinMax = false;

if (isset($pkod['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_PKOD, $pkod['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($pkod['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_PKOD, $pkod['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_PKOD, $pkod, $comparison);

}

/\*\*

\* Filter the query on the kkod column

\*

\* Example usage:

\* <code>

\* $query->filterByKkod('fooValue'); // WHERE kkod = 'fooValue'

\* $query->filterByKkod('%fooValue%'); // WHERE kkod LIKE '%fooValue%'

\* </code>

\*

\* @param string $kkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByKkod($kkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($kkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $kkod)) {

$kkod = str\_replace('\*', '%', $kkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_KKOD, $kkod, $comparison);

}

/\*\*

\* Filter the query on the vvkod column

\*

\* Example usage:

\* <code>

\* $query->filterByVvkod('fooValue'); // WHERE vvkod = 'fooValue'

\* $query->filterByVvkod('%fooValue%'); // WHERE vvkod LIKE '%fooValue%'

\* </code>

\*

\* @param string $vvkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByVvkod($vvkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($vvkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $vvkod)) {

$vvkod = str\_replace('\*', '%', $vvkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_VVKOD, $vvkod, $comparison);

}

/\*\*

\* Filter the query on the kszelv column

\*

\* Example usage:

\* <code>

\* $query->filterByKszelv('fooValue'); // WHERE kszelv = 'fooValue'

\* $query->filterByKszelv('%fooValue%'); // WHERE kszelv LIKE '%fooValue%'

\* </code>

\*

\* @param string $kszelv The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByKszelv($kszelv = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($kszelv)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $kszelv)) {

$kszelv = str\_replace('\*', '%', $kszelv);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_KSZELV, $kszelv, $comparison);

}

/\*\*

\* Filter the query on the vszelv column

\*

\* Example usage:

\* <code>

\* $query->filterByVszelv('fooValue'); // WHERE vszelv = 'fooValue'

\* $query->filterByVszelv('%fooValue%'); // WHERE vszelv LIKE '%fooValue%'

\* </code>

\*

\* @param string $vszelv The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByVszelv($vszelv = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($vszelv)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $vszelv)) {

$vszelv = str\_replace('\*', '%', $vszelv);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_VSZELV, $vszelv, $comparison);

}

/\*\*

\* Filter the query on the RSHOSSZ column

\*

\* Example usage:

\* <code>

\* $query->filterByRshossz(1234); // WHERE RSHOSSZ = 1234

\* $query->filterByRshossz(array(12, 34)); // WHERE RSHOSSZ IN (12, 34)

\* $query->filterByRshossz(array('min' => 12)); // WHERE RSHOSSZ > 12

\* </code>

\*

\* @param mixed $rshossz The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByRshossz($rshossz = null, $comparison = null)

{

if (is\_array($rshossz)) {

$useMinMax = false;

if (isset($rshossz['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_RSHOSSZ, $rshossz['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($rshossz['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_RSHOSSZ, $rshossz['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_RSHOSSZ, $rshossz, $comparison);

}

/\*\*

\* Filter the query on the ANF column

\*

\* Example usage:

\* <code>

\* $query->filterByAnf(1234); // WHERE ANF = 1234

\* $query->filterByAnf(array(12, 34)); // WHERE ANF IN (12, 34)

\* $query->filterByAnf(array('min' => 12)); // WHERE ANF > 12

\* </code>

\*

\* @param mixed $anf The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByAnf($anf = null, $comparison = null)

{

if (is\_array($anf)) {

$useMinMax = false;

if (isset($anf['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_ANF, $anf['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($anf['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_ANF, $anf['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_ANF, $anf, $comparison);

}

/\*\*

\* Filter the query on the ANET column

\*

\* Example usage:

\* <code>

\* $query->filterByAnet(1234); // WHERE ANET = 1234

\* $query->filterByAnet(array(12, 34)); // WHERE ANET IN (12, 34)

\* $query->filterByAnet(array('min' => 12)); // WHERE ANET > 12

\* </code>

\*

\* @param mixed $anet The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByAnet($anet = null, $comparison = null)

{

if (is\_array($anet)) {

$useMinMax = false;

if (isset($anet['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_ANET, $anet['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($anet['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_ANET, $anet['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_ANET, $anet, $comparison);

}

/\*\*

\* Filter the query on the MOF column

\*

\* Example usage:

\* <code>

\* $query->filterByMof(1234); // WHERE MOF = 1234

\* $query->filterByMof(array(12, 34)); // WHERE MOF IN (12, 34)

\* $query->filterByMof(array('min' => 12)); // WHERE MOF > 12

\* </code>

\*

\* @param mixed $mof The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByMof($mof = null, $comparison = null)

{

if (is\_array($mof)) {

$useMinMax = false;

if (isset($mof['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_MOF, $mof['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($mof['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_MOF, $mof['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_MOF, $mof, $comparison);

}

/\*\*

\* Filter the query on the ONGJ column

\*

\* Example usage:

\* <code>

\* $query->filterByOngj(1234); // WHERE ONGJ = 1234

\* $query->filterByOngj(array(12, 34)); // WHERE ONGJ IN (12, 34)

\* $query->filterByOngj(array('min' => 12)); // WHERE ONGJ > 12

\* </code>

\*

\* @param mixed $ongj The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByOngj($ongj = null, $comparison = null)

{

if (is\_array($ongj)) {

$useMinMax = false;

if (isset($ongj['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_ONGJ, $ongj['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($ongj['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_ONGJ, $ongj['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_ONGJ, $ongj, $comparison);

}

/\*\*

\* Filter the query on the OJ column

\*

\* Example usage:

\* <code>

\* $query->filterByOj(1234); // WHERE OJ = 1234

\* $query->filterByOj(array(12, 34)); // WHERE OJ IN (12, 34)

\* $query->filterByOj(array('min' => 12)); // WHERE OJ > 12

\* </code>

\*

\* @param mixed $oj The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByOj($oj = null, $comparison = null)

{

if (is\_array($oj)) {

$useMinMax = false;

if (isset($oj['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_OJ, $oj['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($oj['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_OJ, $oj['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_OJ, $oj, $comparison);

}

/\*\*

\* Filter the query on the OMOT column

\*

\* Example usage:

\* <code>

\* $query->filterByOmot(1234); // WHERE OMOT = 1234

\* $query->filterByOmot(array(12, 34)); // WHERE OMOT IN (12, 34)

\* $query->filterByOmot(array('min' => 12)); // WHERE OMOT > 12

\* </code>

\*

\* @param mixed $omot The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByOmot($omot = null, $comparison = null)

{

if (is\_array($omot)) {

$useMinMax = false;

if (isset($omot['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_OMOT, $omot['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($omot['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_OMOT, $omot['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_OMOT, $omot, $comparison);

}

/\*\*

\* Filter the query on the EV column

\*

\* Example usage:

\* <code>

\* $query->filterByEv(1234); // WHERE EV = 1234

\* $query->filterByEv(array(12, 34)); // WHERE EV IN (12, 34)

\* $query->filterByEv(array('min' => 12)); // WHERE EV > 12

\* </code>

\*

\* @param mixed $ev The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByEv($ev = null, $comparison = null)

{

if (is\_array($ev)) {

$useMinMax = false;

if (isset($ev['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_EV, $ev['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($ev['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_EV, $ev['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_EV, $ev, $comparison);

}

/\*\*

\* Filter the query on the ASZ column

\*

\* Example usage:

\* <code>

\* $query->filterByAsz(1234); // WHERE ASZ = 1234

\* $query->filterByAsz(array(12, 34)); // WHERE ASZ IN (12, 34)

\* $query->filterByAsz(array('min' => 12)); // WHERE ASZ > 12

\* </code>

\*

\* @param mixed $asz The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByAsz($asz = null, $comparison = null)

{

if (is\_array($asz)) {

$useMinMax = false;

if (isset($asz['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_ASZ, $asz['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($asz['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_ASZ, $asz['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_ASZ, $asz, $comparison);

}

/\*\*

\* Filter the query on the BUSZCS column

\*

\* Example usage:

\* <code>

\* $query->filterByBuszcs(1234); // WHERE BUSZCS = 1234

\* $query->filterByBuszcs(array(12, 34)); // WHERE BUSZCS IN (12, 34)

\* $query->filterByBuszcs(array('min' => 12)); // WHERE BUSZCS > 12

\* </code>

\*

\* @param mixed $buszcs The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByBuszcs($buszcs = null, $comparison = null)

{

if (is\_array($buszcs)) {

$useMinMax = false;

if (isset($buszcs['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_BUSZCS, $buszcs['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($buszcs['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_BUSZCS, $buszcs['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_BUSZCS, $buszcs, $comparison);

}

/\*\*

\* Filter the query on the BUSZE column

\*

\* Example usage:

\* <code>

\* $query->filterByBusze(1234); // WHERE BUSZE = 1234

\* $query->filterByBusze(array(12, 34)); // WHERE BUSZE IN (12, 34)

\* $query->filterByBusze(array('min' => 12)); // WHERE BUSZE > 12

\* </code>

\*

\* @param mixed $busze The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByBusze($busze = null, $comparison = null)

{

if (is\_array($busze)) {

$useMinMax = false;

if (isset($busze['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_BUSZE, $busze['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($busze['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_BUSZE, $busze['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_BUSZE, $busze, $comparison);

}

/\*\*

\* Filter the query on the OBUSZ column

\*

\* Example usage:

\* <code>

\* $query->filterByObusz('fooValue'); // WHERE OBUSZ = 'fooValue'

\* $query->filterByObusz('%fooValue%'); // WHERE OBUSZ LIKE '%fooValue%'

\* </code>

\*

\* @param string $obusz The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByObusz($obusz = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($obusz)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $obusz)) {

$obusz = str\_replace('\*', '%', $obusz);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_OBUSZ, $obusz, $comparison);

}

/\*\*

\* Filter the query on the NYSZER column

\*

\* Example usage:

\* <code>

\* $query->filterByNyszer('fooValue'); // WHERE NYSZER = 'fooValue'

\* $query->filterByNyszer('%fooValue%'); // WHERE NYSZER LIKE '%fooValue%'

\* </code>

\*

\* @param string $nyszer The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByNyszer($nyszer = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($nyszer)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $nyszer)) {

$nyszer = str\_replace('\*', '%', $nyszer);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_NYSZER, $nyszer, $comparison);

}

/\*\*

\* Filter the query on the POTKTGK column

\*

\* Example usage:

\* <code>

\* $query->filterByPotktgk('fooValue'); // WHERE POTKTGK = 'fooValue'

\* $query->filterByPotktgk('%fooValue%'); // WHERE POTKTGK LIKE '%fooValue%'

\* </code>

\*

\* @param string $potktgk The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByPotktgk($potktgk = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($potktgk)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $potktgk)) {

$potktgk = str\_replace('\*', '%', $potktgk);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_POTKTGK, $potktgk, $comparison);

}

/\*\*

\* Filter the query on the KTGK column

\*

\* Example usage:

\* <code>

\* $query->filterByKtgk('fooValue'); // WHERE KTGK = 'fooValue'

\* $query->filterByKtgk('%fooValue%'); // WHERE KTGK LIKE '%fooValue%'

\* </code>

\*

\* @param string $ktgk The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByKtgk($ktgk = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($ktgk)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $ktgk)) {

$ktgk = str\_replace('\*', '%', $ktgk);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_KTGK, $ktgk, $comparison);

}

/\*\*

\* Filter the query on the NTGK column

\*

\* Example usage:

\* <code>

\* $query->filterByNtgk('fooValue'); // WHERE NTGK = 'fooValue'

\* $query->filterByNtgk('%fooValue%'); // WHERE NTGK LIKE '%fooValue%'

\* </code>

\*

\* @param string $ntgk The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByNtgk($ntgk = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($ntgk)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $ntgk)) {

$ntgk = str\_replace('\*', '%', $ntgk);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_NTGK, $ntgk, $comparison);

}

/\*\*

\* Filter the query on the KNTGK column

\*

\* Example usage:

\* <code>

\* $query->filterByKntgk('fooValue'); // WHERE KNTGK = 'fooValue'

\* $query->filterByKntgk('%fooValue%'); // WHERE KNTGK LIKE '%fooValue%'

\* </code>

\*

\* @param string $kntgk The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByKntgk($kntgk = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($kntgk)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $kntgk)) {

$kntgk = str\_replace('\*', '%', $kntgk);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_KNTGK, $kntgk, $comparison);

}

/\*\*

\* Filter the query on the OTGK column

\*

\* Example usage:

\* <code>

\* $query->filterByOtgk('fooValue'); // WHERE OTGK = 'fooValue'

\* $query->filterByOtgk('%fooValue%'); // WHERE OTGK LIKE '%fooValue%'

\* </code>

\*

\* @param string $otgk The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByOtgk($otgk = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($otgk)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $otgk)) {

$otgk = str\_replace('\*', '%', $otgk);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_OTGK, $otgk, $comparison);

}

/\*\*

\* Filter the query on the SZGK column

\*

\* Example usage:

\* <code>

\* $query->filterBySzgk('fooValue'); // WHERE SZGK = 'fooValue'

\* $query->filterBySzgk('%fooValue%'); // WHERE SZGK LIKE '%fooValue%'

\* </code>

\*

\* @param string $szgk The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterBySzgk($szgk = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($szgk)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $szgk)) {

$szgk = str\_replace('\*', '%', $szgk);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_SZGK, $szgk, $comparison);

}

/\*\*

\* Filter the query on the OSZGK column

\*

\* Example usage:

\* <code>

\* $query->filterByOszgk('fooValue'); // WHERE OSZGK = 'fooValue'

\* $query->filterByOszgk('%fooValue%'); // WHERE OSZGK LIKE '%fooValue%'

\* </code>

\*

\* @param string $oszgk The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByOszgk($oszgk = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($oszgk)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $oszgk)) {

$oszgk = str\_replace('\*', '%', $oszgk);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_OSZGK, $oszgk, $comparison);

}

/\*\*

\* Filter the query on the MKP column

\*

\* Example usage:

\* <code>

\* $query->filterByMkp('fooValue'); // WHERE MKP = 'fooValue'

\* $query->filterByMkp('%fooValue%'); // WHERE MKP LIKE '%fooValue%'

\* </code>

\*

\* @param string $mkp The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByMkp($mkp = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($mkp)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $mkp)) {

$mkp = str\_replace('\*', '%', $mkp);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_MKP, $mkp, $comparison);

}

/\*\*

\* Filter the query on the KPF column

\*

\* Example usage:

\* <code>

\* $query->filterByKpf('fooValue'); // WHERE KPF = 'fooValue'

\* $query->filterByKpf('%fooValue%'); // WHERE KPF LIKE '%fooValue%'

\* </code>

\*

\* @param string $kpf The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByKpf($kpf = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($kpf)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $kpf)) {

$kpf = str\_replace('\*', '%', $kpf);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_KPF, $kpf, $comparison);

}

/\*\*

\* Filter the query on the LASSU column

\*

\* Example usage:

\* <code>

\* $query->filterByLassu('fooValue'); // WHERE LASSU = 'fooValue'

\* $query->filterByLassu('%fooValue%'); // WHERE LASSU LIKE '%fooValue%'

\* </code>

\*

\* @param string $lassu The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByLassu($lassu = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($lassu)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $lassu)) {

$lassu = str\_replace('\*', '%', $lassu);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_LASSU, $lassu, $comparison);

}

/\*\*

\* Filter the query on the SPEC column

\*

\* Example usage:

\* <code>

\* $query->filterBySpec('fooValue'); // WHERE SPEC = 'fooValue'

\* $query->filterBySpec('%fooValue%'); // WHERE SPEC LIKE '%fooValue%'

\* </code>

\*

\* @param string $spec The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterBySpec($spec = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($spec)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $spec)) {

$spec = str\_replace('\*', '%', $spec);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_SPEC, $spec, $comparison);

}

/\*\*

\* Filter the query on the FMEGB column

\*

\* Example usage:

\* <code>

\* $query->filterByFmegb('fooValue'); // WHERE FMEGB = 'fooValue'

\* $query->filterByFmegb('%fooValue%'); // WHERE FMEGB LIKE '%fooValue%'

\* </code>

\*

\* @param string $fmegb The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByFmegb($fmegb = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($fmegb)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $fmegb)) {

$fmegb = str\_replace('\*', '%', $fmegb);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_FMEGB, $fmegb, $comparison);

}

/\*\*

\* Filter the query on the ADATFORR column

\*

\* Example usage:

\* <code>

\* $query->filterByAdatforr('fooValue'); // WHERE ADATFORR = 'fooValue'

\* $query->filterByAdatforr('%fooValue%'); // WHERE ADATFORR LIKE '%fooValue%'

\* </code>

\*

\* @param string $adatforr The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByAdatforr($adatforr = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($adatforr)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $adatforr)) {

$adatforr = str\_replace('\*', '%', $adatforr);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_ADATFORR, $adatforr, $comparison);

}

/\*\*

\* Filter the query on the SZAMLNAP column

\*

\* Example usage:

\* <code>

\* $query->filterBySzamlnap('fooValue'); // WHERE SZAMLNAP = 'fooValue'

\* $query->filterBySzamlnap('%fooValue%'); // WHERE SZAMLNAP LIKE '%fooValue%'

\* </code>

\*

\* @param string $szamlnap The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterBySzamlnap($szamlnap = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($szamlnap)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $szamlnap)) {

$szamlnap = str\_replace('\*', '%', $szamlnap);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_SZAMLNAP, $szamlnap, $comparison);

}

/\*\*

\* Filter the query on the JELLEG1 column

\*

\* Example usage:

\* <code>

\* $query->filterByJelleg1('fooValue'); // WHERE JELLEG1 = 'fooValue'

\* $query->filterByJelleg1('%fooValue%'); // WHERE JELLEG1 LIKE '%fooValue%'

\* </code>

\*

\* @param string $jelleg1 The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByJelleg1($jelleg1 = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($jelleg1)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $jelleg1)) {

$jelleg1 = str\_replace('\*', '%', $jelleg1);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_JELLEG1, $jelleg1, $comparison);

}

/\*\*

\* Filter the query on the JELLEG2 column

\*

\* Example usage:

\* <code>

\* $query->filterByJelleg2('fooValue'); // WHERE JELLEG2 = 'fooValue'

\* $query->filterByJelleg2('%fooValue%'); // WHERE JELLEG2 LIKE '%fooValue%'

\* </code>

\*

\* @param string $jelleg2 The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByJelleg2($jelleg2 = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($jelleg2)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $jelleg2)) {

$jelleg2 = str\_replace('\*', '%', $jelleg2);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_JELLEG2, $jelleg2, $comparison);

}

/\*\*

\* Filter the query on the FMEGJ column

\*

\* Example usage:

\* <code>

\* $query->filterByFmegj('fooValue'); // WHERE FMEGJ = 'fooValue'

\* $query->filterByFmegj('%fooValue%'); // WHERE FMEGJ LIKE '%fooValue%'

\* </code>

\*

\* @param string $fmegj The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function filterByFmegj($fmegj = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($fmegj)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $fmegj)) {

$fmegj = str\_replace('\*', '%', $fmegj);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomTableMap::COL\_FMEGJ, $fmegj, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgKeresztmetszetiForgalom $stgKeresztmetszetiForgalom Object to remove from the list of results

\*

\* @return $this|ChildStgKeresztmetszetiForgalomQuery The current query, for fluid interface

\*/

public function prune($stgKeresztmetszetiForgalom = null)

{

if ($stgKeresztmetszetiForgalom) {

throw new LogicException('StgKeresztmetszetiForgalom object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_keresztmetszeti\_forgalom table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKeresztmetszetiForgalomTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgKeresztmetszetiForgalomTableMap::clearInstancePool();

StgKeresztmetszetiForgalomTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKeresztmetszetiForgalomTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgKeresztmetszetiForgalomTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgKeresztmetszetiForgalomTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgKeresztmetszetiForgalomTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgKeresztmetszetiForgalomQuery

#### StgKeresztmetszetiForgalomV1.php

<?php

namespace Base;

use \StgKeresztmetszetiForgalomV1Query as ChildStgKeresztmetszetiForgalomV1Query;

use \Exception;

use \PDO;

use Map\StgKeresztmetszetiForgalomV1TableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgKeresztmetszetiForgalomV1 implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgKeresztmetszetiForgalomV1TableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the partnr field.

\* @var int

\*/

protected $partnr;

/\*\*

\* The value for the nrparts field.

\* @var string

\*/

protected $nrparts;

/\*\*

\* The value for the pointnr field.

\* @var string

\*/

protected $pointnr;

/\*\*

\* The value for the nrpoints field.

\* @var string

\*/

protected $nrpoints;

/\*\*

\* The value for the x field.

\* @var string

\*/

protected $x;

/\*\*

\* The value for the y field.

\* @var string

\*/

protected $y;

/\*\*

\* The value for the measure field.

\* @var string

\*/

protected $measure;

/\*\*

\* The value for the kezd field.

\* @var string

\*/

protected $kezd;

/\*\*

\* The value for the vege field.

\* @var string

\*/

protected $vege;

/\*\*

\* The value for the kszam field.

\* @var string

\*/

protected $kszam;

/\*\*

\* The value for the pkod field.

\* @var string

\*/

protected $pkod;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgKeresztmetszetiForgalomV1 object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgKeresztmetszetiForgalomV1</code> instance. If

\* <code>obj</code> is an instance of <code>StgKeresztmetszetiForgalomV1</code>, delegates to

\* <code>equals(StgKeresztmetszetiForgalomV1)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgKeresztmetszetiForgalomV1 The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [partnr] column value.

\*

\* @return int

\*/

public function getPartnr()

{

return $this->partnr;

}

/\*\*

\* Get the [nrparts] column value.

\*

\* @return string

\*/

public function getNrparts()

{

return $this->nrparts;

}

/\*\*

\* Get the [pointnr] column value.

\*

\* @return string

\*/

public function getPointnr()

{

return $this->pointnr;

}

/\*\*

\* Get the [nrpoints] column value.

\*

\* @return string

\*/

public function getNrpoints()

{

return $this->nrpoints;

}

/\*\*

\* Get the [x] column value.

\*

\* @return string

\*/

public function getX()

{

return $this->x;

}

/\*\*

\* Get the [y] column value.

\*

\* @return string

\*/

public function getY()

{

return $this->y;

}

/\*\*

\* Get the [measure] column value.

\*

\* @return string

\*/

public function getMeasure()

{

return $this->measure;

}

/\*\*

\* Get the [kezd] column value.

\*

\* @return string

\*/

public function getKezd()

{

return $this->kezd;

}

/\*\*

\* Get the [vege] column value.

\*

\* @return string

\*/

public function getVege()

{

return $this->vege;

}

/\*\*

\* Get the [kszam] column value.

\*

\* @return string

\*/

public function getKszam()

{

return $this->kszam;

}

/\*\*

\* Get the [pkod] column value.

\*

\* @return string

\*/

public function getPkod()

{

return $this->pkod;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgKeresztmetszetiForgalomV1TableMap::translateFieldName('Partnr', TableMap::TYPE\_PHPNAME, $indexType)];

$this->partnr = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgKeresztmetszetiForgalomV1TableMap::translateFieldName('Nrparts', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nrparts = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgKeresztmetszetiForgalomV1TableMap::translateFieldName('Pointnr', TableMap::TYPE\_PHPNAME, $indexType)];

$this->pointnr = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgKeresztmetszetiForgalomV1TableMap::translateFieldName('Nrpoints', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nrpoints = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgKeresztmetszetiForgalomV1TableMap::translateFieldName('X', TableMap::TYPE\_PHPNAME, $indexType)];

$this->x = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 5 + $startcol : StgKeresztmetszetiForgalomV1TableMap::translateFieldName('Y', TableMap::TYPE\_PHPNAME, $indexType)];

$this->y = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 6 + $startcol : StgKeresztmetszetiForgalomV1TableMap::translateFieldName('Measure', TableMap::TYPE\_PHPNAME, $indexType)];

$this->measure = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 7 + $startcol : StgKeresztmetszetiForgalomV1TableMap::translateFieldName('Kezd', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kezd = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 8 + $startcol : StgKeresztmetszetiForgalomV1TableMap::translateFieldName('Vege', TableMap::TYPE\_PHPNAME, $indexType)];

$this->vege = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 9 + $startcol : StgKeresztmetszetiForgalomV1TableMap::translateFieldName('Kszam', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kszam = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 10 + $startcol : StgKeresztmetszetiForgalomV1TableMap::translateFieldName('Pkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->pkod = (null !== $col) ? (string) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 11; // 11 = StgKeresztmetszetiForgalomV1TableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgKeresztmetszetiForgalomV1'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [partnr] column.

\*

\* @param int $v new value

\* @return $this|\StgKeresztmetszetiForgalomV1 The current object (for fluent API support)

\*/

public function setPartnr($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->partnr !== $v) {

$this->partnr = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomV1TableMap::COL\_PARTNR] = true;

}

return $this;

} // setPartnr()

/\*\*

\* Set the value of [nrparts] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalomV1 The current object (for fluent API support)

\*/

public function setNrparts($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->nrparts !== $v) {

$this->nrparts = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomV1TableMap::COL\_NRPARTS] = true;

}

return $this;

} // setNrparts()

/\*\*

\* Set the value of [pointnr] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalomV1 The current object (for fluent API support)

\*/

public function setPointnr($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->pointnr !== $v) {

$this->pointnr = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomV1TableMap::COL\_POINTNR] = true;

}

return $this;

} // setPointnr()

/\*\*

\* Set the value of [nrpoints] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalomV1 The current object (for fluent API support)

\*/

public function setNrpoints($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->nrpoints !== $v) {

$this->nrpoints = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomV1TableMap::COL\_NRPOINTS] = true;

}

return $this;

} // setNrpoints()

/\*\*

\* Set the value of [x] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalomV1 The current object (for fluent API support)

\*/

public function setX($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->x !== $v) {

$this->x = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomV1TableMap::COL\_X] = true;

}

return $this;

} // setX()

/\*\*

\* Set the value of [y] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalomV1 The current object (for fluent API support)

\*/

public function setY($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->y !== $v) {

$this->y = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomV1TableMap::COL\_Y] = true;

}

return $this;

} // setY()

/\*\*

\* Set the value of [measure] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalomV1 The current object (for fluent API support)

\*/

public function setMeasure($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->measure !== $v) {

$this->measure = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomV1TableMap::COL\_MEASURE] = true;

}

return $this;

} // setMeasure()

/\*\*

\* Set the value of [kezd] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalomV1 The current object (for fluent API support)

\*/

public function setKezd($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->kezd !== $v) {

$this->kezd = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomV1TableMap::COL\_KEZD] = true;

}

return $this;

} // setKezd()

/\*\*

\* Set the value of [vege] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalomV1 The current object (for fluent API support)

\*/

public function setVege($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->vege !== $v) {

$this->vege = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomV1TableMap::COL\_VEGE] = true;

}

return $this;

} // setVege()

/\*\*

\* Set the value of [kszam] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalomV1 The current object (for fluent API support)

\*/

public function setKszam($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->kszam !== $v) {

$this->kszam = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomV1TableMap::COL\_KSZAM] = true;

}

return $this;

} // setKszam()

/\*\*

\* Set the value of [pkod] column.

\*

\* @param string $v new value

\* @return $this|\StgKeresztmetszetiForgalomV1 The current object (for fluent API support)

\*/

public function setPkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->pkod !== $v) {

$this->pkod = $v;

$this->modifiedColumns[StgKeresztmetszetiForgalomV1TableMap::COL\_PKOD] = true;

}

return $this;

} // setPkod()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgKeresztmetszetiForgalomV1TableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgKeresztmetszetiForgalomV1Query::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgKeresztmetszetiForgalomV1::setDeleted()

\* @see StgKeresztmetszetiForgalomV1::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKeresztmetszetiForgalomV1TableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgKeresztmetszetiForgalomV1Query::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKeresztmetszetiForgalomV1TableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgKeresztmetszetiForgalomV1TableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_PARTNR)) {

$modifiedColumns[':p' . $index++] = 'PARTNR';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_NRPARTS)) {

$modifiedColumns[':p' . $index++] = 'NRPARTS';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_POINTNR)) {

$modifiedColumns[':p' . $index++] = 'POINTNR';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_NRPOINTS)) {

$modifiedColumns[':p' . $index++] = 'NRPOINTS';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_X)) {

$modifiedColumns[':p' . $index++] = 'X';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_Y)) {

$modifiedColumns[':p' . $index++] = 'Y';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_MEASURE)) {

$modifiedColumns[':p' . $index++] = 'MEASURE';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_KEZD)) {

$modifiedColumns[':p' . $index++] = 'KEZD';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_VEGE)) {

$modifiedColumns[':p' . $index++] = 'VEGE';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_KSZAM)) {

$modifiedColumns[':p' . $index++] = 'KSZAM';

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_PKOD)) {

$modifiedColumns[':p' . $index++] = 'PKOD';

}

$sql = sprintf(

'INSERT INTO stg\_keresztmetszeti\_forgalom\_v1 (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'PARTNR':

$stmt->bindValue($identifier, $this->partnr, PDO::PARAM\_INT);

break;

case 'NRPARTS':

$stmt->bindValue($identifier, $this->nrparts, PDO::PARAM\_STR);

break;

case 'POINTNR':

$stmt->bindValue($identifier, $this->pointnr, PDO::PARAM\_STR);

break;

case 'NRPOINTS':

$stmt->bindValue($identifier, $this->nrpoints, PDO::PARAM\_STR);

break;

case 'X':

$stmt->bindValue($identifier, $this->x, PDO::PARAM\_STR);

break;

case 'Y':

$stmt->bindValue($identifier, $this->y, PDO::PARAM\_STR);

break;

case 'MEASURE':

$stmt->bindValue($identifier, $this->measure, PDO::PARAM\_STR);

break;

case 'KEZD':

$stmt->bindValue($identifier, $this->kezd, PDO::PARAM\_STR);

break;

case 'VEGE':

$stmt->bindValue($identifier, $this->vege, PDO::PARAM\_STR);

break;

case 'KSZAM':

$stmt->bindValue($identifier, $this->kszam, PDO::PARAM\_STR);

break;

case 'PKOD':

$stmt->bindValue($identifier, $this->pkod, PDO::PARAM\_STR);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgKeresztmetszetiForgalomV1TableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getPartnr();

break;

case 1:

return $this->getNrparts();

break;

case 2:

return $this->getPointnr();

break;

case 3:

return $this->getNrpoints();

break;

case 4:

return $this->getX();

break;

case 5:

return $this->getY();

break;

case 6:

return $this->getMeasure();

break;

case 7:

return $this->getKezd();

break;

case 8:

return $this->getVege();

break;

case 9:

return $this->getKszam();

break;

case 10:

return $this->getPkod();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgKeresztmetszetiForgalomV1'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgKeresztmetszetiForgalomV1'][$this->getPrimaryKey()] = true;

$keys = StgKeresztmetszetiForgalomV1TableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getPartnr(),

$keys[1] => $this->getNrparts(),

$keys[2] => $this->getPointnr(),

$keys[3] => $this->getNrpoints(),

$keys[4] => $this->getX(),

$keys[5] => $this->getY(),

$keys[6] => $this->getMeasure(),

$keys[7] => $this->getKezd(),

$keys[8] => $this->getVege(),

$keys[9] => $this->getKszam(),

$keys[10] => $this->getPkod(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgKeresztmetszetiForgalomV1

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgKeresztmetszetiForgalomV1TableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgKeresztmetszetiForgalomV1

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setPartnr($value);

break;

case 1:

$this->setNrparts($value);

break;

case 2:

$this->setPointnr($value);

break;

case 3:

$this->setNrpoints($value);

break;

case 4:

$this->setX($value);

break;

case 5:

$this->setY($value);

break;

case 6:

$this->setMeasure($value);

break;

case 7:

$this->setKezd($value);

break;

case 8:

$this->setVege($value);

break;

case 9:

$this->setKszam($value);

break;

case 10:

$this->setPkod($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgKeresztmetszetiForgalomV1TableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setPartnr($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setNrparts($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setPointnr($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setNrpoints($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setX($arr[$keys[4]]);

}

if (array\_key\_exists($keys[5], $arr)) {

$this->setY($arr[$keys[5]]);

}

if (array\_key\_exists($keys[6], $arr)) {

$this->setMeasure($arr[$keys[6]]);

}

if (array\_key\_exists($keys[7], $arr)) {

$this->setKezd($arr[$keys[7]]);

}

if (array\_key\_exists($keys[8], $arr)) {

$this->setVege($arr[$keys[8]]);

}

if (array\_key\_exists($keys[9], $arr)) {

$this->setKszam($arr[$keys[9]]);

}

if (array\_key\_exists($keys[10], $arr)) {

$this->setPkod($arr[$keys[10]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgKeresztmetszetiForgalomV1 The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgKeresztmetszetiForgalomV1TableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_PARTNR)) {

$criteria->add(StgKeresztmetszetiForgalomV1TableMap::COL\_PARTNR, $this->partnr);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_NRPARTS)) {

$criteria->add(StgKeresztmetszetiForgalomV1TableMap::COL\_NRPARTS, $this->nrparts);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_POINTNR)) {

$criteria->add(StgKeresztmetszetiForgalomV1TableMap::COL\_POINTNR, $this->pointnr);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_NRPOINTS)) {

$criteria->add(StgKeresztmetszetiForgalomV1TableMap::COL\_NRPOINTS, $this->nrpoints);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_X)) {

$criteria->add(StgKeresztmetszetiForgalomV1TableMap::COL\_X, $this->x);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_Y)) {

$criteria->add(StgKeresztmetszetiForgalomV1TableMap::COL\_Y, $this->y);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_MEASURE)) {

$criteria->add(StgKeresztmetszetiForgalomV1TableMap::COL\_MEASURE, $this->measure);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_KEZD)) {

$criteria->add(StgKeresztmetszetiForgalomV1TableMap::COL\_KEZD, $this->kezd);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_VEGE)) {

$criteria->add(StgKeresztmetszetiForgalomV1TableMap::COL\_VEGE, $this->vege);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_KSZAM)) {

$criteria->add(StgKeresztmetszetiForgalomV1TableMap::COL\_KSZAM, $this->kszam);

}

if ($this->isColumnModified(StgKeresztmetszetiForgalomV1TableMap::COL\_PKOD)) {

$criteria->add(StgKeresztmetszetiForgalomV1TableMap::COL\_PKOD, $this->pkod);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

$criteria = new Criteria(StgKeresztmetszetiForgalomV1TableMap::DATABASE\_NAME);

$criteria->add(StgKeresztmetszetiForgalomV1TableMap::COL\_PARTNR, $this->partnr);

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = null !== $this->getPartnr();

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns the primary key for this object (row).

\* @return int

\*/

public function getPrimaryKey()

{

return $this->getPartnr();

}

/\*\*

\* Generic method to set the primary key (partnr column).

\*

\* @param int $key Primary key.

\* @return void

\*/

public function setPrimaryKey($key)

{

$this->setPartnr($key);

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return null === $this->getPartnr();

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgKeresztmetszetiForgalomV1 (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setPartnr($this->getPartnr());

$copyObj->setNrparts($this->getNrparts());

$copyObj->setPointnr($this->getPointnr());

$copyObj->setNrpoints($this->getNrpoints());

$copyObj->setX($this->getX());

$copyObj->setY($this->getY());

$copyObj->setMeasure($this->getMeasure());

$copyObj->setKezd($this->getKezd());

$copyObj->setVege($this->getVege());

$copyObj->setKszam($this->getKszam());

$copyObj->setPkod($this->getPkod());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgKeresztmetszetiForgalomV1 Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->partnr = null;

$this->nrparts = null;

$this->pointnr = null;

$this->nrpoints = null;

$this->x = null;

$this->y = null;

$this->measure = null;

$this->kezd = null;

$this->vege = null;

$this->kszam = null;

$this->pkod = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgKeresztmetszetiForgalomV1TableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgKeresztmetszetiForgalomV1Query.php

<?php

namespace Base;

use \StgKeresztmetszetiForgalomV1 as ChildStgKeresztmetszetiForgalomV1;

use \StgKeresztmetszetiForgalomV1Query as ChildStgKeresztmetszetiForgalomV1Query;

use \Exception;

use \PDO;

use Map\StgKeresztmetszetiForgalomV1TableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_keresztmetszeti\_forgalom\_v1' table.

\*

\*

\*

\* @method ChildStgKeresztmetszetiForgalomV1Query orderByPartnr($order = Criteria::ASC) Order by the partnr column

\* @method ChildStgKeresztmetszetiForgalomV1Query orderByNrparts($order = Criteria::ASC) Order by the nrparts column

\* @method ChildStgKeresztmetszetiForgalomV1Query orderByPointnr($order = Criteria::ASC) Order by the pointnr column

\* @method ChildStgKeresztmetszetiForgalomV1Query orderByNrpoints($order = Criteria::ASC) Order by the nrpointS column

\* @method ChildStgKeresztmetszetiForgalomV1Query orderByX($order = Criteria::ASC) Order by the x column

\* @method ChildStgKeresztmetszetiForgalomV1Query orderByY($order = Criteria::ASC) Order by the y column

\* @method ChildStgKeresztmetszetiForgalomV1Query orderByMeasure($order = Criteria::ASC) Order by the measure column

\* @method ChildStgKeresztmetszetiForgalomV1Query orderByKezd($order = Criteria::ASC) Order by the kezd column

\* @method ChildStgKeresztmetszetiForgalomV1Query orderByVege($order = Criteria::ASC) Order by the vege column

\* @method ChildStgKeresztmetszetiForgalomV1Query orderByKszam($order = Criteria::ASC) Order by the kszam column

\* @method ChildStgKeresztmetszetiForgalomV1Query orderByPkod($order = Criteria::ASC) Order by the pkod column

\*

\* @method ChildStgKeresztmetszetiForgalomV1Query groupByPartnr() Group by the partnr column

\* @method ChildStgKeresztmetszetiForgalomV1Query groupByNrparts() Group by the nrparts column

\* @method ChildStgKeresztmetszetiForgalomV1Query groupByPointnr() Group by the pointnr column

\* @method ChildStgKeresztmetszetiForgalomV1Query groupByNrpoints() Group by the nrpointS column

\* @method ChildStgKeresztmetszetiForgalomV1Query groupByX() Group by the x column

\* @method ChildStgKeresztmetszetiForgalomV1Query groupByY() Group by the y column

\* @method ChildStgKeresztmetszetiForgalomV1Query groupByMeasure() Group by the measure column

\* @method ChildStgKeresztmetszetiForgalomV1Query groupByKezd() Group by the kezd column

\* @method ChildStgKeresztmetszetiForgalomV1Query groupByVege() Group by the vege column

\* @method ChildStgKeresztmetszetiForgalomV1Query groupByKszam() Group by the kszam column

\* @method ChildStgKeresztmetszetiForgalomV1Query groupByPkod() Group by the pkod column

\*

\* @method ChildStgKeresztmetszetiForgalomV1Query leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgKeresztmetszetiForgalomV1Query rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgKeresztmetszetiForgalomV1Query innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgKeresztmetszetiForgalomV1 findOne(ConnectionInterface $con = null) Return the first ChildStgKeresztmetszetiForgalomV1 matching the query

\* @method ChildStgKeresztmetszetiForgalomV1 findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgKeresztmetszetiForgalomV1 matching the query, or a new ChildStgKeresztmetszetiForgalomV1 object populated from the query conditions when no match is found

\*

\* @method ChildStgKeresztmetszetiForgalomV1 findOneByPartnr(int $partnr) Return the first ChildStgKeresztmetszetiForgalomV1 filtered by the partnr column

\* @method ChildStgKeresztmetszetiForgalomV1 findOneByNrparts(string $nrparts) Return the first ChildStgKeresztmetszetiForgalomV1 filtered by the nrparts column

\* @method ChildStgKeresztmetszetiForgalomV1 findOneByPointnr(string $pointnr) Return the first ChildStgKeresztmetszetiForgalomV1 filtered by the pointnr column

\* @method ChildStgKeresztmetszetiForgalomV1 findOneByNrpoints(string $nrpointS) Return the first ChildStgKeresztmetszetiForgalomV1 filtered by the nrpointS column

\* @method ChildStgKeresztmetszetiForgalomV1 findOneByX(string $x) Return the first ChildStgKeresztmetszetiForgalomV1 filtered by the x column

\* @method ChildStgKeresztmetszetiForgalomV1 findOneByY(string $y) Return the first ChildStgKeresztmetszetiForgalomV1 filtered by the y column

\* @method ChildStgKeresztmetszetiForgalomV1 findOneByMeasure(string $measure) Return the first ChildStgKeresztmetszetiForgalomV1 filtered by the measure column

\* @method ChildStgKeresztmetszetiForgalomV1 findOneByKezd(string $kezd) Return the first ChildStgKeresztmetszetiForgalomV1 filtered by the kezd column

\* @method ChildStgKeresztmetszetiForgalomV1 findOneByVege(string $vege) Return the first ChildStgKeresztmetszetiForgalomV1 filtered by the vege column

\* @method ChildStgKeresztmetszetiForgalomV1 findOneByKszam(string $kszam) Return the first ChildStgKeresztmetszetiForgalomV1 filtered by the kszam column

\* @method ChildStgKeresztmetszetiForgalomV1 findOneByPkod(string $pkod) Return the first ChildStgKeresztmetszetiForgalomV1 filtered by the pkod column

\*

\* @method ChildStgKeresztmetszetiForgalomV1[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgKeresztmetszetiForgalomV1 objects based on current ModelCriteria

\* @method ChildStgKeresztmetszetiForgalomV1[]|ObjectCollection findByPartnr(int $partnr) Return ChildStgKeresztmetszetiForgalomV1 objects filtered by the partnr column

\* @method ChildStgKeresztmetszetiForgalomV1[]|ObjectCollection findByNrparts(string $nrparts) Return ChildStgKeresztmetszetiForgalomV1 objects filtered by the nrparts column

\* @method ChildStgKeresztmetszetiForgalomV1[]|ObjectCollection findByPointnr(string $pointnr) Return ChildStgKeresztmetszetiForgalomV1 objects filtered by the pointnr column

\* @method ChildStgKeresztmetszetiForgalomV1[]|ObjectCollection findByNrpoints(string $nrpointS) Return ChildStgKeresztmetszetiForgalomV1 objects filtered by the nrpointS column

\* @method ChildStgKeresztmetszetiForgalomV1[]|ObjectCollection findByX(string $x) Return ChildStgKeresztmetszetiForgalomV1 objects filtered by the x column

\* @method ChildStgKeresztmetszetiForgalomV1[]|ObjectCollection findByY(string $y) Return ChildStgKeresztmetszetiForgalomV1 objects filtered by the y column

\* @method ChildStgKeresztmetszetiForgalomV1[]|ObjectCollection findByMeasure(string $measure) Return ChildStgKeresztmetszetiForgalomV1 objects filtered by the measure column

\* @method ChildStgKeresztmetszetiForgalomV1[]|ObjectCollection findByKezd(string $kezd) Return ChildStgKeresztmetszetiForgalomV1 objects filtered by the kezd column

\* @method ChildStgKeresztmetszetiForgalomV1[]|ObjectCollection findByVege(string $vege) Return ChildStgKeresztmetszetiForgalomV1 objects filtered by the vege column

\* @method ChildStgKeresztmetszetiForgalomV1[]|ObjectCollection findByKszam(string $kszam) Return ChildStgKeresztmetszetiForgalomV1 objects filtered by the kszam column

\* @method ChildStgKeresztmetszetiForgalomV1[]|ObjectCollection findByPkod(string $pkod) Return ChildStgKeresztmetszetiForgalomV1 objects filtered by the pkod column

\* @method ChildStgKeresztmetszetiForgalomV1[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgKeresztmetszetiForgalomV1Query extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgKeresztmetszetiForgalomV1Query object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgKeresztmetszetiForgalomV1', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgKeresztmetszetiForgalomV1Query object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgKeresztmetszetiForgalomV1Query

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgKeresztmetszetiForgalomV1Query) {

return $criteria;

}

$query = new ChildStgKeresztmetszetiForgalomV1Query();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgKeresztmetszetiForgalomV1|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

if ($key === null) {

return null;

}

if ((null !== ($obj = StgKeresztmetszetiForgalomV1TableMap::getInstanceFromPool((string) $key))) && !$this->formatter) {

// the object is already in the instance pool

return $obj;

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgKeresztmetszetiForgalomV1TableMap::DATABASE\_NAME);

}

$this->basePreSelect($con);

if ($this->formatter || $this->modelAlias || $this->with || $this->select

|| $this->selectColumns || $this->asColumns || $this->selectModifiers

|| $this->map || $this->having || $this->joins) {

return $this->findPkComplex($key, $con);

} else {

return $this->findPkSimple($key, $con);

}

}

/\*\*

\* Find object by primary key using raw SQL to go fast.

\* Bypass doSelect() and the object formatter by using generated code.

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con A connection object

\*

\* @return ChildStgKeresztmetszetiForgalomV1 A model object, or null if the key is not found

\*/

protected function findPkSimple($key, ConnectionInterface $con)

{

$sql = 'SELECT PARTNR, NRPARTS, POINTNR, NRPOINTS, X, Y, MEASURE, KEZD, VEGE, KSZAM, PKOD FROM stg\_keresztmetszeti\_forgalom\_v1 WHERE PARTNR = :p0';

try {

$stmt = $con->prepare($sql);

$stmt->bindValue(':p0', $key, PDO::PARAM\_INT);

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute SELECT statement [%s]', $sql), 0, $e);

}

$obj = null;

if ($row = $stmt->fetch(\PDO::FETCH\_NUM)) {

/\*\* @var ChildStgKeresztmetszetiForgalomV1 $obj \*/

$obj = new ChildStgKeresztmetszetiForgalomV1();

$obj->hydrate($row);

StgKeresztmetszetiForgalomV1TableMap::addInstanceToPool($obj, (string) $key);

}

$stmt->closeCursor();

return $obj;

}

/\*\*

\* Find object by primary key.

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con A connection object

\*

\* @return ChildStgKeresztmetszetiForgalomV1|array|mixed the result, formatted by the current formatter

\*/

protected function findPkComplex($key, ConnectionInterface $con)

{

// As the query uses a PK condition, no limit(1) is necessary.

$criteria = $this->isKeepQuery() ? clone $this : $this;

$dataFetcher = $criteria

->filterByPrimaryKey($key)

->doSelect($con);

return $criteria->getFormatter()->init($criteria)->formatOne($dataFetcher);

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(12, 56, 832), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getReadConnection($this->getDbName());

}

$this->basePreSelect($con);

$criteria = $this->isKeepQuery() ? clone $this : $this;

$dataFetcher = $criteria

->filterByPrimaryKeys($keys)

->doSelect($con);

return $criteria->getFormatter()->init($criteria)->format($dataFetcher);

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgKeresztmetszetiForgalomV1Query The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

return $this->addUsingAlias(StgKeresztmetszetiForgalomV1TableMap::COL\_PARTNR, $key, Criteria::EQUAL);

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgKeresztmetszetiForgalomV1Query The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

return $this->addUsingAlias(StgKeresztmetszetiForgalomV1TableMap::COL\_PARTNR, $keys, Criteria::IN);

}

/\*\*

\* Filter the query on the partnr column

\*

\* Example usage:

\* <code>

\* $query->filterByPartnr(1234); // WHERE partnr = 1234

\* $query->filterByPartnr(array(12, 34)); // WHERE partnr IN (12, 34)

\* $query->filterByPartnr(array('min' => 12)); // WHERE partnr > 12

\* </code>

\*

\* @param mixed $partnr The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomV1Query The current query, for fluid interface

\*/

public function filterByPartnr($partnr = null, $comparison = null)

{

if (is\_array($partnr)) {

$useMinMax = false;

if (isset($partnr['min'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomV1TableMap::COL\_PARTNR, $partnr['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($partnr['max'])) {

$this->addUsingAlias(StgKeresztmetszetiForgalomV1TableMap::COL\_PARTNR, $partnr['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomV1TableMap::COL\_PARTNR, $partnr, $comparison);

}

/\*\*

\* Filter the query on the nrparts column

\*

\* Example usage:

\* <code>

\* $query->filterByNrparts('fooValue'); // WHERE nrparts = 'fooValue'

\* $query->filterByNrparts('%fooValue%'); // WHERE nrparts LIKE '%fooValue%'

\* </code>

\*

\* @param string $nrparts The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomV1Query The current query, for fluid interface

\*/

public function filterByNrparts($nrparts = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($nrparts)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $nrparts)) {

$nrparts = str\_replace('\*', '%', $nrparts);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomV1TableMap::COL\_NRPARTS, $nrparts, $comparison);

}

/\*\*

\* Filter the query on the pointnr column

\*

\* Example usage:

\* <code>

\* $query->filterByPointnr('fooValue'); // WHERE pointnr = 'fooValue'

\* $query->filterByPointnr('%fooValue%'); // WHERE pointnr LIKE '%fooValue%'

\* </code>

\*

\* @param string $pointnr The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomV1Query The current query, for fluid interface

\*/

public function filterByPointnr($pointnr = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($pointnr)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $pointnr)) {

$pointnr = str\_replace('\*', '%', $pointnr);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomV1TableMap::COL\_POINTNR, $pointnr, $comparison);

}

/\*\*

\* Filter the query on the nrpointS column

\*

\* Example usage:

\* <code>

\* $query->filterByNrpoints('fooValue'); // WHERE nrpointS = 'fooValue'

\* $query->filterByNrpoints('%fooValue%'); // WHERE nrpointS LIKE '%fooValue%'

\* </code>

\*

\* @param string $nrpoints The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomV1Query The current query, for fluid interface

\*/

public function filterByNrpoints($nrpoints = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($nrpoints)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $nrpoints)) {

$nrpoints = str\_replace('\*', '%', $nrpoints);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomV1TableMap::COL\_NRPOINTS, $nrpoints, $comparison);

}

/\*\*

\* Filter the query on the x column

\*

\* Example usage:

\* <code>

\* $query->filterByX('fooValue'); // WHERE x = 'fooValue'

\* $query->filterByX('%fooValue%'); // WHERE x LIKE '%fooValue%'

\* </code>

\*

\* @param string $x The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomV1Query The current query, for fluid interface

\*/

public function filterByX($x = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($x)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $x)) {

$x = str\_replace('\*', '%', $x);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomV1TableMap::COL\_X, $x, $comparison);

}

/\*\*

\* Filter the query on the y column

\*

\* Example usage:

\* <code>

\* $query->filterByY('fooValue'); // WHERE y = 'fooValue'

\* $query->filterByY('%fooValue%'); // WHERE y LIKE '%fooValue%'

\* </code>

\*

\* @param string $y The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomV1Query The current query, for fluid interface

\*/

public function filterByY($y = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($y)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $y)) {

$y = str\_replace('\*', '%', $y);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomV1TableMap::COL\_Y, $y, $comparison);

}

/\*\*

\* Filter the query on the measure column

\*

\* Example usage:

\* <code>

\* $query->filterByMeasure('fooValue'); // WHERE measure = 'fooValue'

\* $query->filterByMeasure('%fooValue%'); // WHERE measure LIKE '%fooValue%'

\* </code>

\*

\* @param string $measure The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomV1Query The current query, for fluid interface

\*/

public function filterByMeasure($measure = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($measure)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $measure)) {

$measure = str\_replace('\*', '%', $measure);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomV1TableMap::COL\_MEASURE, $measure, $comparison);

}

/\*\*

\* Filter the query on the kezd column

\*

\* Example usage:

\* <code>

\* $query->filterByKezd('fooValue'); // WHERE kezd = 'fooValue'

\* $query->filterByKezd('%fooValue%'); // WHERE kezd LIKE '%fooValue%'

\* </code>

\*

\* @param string $kezd The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomV1Query The current query, for fluid interface

\*/

public function filterByKezd($kezd = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($kezd)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $kezd)) {

$kezd = str\_replace('\*', '%', $kezd);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomV1TableMap::COL\_KEZD, $kezd, $comparison);

}

/\*\*

\* Filter the query on the vege column

\*

\* Example usage:

\* <code>

\* $query->filterByVege('fooValue'); // WHERE vege = 'fooValue'

\* $query->filterByVege('%fooValue%'); // WHERE vege LIKE '%fooValue%'

\* </code>

\*

\* @param string $vege The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomV1Query The current query, for fluid interface

\*/

public function filterByVege($vege = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($vege)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $vege)) {

$vege = str\_replace('\*', '%', $vege);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomV1TableMap::COL\_VEGE, $vege, $comparison);

}

/\*\*

\* Filter the query on the kszam column

\*

\* Example usage:

\* <code>

\* $query->filterByKszam('fooValue'); // WHERE kszam = 'fooValue'

\* $query->filterByKszam('%fooValue%'); // WHERE kszam LIKE '%fooValue%'

\* </code>

\*

\* @param string $kszam The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomV1Query The current query, for fluid interface

\*/

public function filterByKszam($kszam = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($kszam)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $kszam)) {

$kszam = str\_replace('\*', '%', $kszam);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomV1TableMap::COL\_KSZAM, $kszam, $comparison);

}

/\*\*

\* Filter the query on the pkod column

\*

\* Example usage:

\* <code>

\* $query->filterByPkod('fooValue'); // WHERE pkod = 'fooValue'

\* $query->filterByPkod('%fooValue%'); // WHERE pkod LIKE '%fooValue%'

\* </code>

\*

\* @param string $pkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKeresztmetszetiForgalomV1Query The current query, for fluid interface

\*/

public function filterByPkod($pkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($pkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $pkod)) {

$pkod = str\_replace('\*', '%', $pkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKeresztmetszetiForgalomV1TableMap::COL\_PKOD, $pkod, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgKeresztmetszetiForgalomV1 $stgKeresztmetszetiForgalomV1 Object to remove from the list of results

\*

\* @return $this|ChildStgKeresztmetszetiForgalomV1Query The current query, for fluid interface

\*/

public function prune($stgKeresztmetszetiForgalomV1 = null)

{

if ($stgKeresztmetszetiForgalomV1) {

$this->addUsingAlias(StgKeresztmetszetiForgalomV1TableMap::COL\_PARTNR, $stgKeresztmetszetiForgalomV1->getPartnr(), Criteria::NOT\_EQUAL);

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_keresztmetszeti\_forgalom\_v1 table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKeresztmetszetiForgalomV1TableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgKeresztmetszetiForgalomV1TableMap::clearInstancePool();

StgKeresztmetszetiForgalomV1TableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKeresztmetszetiForgalomV1TableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgKeresztmetszetiForgalomV1TableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgKeresztmetszetiForgalomV1TableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgKeresztmetszetiForgalomV1TableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgKeresztmetszetiForgalomV1Query

#### StgKollegium.php

<?php

namespace Base;

use \StgKollegiumQuery as ChildStgKollegiumQuery;

use \Exception;

use \PDO;

use Map\StgKollegiumTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgKollegium implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgKollegiumTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the kollegiumban\_lako\_nappali\_alt\_tanulok\_szama field.

\* @var int

\*/

protected $kollegiumban\_lako\_nappali\_alt\_tanulok\_szama;

/\*\*

\* The value for the kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama field.

\* @var int

\*/

protected $kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama;

/\*\*

\* The value for the kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama field.

\* @var int

\*/

protected $kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama;

/\*\*

\* The value for the kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama field.

\* @var int

\*/

protected $kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgKollegium object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgKollegium</code> instance. If

\* <code>obj</code> is an instance of <code>StgKollegium</code>, delegates to

\* <code>equals(StgKollegium)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgKollegium The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [kollegiumban\_lako\_nappali\_alt\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getKollegiumbanLakoNappaliAltTanulokSzama()

{

return $this->kollegiumban\_lako\_nappali\_alt\_tanulok\_szama;

}

/\*\*

\* Get the [kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getKollegiumbanLakoNappaliKozepiskolaiTanulokSzama()

{

return $this->kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama;

}

/\*\*

\* Get the [kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama()

{

return $this->kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama;

}

/\*\*

\* Get the [kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama()

{

return $this->kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgKollegiumTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgKollegiumTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgKollegiumTableMap::translateFieldName('KollegiumbanLakoNappaliAltTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kollegiumban\_lako\_nappali\_alt\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgKollegiumTableMap::translateFieldName('KollegiumbanLakoNappaliKozepiskolaiTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgKollegiumTableMap::translateFieldName('KollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 5 + $startcol : StgKollegiumTableMap::translateFieldName('KollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 6; // 6 = StgKollegiumTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgKollegium'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgKollegium The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgKollegiumTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgKollegium The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgKollegiumTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [kollegiumban\_lako\_nappali\_alt\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgKollegium The current object (for fluent API support)

\*/

public function setKollegiumbanLakoNappaliAltTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->kollegiumban\_lako\_nappali\_alt\_tanulok\_szama !== $v) {

$this->kollegiumban\_lako\_nappali\_alt\_tanulok\_szama = $v;

$this->modifiedColumns[StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA] = true;

}

return $this;

} // setKollegiumbanLakoNappaliAltTanulokSzama()

/\*\*

\* Set the value of [kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgKollegium The current object (for fluent API support)

\*/

public function setKollegiumbanLakoNappaliKozepiskolaiTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama !== $v) {

$this->kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama = $v;

$this->modifiedColumns[StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA] = true;

}

return $this;

} // setKollegiumbanLakoNappaliKozepiskolaiTanulokSzama()

/\*\*

\* Set the value of [kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgKollegium The current object (for fluent API support)

\*/

public function setKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama !== $v) {

$this->kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama = $v;

$this->modifiedColumns[StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA] = true;

}

return $this;

} // setKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama()

/\*\*

\* Set the value of [kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgKollegium The current object (for fluent API support)

\*/

public function setKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama !== $v) {

$this->kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama = $v;

$this->modifiedColumns[StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA] = true;

}

return $this;

} // setKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgKollegiumTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgKollegiumQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgKollegium::setDeleted()

\* @see StgKollegium::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKollegiumTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgKollegiumQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKollegiumTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgKollegiumTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgKollegiumTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgKollegiumTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA';

}

if ($this->isColumnModified(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA';

}

if ($this->isColumnModified(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA';

}

if ($this->isColumnModified(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_kollegium (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->kollegiumban\_lako\_nappali\_alt\_tanulok\_szama, PDO::PARAM\_INT);

break;

case 'KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama, PDO::PARAM\_INT);

break;

case 'KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama, PDO::PARAM\_INT);

break;

case 'KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgKollegiumTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getKollegiumbanLakoNappaliAltTanulokSzama();

break;

case 3:

return $this->getKollegiumbanLakoNappaliKozepiskolaiTanulokSzama();

break;

case 4:

return $this->getKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama();

break;

case 5:

return $this->getKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgKollegium'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgKollegium'][$this->getPrimaryKey()] = true;

$keys = StgKollegiumTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getKollegiumbanLakoNappaliAltTanulokSzama(),

$keys[3] => $this->getKollegiumbanLakoNappaliKozepiskolaiTanulokSzama(),

$keys[4] => $this->getKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama(),

$keys[5] => $this->getKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgKollegium

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgKollegiumTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgKollegium

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setKollegiumbanLakoNappaliAltTanulokSzama($value);

break;

case 3:

$this->setKollegiumbanLakoNappaliKozepiskolaiTanulokSzama($value);

break;

case 4:

$this->setKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama($value);

break;

case 5:

$this->setKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgKollegiumTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setKollegiumbanLakoNappaliAltTanulokSzama($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setKollegiumbanLakoNappaliKozepiskolaiTanulokSzama($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama($arr[$keys[4]]);

}

if (array\_key\_exists($keys[5], $arr)) {

$this->setKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama($arr[$keys[5]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgKollegium The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgKollegiumTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgKollegiumTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgKollegiumTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgKollegiumTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgKollegiumTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA)) {

$criteria->add(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA, $this->kollegiumban\_lako\_nappali\_alt\_tanulok\_szama);

}

if ($this->isColumnModified(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA)) {

$criteria->add(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $this->kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama);

}

if ($this->isColumnModified(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA)) {

$criteria->add(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA, $this->kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama);

}

if ($this->isColumnModified(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA)) {

$criteria->add(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA, $this->kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgKollegium object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgKollegium (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setKollegiumbanLakoNappaliAltTanulokSzama($this->getKollegiumbanLakoNappaliAltTanulokSzama());

$copyObj->setKollegiumbanLakoNappaliKozepiskolaiTanulokSzama($this->getKollegiumbanLakoNappaliKozepiskolaiTanulokSzama());

$copyObj->setKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama($this->getKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama());

$copyObj->setKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama($this->getKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgKollegium Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->kollegiumban\_lako\_nappali\_alt\_tanulok\_szama = null;

$this->kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama = null;

$this->kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama = null;

$this->kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgKollegiumTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgKollegiumQuery.php

<?php

namespace Base;

use \StgKollegium as ChildStgKollegium;

use \StgKollegiumQuery as ChildStgKollegiumQuery;

use \Exception;

use Map\StgKollegiumTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_kollegium' table.

\*

\*

\*

\* @method ChildStgKollegiumQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgKollegiumQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgKollegiumQuery orderByKollegiumbanLakoNappaliAltTanulokSzama($order = Criteria::ASC) Order by the kollegiumban\_lako\_nappali\_alt\_tanulok\_szama column

\* @method ChildStgKollegiumQuery orderByKollegiumbanLakoNappaliKozepiskolaiTanulokSzama($order = Criteria::ASC) Order by the kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgKollegiumQuery orderByKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama($order = Criteria::ASC) Order by the kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama column

\* @method ChildStgKollegiumQuery orderByKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama($order = Criteria::ASC) Order by the kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama column

\*

\* @method ChildStgKollegiumQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgKollegiumQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgKollegiumQuery groupByKollegiumbanLakoNappaliAltTanulokSzama() Group by the kollegiumban\_lako\_nappali\_alt\_tanulok\_szama column

\* @method ChildStgKollegiumQuery groupByKollegiumbanLakoNappaliKozepiskolaiTanulokSzama() Group by the kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgKollegiumQuery groupByKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama() Group by the kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama column

\* @method ChildStgKollegiumQuery groupByKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama() Group by the kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama column

\*

\* @method ChildStgKollegiumQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgKollegiumQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgKollegiumQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgKollegium findOne(ConnectionInterface $con = null) Return the first ChildStgKollegium matching the query

\* @method ChildStgKollegium findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgKollegium matching the query, or a new ChildStgKollegium object populated from the query conditions when no match is found

\*

\* @method ChildStgKollegium findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgKollegium filtered by the telepules\_nev column

\* @method ChildStgKollegium findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgKollegium filtered by the telepules\_KSHKOD column

\* @method ChildStgKollegium findOneByKollegiumbanLakoNappaliAltTanulokSzama(int $kollegiumban\_lako\_nappali\_alt\_tanulok\_szama) Return the first ChildStgKollegium filtered by the kollegiumban\_lako\_nappali\_alt\_tanulok\_szama column

\* @method ChildStgKollegium findOneByKollegiumbanLakoNappaliKozepiskolaiTanulokSzama(int $kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama) Return the first ChildStgKollegium filtered by the kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgKollegium findOneByKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama(int $kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama) Return the first ChildStgKollegium filtered by the kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama column

\* @method ChildStgKollegium findOneByKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama(int $kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama) Return the first ChildStgKollegium filtered by the kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama column

\*

\* @method ChildStgKollegium[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgKollegium objects based on current ModelCriteria

\* @method ChildStgKollegium[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgKollegium objects filtered by the telepules\_nev column

\* @method ChildStgKollegium[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgKollegium objects filtered by the telepules\_KSHKOD column

\* @method ChildStgKollegium[]|ObjectCollection findByKollegiumbanLakoNappaliAltTanulokSzama(int $kollegiumban\_lako\_nappali\_alt\_tanulok\_szama) Return ChildStgKollegium objects filtered by the kollegiumban\_lako\_nappali\_alt\_tanulok\_szama column

\* @method ChildStgKollegium[]|ObjectCollection findByKollegiumbanLakoNappaliKozepiskolaiTanulokSzama(int $kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama) Return ChildStgKollegium objects filtered by the kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgKollegium[]|ObjectCollection findByKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama(int $kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama) Return ChildStgKollegium objects filtered by the kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama column

\* @method ChildStgKollegium[]|ObjectCollection findByKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama(int $kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama) Return ChildStgKollegium objects filtered by the kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama column

\* @method ChildStgKollegium[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgKollegiumQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgKollegiumQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgKollegium', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgKollegiumQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgKollegiumQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgKollegiumQuery) {

return $criteria;

}

$query = new ChildStgKollegiumQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgKollegium|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgKollegium object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgKollegium object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgKollegiumQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgKollegium object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgKollegiumQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgKollegium object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKollegiumQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKollegiumTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKollegiumQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKollegiumTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the kollegiumban\_lako\_nappali\_alt\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByKollegiumbanLakoNappaliAltTanulokSzama(1234); // WHERE kollegiumban\_lako\_nappali\_alt\_tanulok\_szama = 1234

\* $query->filterByKollegiumbanLakoNappaliAltTanulokSzama(array(12, 34)); // WHERE kollegiumban\_lako\_nappali\_alt\_tanulok\_szama IN (12, 34)

\* $query->filterByKollegiumbanLakoNappaliAltTanulokSzama(array('min' => 12)); // WHERE kollegiumban\_lako\_nappali\_alt\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $kollegiumbanLakoNappaliAltTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKollegiumQuery The current query, for fluid interface

\*/

public function filterByKollegiumbanLakoNappaliAltTanulokSzama($kollegiumbanLakoNappaliAltTanulokSzama = null, $comparison = null)

{

if (is\_array($kollegiumbanLakoNappaliAltTanulokSzama)) {

$useMinMax = false;

if (isset($kollegiumbanLakoNappaliAltTanulokSzama['min'])) {

$this->addUsingAlias(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA, $kollegiumbanLakoNappaliAltTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($kollegiumbanLakoNappaliAltTanulokSzama['max'])) {

$this->addUsingAlias(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA, $kollegiumbanLakoNappaliAltTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA, $kollegiumbanLakoNappaliAltTanulokSzama, $comparison);

}

/\*\*

\* Filter the query on the kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByKollegiumbanLakoNappaliKozepiskolaiTanulokSzama(1234); // WHERE kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama = 1234

\* $query->filterByKollegiumbanLakoNappaliKozepiskolaiTanulokSzama(array(12, 34)); // WHERE kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama IN (12, 34)

\* $query->filterByKollegiumbanLakoNappaliKozepiskolaiTanulokSzama(array('min' => 12)); // WHERE kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $kollegiumbanLakoNappaliKozepiskolaiTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKollegiumQuery The current query, for fluid interface

\*/

public function filterByKollegiumbanLakoNappaliKozepiskolaiTanulokSzama($kollegiumbanLakoNappaliKozepiskolaiTanulokSzama = null, $comparison = null)

{

if (is\_array($kollegiumbanLakoNappaliKozepiskolaiTanulokSzama)) {

$useMinMax = false;

if (isset($kollegiumbanLakoNappaliKozepiskolaiTanulokSzama['min'])) {

$this->addUsingAlias(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $kollegiumbanLakoNappaliKozepiskolaiTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($kollegiumbanLakoNappaliKozepiskolaiTanulokSzama['max'])) {

$this->addUsingAlias(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $kollegiumbanLakoNappaliKozepiskolaiTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $kollegiumbanLakoNappaliKozepiskolaiTanulokSzama, $comparison);

}

/\*\*

\* Filter the query on the kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama(1234); // WHERE kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama = 1234

\* $query->filterByKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama(array(12, 34)); // WHERE kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama IN (12, 34)

\* $query->filterByKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama(array('min' => 12)); // WHERE kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $kollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKollegiumQuery The current query, for fluid interface

\*/

public function filterByKollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama($kollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama = null, $comparison = null)

{

if (is\_array($kollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama)) {

$useMinMax = false;

if (isset($kollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama['min'])) {

$this->addUsingAlias(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA, $kollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($kollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama['max'])) {

$this->addUsingAlias(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA, $kollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA, $kollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama, $comparison);

}

/\*\*

\* Filter the query on the kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama(1234); // WHERE kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama = 1234

\* $query->filterByKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama(array(12, 34)); // WHERE kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama IN (12, 34)

\* $query->filterByKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama(array('min' => 12)); // WHERE kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $kollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKollegiumQuery The current query, for fluid interface

\*/

public function filterByKollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama($kollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama = null, $comparison = null)

{

if (is\_array($kollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama)) {

$useMinMax = false;

if (isset($kollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama['min'])) {

$this->addUsingAlias(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA, $kollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($kollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama['max'])) {

$this->addUsingAlias(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA, $kollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA, $kollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgKollegium $stgKollegium Object to remove from the list of results

\*

\* @return $this|ChildStgKollegiumQuery The current query, for fluid interface

\*/

public function prune($stgKollegium = null)

{

if ($stgKollegium) {

throw new LogicException('StgKollegium object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_kollegium table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKollegiumTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgKollegiumTableMap::clearInstancePool();

StgKollegiumTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKollegiumTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgKollegiumTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgKollegiumTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgKollegiumTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgKollegiumQuery

#### StgKulturaEsSport.php

<?php

namespace Base;

use \StgKulturaEsSportQuery as ChildStgKulturaEsSportQuery;

use \Exception;

use \PDO;

use Map\StgKulturaEsSportTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgKulturaEsSport implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgKulturaEsSportTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the filmszinhaz field.

\* @var boolean

\*/

protected $filmszinhaz;

/\*\*

\* The value for the kozmuvelodesi\_intezmeny field.

\* @var boolean

\*/

protected $kozmuvelodesi\_intezmeny;

/\*\*

\* The value for the muzeum field.

\* @var boolean

\*/

protected $muzeum;

/\*\*

\* The value for the piac field.

\* @var boolean

\*/

protected $piac;

/\*\*

\* The value for the sportcsarnok\_sportpalya field.

\* @var boolean

\*/

protected $sportcsarnok\_sportpalya;

/\*\*

\* The value for the strand field.

\* @var boolean

\*/

protected $strand;

/\*\*

\* The value for the telepulesi\_konyvtar field.

\* @var boolean

\*/

protected $telepulesi\_konyvtar;

/\*\*

\* The value for the uszoda\_furdo\_gyogyfurdo field.

\* @var boolean

\*/

protected $uszoda\_furdo\_gyogyfurdo;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgKulturaEsSport object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgKulturaEsSport</code> instance. If

\* <code>obj</code> is an instance of <code>StgKulturaEsSport</code>, delegates to

\* <code>equals(StgKulturaEsSport)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgKulturaEsSport The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [filmszinhaz] column value.

\*

\* @return boolean

\*/

public function getFilmszinhaz()

{

return $this->filmszinhaz;

}

/\*\*

\* Get the [filmszinhaz] column value.

\*

\* @return boolean

\*/

public function isFilmszinhaz()

{

return $this->getFilmszinhaz();

}

/\*\*

\* Get the [kozmuvelodesi\_intezmeny] column value.

\*

\* @return boolean

\*/

public function getKozmuvelodesiIntezmeny()

{

return $this->kozmuvelodesi\_intezmeny;

}

/\*\*

\* Get the [kozmuvelodesi\_intezmeny] column value.

\*

\* @return boolean

\*/

public function isKozmuvelodesiIntezmeny()

{

return $this->getKozmuvelodesiIntezmeny();

}

/\*\*

\* Get the [muzeum] column value.

\*

\* @return boolean

\*/

public function getMuzeum()

{

return $this->muzeum;

}

/\*\*

\* Get the [muzeum] column value.

\*

\* @return boolean

\*/

public function isMuzeum()

{

return $this->getMuzeum();

}

/\*\*

\* Get the [piac] column value.

\*

\* @return boolean

\*/

public function getPiac()

{

return $this->piac;

}

/\*\*

\* Get the [piac] column value.

\*

\* @return boolean

\*/

public function isPiac()

{

return $this->getPiac();

}

/\*\*

\* Get the [sportcsarnok\_sportpalya] column value.

\*

\* @return boolean

\*/

public function getSportcsarnokSportpalya()

{

return $this->sportcsarnok\_sportpalya;

}

/\*\*

\* Get the [sportcsarnok\_sportpalya] column value.

\*

\* @return boolean

\*/

public function isSportcsarnokSportpalya()

{

return $this->getSportcsarnokSportpalya();

}

/\*\*

\* Get the [strand] column value.

\*

\* @return boolean

\*/

public function getStrand()

{

return $this->strand;

}

/\*\*

\* Get the [strand] column value.

\*

\* @return boolean

\*/

public function isStrand()

{

return $this->getStrand();

}

/\*\*

\* Get the [telepulesi\_konyvtar] column value.

\*

\* @return boolean

\*/

public function getTelepulesiKonyvtar()

{

return $this->telepulesi\_konyvtar;

}

/\*\*

\* Get the [telepulesi\_konyvtar] column value.

\*

\* @return boolean

\*/

public function isTelepulesiKonyvtar()

{

return $this->getTelepulesiKonyvtar();

}

/\*\*

\* Get the [uszoda\_furdo\_gyogyfurdo] column value.

\*

\* @return boolean

\*/

public function getUszodaFurdoGyogyfurdo()

{

return $this->uszoda\_furdo\_gyogyfurdo;

}

/\*\*

\* Get the [uszoda\_furdo\_gyogyfurdo] column value.

\*

\* @return boolean

\*/

public function isUszodaFurdoGyogyfurdo()

{

return $this->getUszodaFurdoGyogyfurdo();

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgKulturaEsSportTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgKulturaEsSportTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgKulturaEsSportTableMap::translateFieldName('Filmszinhaz', TableMap::TYPE\_PHPNAME, $indexType)];

$this->filmszinhaz = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgKulturaEsSportTableMap::translateFieldName('KozmuvelodesiIntezmeny', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kozmuvelodesi\_intezmeny = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgKulturaEsSportTableMap::translateFieldName('Muzeum', TableMap::TYPE\_PHPNAME, $indexType)];

$this->muzeum = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 5 + $startcol : StgKulturaEsSportTableMap::translateFieldName('Piac', TableMap::TYPE\_PHPNAME, $indexType)];

$this->piac = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 6 + $startcol : StgKulturaEsSportTableMap::translateFieldName('SportcsarnokSportpalya', TableMap::TYPE\_PHPNAME, $indexType)];

$this->sportcsarnok\_sportpalya = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 7 + $startcol : StgKulturaEsSportTableMap::translateFieldName('Strand', TableMap::TYPE\_PHPNAME, $indexType)];

$this->strand = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 8 + $startcol : StgKulturaEsSportTableMap::translateFieldName('TelepulesiKonyvtar', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepulesi\_konyvtar = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 9 + $startcol : StgKulturaEsSportTableMap::translateFieldName('UszodaFurdoGyogyfurdo', TableMap::TYPE\_PHPNAME, $indexType)];

$this->uszoda\_furdo\_gyogyfurdo = (null !== $col) ? (boolean) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 10; // 10 = StgKulturaEsSportTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgKulturaEsSport'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgKulturaEsSport The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgKulturaEsSportTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgKulturaEsSport The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgKulturaEsSportTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Sets the value of the [filmszinhaz] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgKulturaEsSport The current object (for fluent API support)

\*/

public function setFilmszinhaz($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->filmszinhaz !== $v) {

$this->filmszinhaz = $v;

$this->modifiedColumns[StgKulturaEsSportTableMap::COL\_FILMSZINHAZ] = true;

}

return $this;

} // setFilmszinhaz()

/\*\*

\* Sets the value of the [kozmuvelodesi\_intezmeny] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgKulturaEsSport The current object (for fluent API support)

\*/

public function setKozmuvelodesiIntezmeny($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->kozmuvelodesi\_intezmeny !== $v) {

$this->kozmuvelodesi\_intezmeny = $v;

$this->modifiedColumns[StgKulturaEsSportTableMap::COL\_KOZMUVELODESI\_INTEZMENY] = true;

}

return $this;

} // setKozmuvelodesiIntezmeny()

/\*\*

\* Sets the value of the [muzeum] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgKulturaEsSport The current object (for fluent API support)

\*/

public function setMuzeum($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->muzeum !== $v) {

$this->muzeum = $v;

$this->modifiedColumns[StgKulturaEsSportTableMap::COL\_MUZEUM] = true;

}

return $this;

} // setMuzeum()

/\*\*

\* Sets the value of the [piac] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgKulturaEsSport The current object (for fluent API support)

\*/

public function setPiac($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->piac !== $v) {

$this->piac = $v;

$this->modifiedColumns[StgKulturaEsSportTableMap::COL\_PIAC] = true;

}

return $this;

} // setPiac()

/\*\*

\* Sets the value of the [sportcsarnok\_sportpalya] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgKulturaEsSport The current object (for fluent API support)

\*/

public function setSportcsarnokSportpalya($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->sportcsarnok\_sportpalya !== $v) {

$this->sportcsarnok\_sportpalya = $v;

$this->modifiedColumns[StgKulturaEsSportTableMap::COL\_SPORTCSARNOK\_SPORTPALYA] = true;

}

return $this;

} // setSportcsarnokSportpalya()

/\*\*

\* Sets the value of the [strand] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgKulturaEsSport The current object (for fluent API support)

\*/

public function setStrand($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->strand !== $v) {

$this->strand = $v;

$this->modifiedColumns[StgKulturaEsSportTableMap::COL\_STRAND] = true;

}

return $this;

} // setStrand()

/\*\*

\* Sets the value of the [telepulesi\_konyvtar] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgKulturaEsSport The current object (for fluent API support)

\*/

public function setTelepulesiKonyvtar($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->telepulesi\_konyvtar !== $v) {

$this->telepulesi\_konyvtar = $v;

$this->modifiedColumns[StgKulturaEsSportTableMap::COL\_TELEPULESI\_KONYVTAR] = true;

}

return $this;

} // setTelepulesiKonyvtar()

/\*\*

\* Sets the value of the [uszoda\_furdo\_gyogyfurdo] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgKulturaEsSport The current object (for fluent API support)

\*/

public function setUszodaFurdoGyogyfurdo($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->uszoda\_furdo\_gyogyfurdo !== $v) {

$this->uszoda\_furdo\_gyogyfurdo = $v;

$this->modifiedColumns[StgKulturaEsSportTableMap::COL\_USZODA\_FURDO\_GYOGYFURDO] = true;

}

return $this;

} // setUszodaFurdoGyogyfurdo()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgKulturaEsSportTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgKulturaEsSportQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgKulturaEsSport::setDeleted()

\* @see StgKulturaEsSport::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKulturaEsSportTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgKulturaEsSportQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKulturaEsSportTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgKulturaEsSportTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_FILMSZINHAZ)) {

$modifiedColumns[':p' . $index++] = 'FILMSZINHAZ';

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_KOZMUVELODESI\_INTEZMENY)) {

$modifiedColumns[':p' . $index++] = 'KOZMUVELODESI\_INTEZMENY';

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_MUZEUM)) {

$modifiedColumns[':p' . $index++] = 'MUZEUM';

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_PIAC)) {

$modifiedColumns[':p' . $index++] = 'PIAC';

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_SPORTCSARNOK\_SPORTPALYA)) {

$modifiedColumns[':p' . $index++] = 'SPORTCSARNOK\_SPORTPALYA';

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_STRAND)) {

$modifiedColumns[':p' . $index++] = 'STRAND';

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_TELEPULESI\_KONYVTAR)) {

$modifiedColumns[':p' . $index++] = 'TELEPULESI\_KONYVTAR';

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_USZODA\_FURDO\_GYOGYFURDO)) {

$modifiedColumns[':p' . $index++] = 'USZODA\_FURDO\_GYOGYFURDO';

}

$sql = sprintf(

'INSERT INTO stg\_kultura\_es\_sport (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'FILMSZINHAZ':

$stmt->bindValue($identifier, (int) $this->filmszinhaz, PDO::PARAM\_INT);

break;

case 'KOZMUVELODESI\_INTEZMENY':

$stmt->bindValue($identifier, (int) $this->kozmuvelodesi\_intezmeny, PDO::PARAM\_INT);

break;

case 'MUZEUM':

$stmt->bindValue($identifier, (int) $this->muzeum, PDO::PARAM\_INT);

break;

case 'PIAC':

$stmt->bindValue($identifier, (int) $this->piac, PDO::PARAM\_INT);

break;

case 'SPORTCSARNOK\_SPORTPALYA':

$stmt->bindValue($identifier, (int) $this->sportcsarnok\_sportpalya, PDO::PARAM\_INT);

break;

case 'STRAND':

$stmt->bindValue($identifier, (int) $this->strand, PDO::PARAM\_INT);

break;

case 'TELEPULESI\_KONYVTAR':

$stmt->bindValue($identifier, (int) $this->telepulesi\_konyvtar, PDO::PARAM\_INT);

break;

case 'USZODA\_FURDO\_GYOGYFURDO':

$stmt->bindValue($identifier, (int) $this->uszoda\_furdo\_gyogyfurdo, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgKulturaEsSportTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getFilmszinhaz();

break;

case 3:

return $this->getKozmuvelodesiIntezmeny();

break;

case 4:

return $this->getMuzeum();

break;

case 5:

return $this->getPiac();

break;

case 6:

return $this->getSportcsarnokSportpalya();

break;

case 7:

return $this->getStrand();

break;

case 8:

return $this->getTelepulesiKonyvtar();

break;

case 9:

return $this->getUszodaFurdoGyogyfurdo();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgKulturaEsSport'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgKulturaEsSport'][$this->getPrimaryKey()] = true;

$keys = StgKulturaEsSportTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getFilmszinhaz(),

$keys[3] => $this->getKozmuvelodesiIntezmeny(),

$keys[4] => $this->getMuzeum(),

$keys[5] => $this->getPiac(),

$keys[6] => $this->getSportcsarnokSportpalya(),

$keys[7] => $this->getStrand(),

$keys[8] => $this->getTelepulesiKonyvtar(),

$keys[9] => $this->getUszodaFurdoGyogyfurdo(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgKulturaEsSport

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgKulturaEsSportTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgKulturaEsSport

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setFilmszinhaz($value);

break;

case 3:

$this->setKozmuvelodesiIntezmeny($value);

break;

case 4:

$this->setMuzeum($value);

break;

case 5:

$this->setPiac($value);

break;

case 6:

$this->setSportcsarnokSportpalya($value);

break;

case 7:

$this->setStrand($value);

break;

case 8:

$this->setTelepulesiKonyvtar($value);

break;

case 9:

$this->setUszodaFurdoGyogyfurdo($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgKulturaEsSportTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setFilmszinhaz($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setKozmuvelodesiIntezmeny($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setMuzeum($arr[$keys[4]]);

}

if (array\_key\_exists($keys[5], $arr)) {

$this->setPiac($arr[$keys[5]]);

}

if (array\_key\_exists($keys[6], $arr)) {

$this->setSportcsarnokSportpalya($arr[$keys[6]]);

}

if (array\_key\_exists($keys[7], $arr)) {

$this->setStrand($arr[$keys[7]]);

}

if (array\_key\_exists($keys[8], $arr)) {

$this->setTelepulesiKonyvtar($arr[$keys[8]]);

}

if (array\_key\_exists($keys[9], $arr)) {

$this->setUszodaFurdoGyogyfurdo($arr[$keys[9]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgKulturaEsSport The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgKulturaEsSportTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgKulturaEsSportTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgKulturaEsSportTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_FILMSZINHAZ)) {

$criteria->add(StgKulturaEsSportTableMap::COL\_FILMSZINHAZ, $this->filmszinhaz);

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_KOZMUVELODESI\_INTEZMENY)) {

$criteria->add(StgKulturaEsSportTableMap::COL\_KOZMUVELODESI\_INTEZMENY, $this->kozmuvelodesi\_intezmeny);

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_MUZEUM)) {

$criteria->add(StgKulturaEsSportTableMap::COL\_MUZEUM, $this->muzeum);

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_PIAC)) {

$criteria->add(StgKulturaEsSportTableMap::COL\_PIAC, $this->piac);

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_SPORTCSARNOK\_SPORTPALYA)) {

$criteria->add(StgKulturaEsSportTableMap::COL\_SPORTCSARNOK\_SPORTPALYA, $this->sportcsarnok\_sportpalya);

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_STRAND)) {

$criteria->add(StgKulturaEsSportTableMap::COL\_STRAND, $this->strand);

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_TELEPULESI\_KONYVTAR)) {

$criteria->add(StgKulturaEsSportTableMap::COL\_TELEPULESI\_KONYVTAR, $this->telepulesi\_konyvtar);

}

if ($this->isColumnModified(StgKulturaEsSportTableMap::COL\_USZODA\_FURDO\_GYOGYFURDO)) {

$criteria->add(StgKulturaEsSportTableMap::COL\_USZODA\_FURDO\_GYOGYFURDO, $this->uszoda\_furdo\_gyogyfurdo);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgKulturaEsSport object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgKulturaEsSport (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setFilmszinhaz($this->getFilmszinhaz());

$copyObj->setKozmuvelodesiIntezmeny($this->getKozmuvelodesiIntezmeny());

$copyObj->setMuzeum($this->getMuzeum());

$copyObj->setPiac($this->getPiac());

$copyObj->setSportcsarnokSportpalya($this->getSportcsarnokSportpalya());

$copyObj->setStrand($this->getStrand());

$copyObj->setTelepulesiKonyvtar($this->getTelepulesiKonyvtar());

$copyObj->setUszodaFurdoGyogyfurdo($this->getUszodaFurdoGyogyfurdo());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgKulturaEsSport Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->filmszinhaz = null;

$this->kozmuvelodesi\_intezmeny = null;

$this->muzeum = null;

$this->piac = null;

$this->sportcsarnok\_sportpalya = null;

$this->strand = null;

$this->telepulesi\_konyvtar = null;

$this->uszoda\_furdo\_gyogyfurdo = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgKulturaEsSportTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgKulturaEsSportQuery.php

<?php

namespace Base;

use \StgKulturaEsSport as ChildStgKulturaEsSport;

use \StgKulturaEsSportQuery as ChildStgKulturaEsSportQuery;

use \Exception;

use Map\StgKulturaEsSportTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_kultura\_es\_sport' table.

\*

\*

\*

\* @method ChildStgKulturaEsSportQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgKulturaEsSportQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgKulturaEsSportQuery orderByFilmszinhaz($order = Criteria::ASC) Order by the filmszinhaz column

\* @method ChildStgKulturaEsSportQuery orderByKozmuvelodesiIntezmeny($order = Criteria::ASC) Order by the kozmuvelodesi\_intezmeny column

\* @method ChildStgKulturaEsSportQuery orderByMuzeum($order = Criteria::ASC) Order by the muzeum column

\* @method ChildStgKulturaEsSportQuery orderByPiac($order = Criteria::ASC) Order by the piac column

\* @method ChildStgKulturaEsSportQuery orderBySportcsarnokSportpalya($order = Criteria::ASC) Order by the sportcsarnok\_sportpalya column

\* @method ChildStgKulturaEsSportQuery orderByStrand($order = Criteria::ASC) Order by the strand column

\* @method ChildStgKulturaEsSportQuery orderByTelepulesiKonyvtar($order = Criteria::ASC) Order by the telepulesi\_konyvtar column

\* @method ChildStgKulturaEsSportQuery orderByUszodaFurdoGyogyfurdo($order = Criteria::ASC) Order by the uszoda\_furdo\_gyogyfurdo column

\*

\* @method ChildStgKulturaEsSportQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgKulturaEsSportQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgKulturaEsSportQuery groupByFilmszinhaz() Group by the filmszinhaz column

\* @method ChildStgKulturaEsSportQuery groupByKozmuvelodesiIntezmeny() Group by the kozmuvelodesi\_intezmeny column

\* @method ChildStgKulturaEsSportQuery groupByMuzeum() Group by the muzeum column

\* @method ChildStgKulturaEsSportQuery groupByPiac() Group by the piac column

\* @method ChildStgKulturaEsSportQuery groupBySportcsarnokSportpalya() Group by the sportcsarnok\_sportpalya column

\* @method ChildStgKulturaEsSportQuery groupByStrand() Group by the strand column

\* @method ChildStgKulturaEsSportQuery groupByTelepulesiKonyvtar() Group by the telepulesi\_konyvtar column

\* @method ChildStgKulturaEsSportQuery groupByUszodaFurdoGyogyfurdo() Group by the uszoda\_furdo\_gyogyfurdo column

\*

\* @method ChildStgKulturaEsSportQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgKulturaEsSportQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgKulturaEsSportQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgKulturaEsSport findOne(ConnectionInterface $con = null) Return the first ChildStgKulturaEsSport matching the query

\* @method ChildStgKulturaEsSport findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgKulturaEsSport matching the query, or a new ChildStgKulturaEsSport object populated from the query conditions when no match is found

\*

\* @method ChildStgKulturaEsSport findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgKulturaEsSport filtered by the telepules\_nev column

\* @method ChildStgKulturaEsSport findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgKulturaEsSport filtered by the telepules\_KSHKOD column

\* @method ChildStgKulturaEsSport findOneByFilmszinhaz(boolean $filmszinhaz) Return the first ChildStgKulturaEsSport filtered by the filmszinhaz column

\* @method ChildStgKulturaEsSport findOneByKozmuvelodesiIntezmeny(boolean $kozmuvelodesi\_intezmeny) Return the first ChildStgKulturaEsSport filtered by the kozmuvelodesi\_intezmeny column

\* @method ChildStgKulturaEsSport findOneByMuzeum(boolean $muzeum) Return the first ChildStgKulturaEsSport filtered by the muzeum column

\* @method ChildStgKulturaEsSport findOneByPiac(boolean $piac) Return the first ChildStgKulturaEsSport filtered by the piac column

\* @method ChildStgKulturaEsSport findOneBySportcsarnokSportpalya(boolean $sportcsarnok\_sportpalya) Return the first ChildStgKulturaEsSport filtered by the sportcsarnok\_sportpalya column

\* @method ChildStgKulturaEsSport findOneByStrand(boolean $strand) Return the first ChildStgKulturaEsSport filtered by the strand column

\* @method ChildStgKulturaEsSport findOneByTelepulesiKonyvtar(boolean $telepulesi\_konyvtar) Return the first ChildStgKulturaEsSport filtered by the telepulesi\_konyvtar column

\* @method ChildStgKulturaEsSport findOneByUszodaFurdoGyogyfurdo(boolean $uszoda\_furdo\_gyogyfurdo) Return the first ChildStgKulturaEsSport filtered by the uszoda\_furdo\_gyogyfurdo column

\*

\* @method ChildStgKulturaEsSport[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgKulturaEsSport objects based on current ModelCriteria

\* @method ChildStgKulturaEsSport[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgKulturaEsSport objects filtered by the telepules\_nev column

\* @method ChildStgKulturaEsSport[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgKulturaEsSport objects filtered by the telepules\_KSHKOD column

\* @method ChildStgKulturaEsSport[]|ObjectCollection findByFilmszinhaz(boolean $filmszinhaz) Return ChildStgKulturaEsSport objects filtered by the filmszinhaz column

\* @method ChildStgKulturaEsSport[]|ObjectCollection findByKozmuvelodesiIntezmeny(boolean $kozmuvelodesi\_intezmeny) Return ChildStgKulturaEsSport objects filtered by the kozmuvelodesi\_intezmeny column

\* @method ChildStgKulturaEsSport[]|ObjectCollection findByMuzeum(boolean $muzeum) Return ChildStgKulturaEsSport objects filtered by the muzeum column

\* @method ChildStgKulturaEsSport[]|ObjectCollection findByPiac(boolean $piac) Return ChildStgKulturaEsSport objects filtered by the piac column

\* @method ChildStgKulturaEsSport[]|ObjectCollection findBySportcsarnokSportpalya(boolean $sportcsarnok\_sportpalya) Return ChildStgKulturaEsSport objects filtered by the sportcsarnok\_sportpalya column

\* @method ChildStgKulturaEsSport[]|ObjectCollection findByStrand(boolean $strand) Return ChildStgKulturaEsSport objects filtered by the strand column

\* @method ChildStgKulturaEsSport[]|ObjectCollection findByTelepulesiKonyvtar(boolean $telepulesi\_konyvtar) Return ChildStgKulturaEsSport objects filtered by the telepulesi\_konyvtar column

\* @method ChildStgKulturaEsSport[]|ObjectCollection findByUszodaFurdoGyogyfurdo(boolean $uszoda\_furdo\_gyogyfurdo) Return ChildStgKulturaEsSport objects filtered by the uszoda\_furdo\_gyogyfurdo column

\* @method ChildStgKulturaEsSport[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgKulturaEsSportQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgKulturaEsSportQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgKulturaEsSport', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgKulturaEsSportQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgKulturaEsSportQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgKulturaEsSportQuery) {

return $criteria;

}

$query = new ChildStgKulturaEsSportQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgKulturaEsSport|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgKulturaEsSport object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgKulturaEsSport object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgKulturaEsSportQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgKulturaEsSport object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgKulturaEsSportQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgKulturaEsSport object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKulturaEsSportQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKulturaEsSportTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKulturaEsSportQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKulturaEsSportTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the filmszinhaz column

\*

\* Example usage:

\* <code>

\* $query->filterByFilmszinhaz(true); // WHERE filmszinhaz = true

\* $query->filterByFilmszinhaz('yes'); // WHERE filmszinhaz = true

\* </code>

\*

\* @param boolean|string $filmszinhaz The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKulturaEsSportQuery The current query, for fluid interface

\*/

public function filterByFilmszinhaz($filmszinhaz = null, $comparison = null)

{

if (is\_string($filmszinhaz)) {

$filmszinhaz = in\_array(strtolower($filmszinhaz), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgKulturaEsSportTableMap::COL\_FILMSZINHAZ, $filmszinhaz, $comparison);

}

/\*\*

\* Filter the query on the kozmuvelodesi\_intezmeny column

\*

\* Example usage:

\* <code>

\* $query->filterByKozmuvelodesiIntezmeny(true); // WHERE kozmuvelodesi\_intezmeny = true

\* $query->filterByKozmuvelodesiIntezmeny('yes'); // WHERE kozmuvelodesi\_intezmeny = true

\* </code>

\*

\* @param boolean|string $kozmuvelodesiIntezmeny The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKulturaEsSportQuery The current query, for fluid interface

\*/

public function filterByKozmuvelodesiIntezmeny($kozmuvelodesiIntezmeny = null, $comparison = null)

{

if (is\_string($kozmuvelodesiIntezmeny)) {

$kozmuvelodesiIntezmeny = in\_array(strtolower($kozmuvelodesiIntezmeny), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgKulturaEsSportTableMap::COL\_KOZMUVELODESI\_INTEZMENY, $kozmuvelodesiIntezmeny, $comparison);

}

/\*\*

\* Filter the query on the muzeum column

\*

\* Example usage:

\* <code>

\* $query->filterByMuzeum(true); // WHERE muzeum = true

\* $query->filterByMuzeum('yes'); // WHERE muzeum = true

\* </code>

\*

\* @param boolean|string $muzeum The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKulturaEsSportQuery The current query, for fluid interface

\*/

public function filterByMuzeum($muzeum = null, $comparison = null)

{

if (is\_string($muzeum)) {

$muzeum = in\_array(strtolower($muzeum), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgKulturaEsSportTableMap::COL\_MUZEUM, $muzeum, $comparison);

}

/\*\*

\* Filter the query on the piac column

\*

\* Example usage:

\* <code>

\* $query->filterByPiac(true); // WHERE piac = true

\* $query->filterByPiac('yes'); // WHERE piac = true

\* </code>

\*

\* @param boolean|string $piac The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKulturaEsSportQuery The current query, for fluid interface

\*/

public function filterByPiac($piac = null, $comparison = null)

{

if (is\_string($piac)) {

$piac = in\_array(strtolower($piac), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgKulturaEsSportTableMap::COL\_PIAC, $piac, $comparison);

}

/\*\*

\* Filter the query on the sportcsarnok\_sportpalya column

\*

\* Example usage:

\* <code>

\* $query->filterBySportcsarnokSportpalya(true); // WHERE sportcsarnok\_sportpalya = true

\* $query->filterBySportcsarnokSportpalya('yes'); // WHERE sportcsarnok\_sportpalya = true

\* </code>

\*

\* @param boolean|string $sportcsarnokSportpalya The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKulturaEsSportQuery The current query, for fluid interface

\*/

public function filterBySportcsarnokSportpalya($sportcsarnokSportpalya = null, $comparison = null)

{

if (is\_string($sportcsarnokSportpalya)) {

$sportcsarnokSportpalya = in\_array(strtolower($sportcsarnokSportpalya), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgKulturaEsSportTableMap::COL\_SPORTCSARNOK\_SPORTPALYA, $sportcsarnokSportpalya, $comparison);

}

/\*\*

\* Filter the query on the strand column

\*

\* Example usage:

\* <code>

\* $query->filterByStrand(true); // WHERE strand = true

\* $query->filterByStrand('yes'); // WHERE strand = true

\* </code>

\*

\* @param boolean|string $strand The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKulturaEsSportQuery The current query, for fluid interface

\*/

public function filterByStrand($strand = null, $comparison = null)

{

if (is\_string($strand)) {

$strand = in\_array(strtolower($strand), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgKulturaEsSportTableMap::COL\_STRAND, $strand, $comparison);

}

/\*\*

\* Filter the query on the telepulesi\_konyvtar column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesiKonyvtar(true); // WHERE telepulesi\_konyvtar = true

\* $query->filterByTelepulesiKonyvtar('yes'); // WHERE telepulesi\_konyvtar = true

\* </code>

\*

\* @param boolean|string $telepulesiKonyvtar The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKulturaEsSportQuery The current query, for fluid interface

\*/

public function filterByTelepulesiKonyvtar($telepulesiKonyvtar = null, $comparison = null)

{

if (is\_string($telepulesiKonyvtar)) {

$telepulesiKonyvtar = in\_array(strtolower($telepulesiKonyvtar), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgKulturaEsSportTableMap::COL\_TELEPULESI\_KONYVTAR, $telepulesiKonyvtar, $comparison);

}

/\*\*

\* Filter the query on the uszoda\_furdo\_gyogyfurdo column

\*

\* Example usage:

\* <code>

\* $query->filterByUszodaFurdoGyogyfurdo(true); // WHERE uszoda\_furdo\_gyogyfurdo = true

\* $query->filterByUszodaFurdoGyogyfurdo('yes'); // WHERE uszoda\_furdo\_gyogyfurdo = true

\* </code>

\*

\* @param boolean|string $uszodaFurdoGyogyfurdo The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKulturaEsSportQuery The current query, for fluid interface

\*/

public function filterByUszodaFurdoGyogyfurdo($uszodaFurdoGyogyfurdo = null, $comparison = null)

{

if (is\_string($uszodaFurdoGyogyfurdo)) {

$uszodaFurdoGyogyfurdo = in\_array(strtolower($uszodaFurdoGyogyfurdo), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgKulturaEsSportTableMap::COL\_USZODA\_FURDO\_GYOGYFURDO, $uszodaFurdoGyogyfurdo, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgKulturaEsSport $stgKulturaEsSport Object to remove from the list of results

\*

\* @return $this|ChildStgKulturaEsSportQuery The current query, for fluid interface

\*/

public function prune($stgKulturaEsSport = null)

{

if ($stgKulturaEsSport) {

throw new LogicException('StgKulturaEsSport object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_kultura\_es\_sport table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKulturaEsSportTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgKulturaEsSportTableMap::clearInstancePool();

StgKulturaEsSportTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKulturaEsSportTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgKulturaEsSportTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgKulturaEsSportTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgKulturaEsSportTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgKulturaEsSportQuery

#### StgKulturalisFunkciokAranya.php

<?php

namespace Base;

use \StgKulturalisFunkciokAranyaQuery as ChildStgKulturalisFunkciokAranyaQuery;

use \Exception;

use \PDO;

use Map\StgKulturalisFunkciokAranyaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgKulturalisFunkciokAranya implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgKulturalisFunkciokAranyaTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the kulturalis\_rendezvenyeken\_resztvevok\_szama field.

\* @var int

\*/

protected $kulturalis\_rendezvenyeken\_resztvevok\_szama;

/\*\*

\* The value for the mozilatogatasok\_szama field.

\* @var int

\*/

protected $mozilatogatasok\_szama;

/\*\*

\* The value for the muzeumi\_latogatok\_szama field.

\* @var int

\*/

protected $muzeumi\_latogatok\_szama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgKulturalisFunkciokAranya object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgKulturalisFunkciokAranya</code> instance. If

\* <code>obj</code> is an instance of <code>StgKulturalisFunkciokAranya</code>, delegates to

\* <code>equals(StgKulturalisFunkciokAranya)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgKulturalisFunkciokAranya The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [kulturalis\_rendezvenyeken\_resztvevok\_szama] column value.

\*

\* @return int

\*/

public function getKulturalisRendezvenyekenResztvevokSzama()

{

return $this->kulturalis\_rendezvenyeken\_resztvevok\_szama;

}

/\*\*

\* Get the [mozilatogatasok\_szama] column value.

\*

\* @return int

\*/

public function getMozilatogatasokSzama()

{

return $this->mozilatogatasok\_szama;

}

/\*\*

\* Get the [muzeumi\_latogatok\_szama] column value.

\*

\* @return int

\*/

public function getMuzeumiLatogatokSzama()

{

return $this->muzeumi\_latogatok\_szama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgKulturalisFunkciokAranyaTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgKulturalisFunkciokAranyaTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgKulturalisFunkciokAranyaTableMap::translateFieldName('KulturalisRendezvenyekenResztvevokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kulturalis\_rendezvenyeken\_resztvevok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgKulturalisFunkciokAranyaTableMap::translateFieldName('MozilatogatasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->mozilatogatasok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgKulturalisFunkciokAranyaTableMap::translateFieldName('MuzeumiLatogatokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->muzeumi\_latogatok\_szama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 5; // 5 = StgKulturalisFunkciokAranyaTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgKulturalisFunkciokAranya'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgKulturalisFunkciokAranya The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgKulturalisFunkciokAranyaTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgKulturalisFunkciokAranya The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgKulturalisFunkciokAranyaTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [kulturalis\_rendezvenyeken\_resztvevok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgKulturalisFunkciokAranya The current object (for fluent API support)

\*/

public function setKulturalisRendezvenyekenResztvevokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->kulturalis\_rendezvenyeken\_resztvevok\_szama !== $v) {

$this->kulturalis\_rendezvenyeken\_resztvevok\_szama = $v;

$this->modifiedColumns[StgKulturalisFunkciokAranyaTableMap::COL\_KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA] = true;

}

return $this;

} // setKulturalisRendezvenyekenResztvevokSzama()

/\*\*

\* Set the value of [mozilatogatasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgKulturalisFunkciokAranya The current object (for fluent API support)

\*/

public function setMozilatogatasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->mozilatogatasok\_szama !== $v) {

$this->mozilatogatasok\_szama = $v;

$this->modifiedColumns[StgKulturalisFunkciokAranyaTableMap::COL\_MOZILATOGATASOK\_SZAMA] = true;

}

return $this;

} // setMozilatogatasokSzama()

/\*\*

\* Set the value of [muzeumi\_latogatok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgKulturalisFunkciokAranya The current object (for fluent API support)

\*/

public function setMuzeumiLatogatokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->muzeumi\_latogatok\_szama !== $v) {

$this->muzeumi\_latogatok\_szama = $v;

$this->modifiedColumns[StgKulturalisFunkciokAranyaTableMap::COL\_MUZEUMI\_LATOGATOK\_SZAMA] = true;

}

return $this;

} // setMuzeumiLatogatokSzama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgKulturalisFunkciokAranyaTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgKulturalisFunkciokAranyaQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgKulturalisFunkciokAranya::setDeleted()

\* @see StgKulturalisFunkciokAranya::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKulturalisFunkciokAranyaTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgKulturalisFunkciokAranyaQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKulturalisFunkciokAranyaTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgKulturalisFunkciokAranyaTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgKulturalisFunkciokAranyaTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgKulturalisFunkciokAranyaTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgKulturalisFunkciokAranyaTableMap::COL\_KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA';

}

if ($this->isColumnModified(StgKulturalisFunkciokAranyaTableMap::COL\_MOZILATOGATASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'MOZILATOGATASOK\_SZAMA';

}

if ($this->isColumnModified(StgKulturalisFunkciokAranyaTableMap::COL\_MUZEUMI\_LATOGATOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'MUZEUMI\_LATOGATOK\_SZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_kulturalis\_funkciok\_aranya (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA':

$stmt->bindValue($identifier, $this->kulturalis\_rendezvenyeken\_resztvevok\_szama, PDO::PARAM\_INT);

break;

case 'MOZILATOGATASOK\_SZAMA':

$stmt->bindValue($identifier, $this->mozilatogatasok\_szama, PDO::PARAM\_INT);

break;

case 'MUZEUMI\_LATOGATOK\_SZAMA':

$stmt->bindValue($identifier, $this->muzeumi\_latogatok\_szama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgKulturalisFunkciokAranyaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getKulturalisRendezvenyekenResztvevokSzama();

break;

case 3:

return $this->getMozilatogatasokSzama();

break;

case 4:

return $this->getMuzeumiLatogatokSzama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgKulturalisFunkciokAranya'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgKulturalisFunkciokAranya'][$this->getPrimaryKey()] = true;

$keys = StgKulturalisFunkciokAranyaTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getKulturalisRendezvenyekenResztvevokSzama(),

$keys[3] => $this->getMozilatogatasokSzama(),

$keys[4] => $this->getMuzeumiLatogatokSzama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgKulturalisFunkciokAranya

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgKulturalisFunkciokAranyaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgKulturalisFunkciokAranya

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setKulturalisRendezvenyekenResztvevokSzama($value);

break;

case 3:

$this->setMozilatogatasokSzama($value);

break;

case 4:

$this->setMuzeumiLatogatokSzama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgKulturalisFunkciokAranyaTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setKulturalisRendezvenyekenResztvevokSzama($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setMozilatogatasokSzama($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setMuzeumiLatogatokSzama($arr[$keys[4]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgKulturalisFunkciokAranya The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgKulturalisFunkciokAranyaTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgKulturalisFunkciokAranyaTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgKulturalisFunkciokAranyaTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgKulturalisFunkciokAranyaTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgKulturalisFunkciokAranyaTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgKulturalisFunkciokAranyaTableMap::COL\_KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA)) {

$criteria->add(StgKulturalisFunkciokAranyaTableMap::COL\_KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA, $this->kulturalis\_rendezvenyeken\_resztvevok\_szama);

}

if ($this->isColumnModified(StgKulturalisFunkciokAranyaTableMap::COL\_MOZILATOGATASOK\_SZAMA)) {

$criteria->add(StgKulturalisFunkciokAranyaTableMap::COL\_MOZILATOGATASOK\_SZAMA, $this->mozilatogatasok\_szama);

}

if ($this->isColumnModified(StgKulturalisFunkciokAranyaTableMap::COL\_MUZEUMI\_LATOGATOK\_SZAMA)) {

$criteria->add(StgKulturalisFunkciokAranyaTableMap::COL\_MUZEUMI\_LATOGATOK\_SZAMA, $this->muzeumi\_latogatok\_szama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgKulturalisFunkciokAranya object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgKulturalisFunkciokAranya (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setKulturalisRendezvenyekenResztvevokSzama($this->getKulturalisRendezvenyekenResztvevokSzama());

$copyObj->setMozilatogatasokSzama($this->getMozilatogatasokSzama());

$copyObj->setMuzeumiLatogatokSzama($this->getMuzeumiLatogatokSzama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgKulturalisFunkciokAranya Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->kulturalis\_rendezvenyeken\_resztvevok\_szama = null;

$this->mozilatogatasok\_szama = null;

$this->muzeumi\_latogatok\_szama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgKulturalisFunkciokAranyaTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgKulturalisFunkciokAranyaQuery.php

<?php

namespace Base;

use \StgKulturalisFunkciokAranya as ChildStgKulturalisFunkciokAranya;

use \StgKulturalisFunkciokAranyaQuery as ChildStgKulturalisFunkciokAranyaQuery;

use \Exception;

use Map\StgKulturalisFunkciokAranyaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_kulturalis\_funkciok\_aranya' table.

\*

\*

\*

\* @method ChildStgKulturalisFunkciokAranyaQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgKulturalisFunkciokAranyaQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgKulturalisFunkciokAranyaQuery orderByKulturalisRendezvenyekenResztvevokSzama($order = Criteria::ASC) Order by the kulturalis\_rendezvenyeken\_resztvevok\_szama column

\* @method ChildStgKulturalisFunkciokAranyaQuery orderByMozilatogatasokSzama($order = Criteria::ASC) Order by the mozilatogatasok\_szama column

\* @method ChildStgKulturalisFunkciokAranyaQuery orderByMuzeumiLatogatokSzama($order = Criteria::ASC) Order by the muzeumi\_latogatok\_szama column

\*

\* @method ChildStgKulturalisFunkciokAranyaQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgKulturalisFunkciokAranyaQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgKulturalisFunkciokAranyaQuery groupByKulturalisRendezvenyekenResztvevokSzama() Group by the kulturalis\_rendezvenyeken\_resztvevok\_szama column

\* @method ChildStgKulturalisFunkciokAranyaQuery groupByMozilatogatasokSzama() Group by the mozilatogatasok\_szama column

\* @method ChildStgKulturalisFunkciokAranyaQuery groupByMuzeumiLatogatokSzama() Group by the muzeumi\_latogatok\_szama column

\*

\* @method ChildStgKulturalisFunkciokAranyaQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgKulturalisFunkciokAranyaQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgKulturalisFunkciokAranyaQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgKulturalisFunkciokAranya findOne(ConnectionInterface $con = null) Return the first ChildStgKulturalisFunkciokAranya matching the query

\* @method ChildStgKulturalisFunkciokAranya findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgKulturalisFunkciokAranya matching the query, or a new ChildStgKulturalisFunkciokAranya object populated from the query conditions when no match is found

\*

\* @method ChildStgKulturalisFunkciokAranya findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgKulturalisFunkciokAranya filtered by the telepules\_nev column

\* @method ChildStgKulturalisFunkciokAranya findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgKulturalisFunkciokAranya filtered by the telepules\_KSHKOD column

\* @method ChildStgKulturalisFunkciokAranya findOneByKulturalisRendezvenyekenResztvevokSzama(int $kulturalis\_rendezvenyeken\_resztvevok\_szama) Return the first ChildStgKulturalisFunkciokAranya filtered by the kulturalis\_rendezvenyeken\_resztvevok\_szama column

\* @method ChildStgKulturalisFunkciokAranya findOneByMozilatogatasokSzama(int $mozilatogatasok\_szama) Return the first ChildStgKulturalisFunkciokAranya filtered by the mozilatogatasok\_szama column

\* @method ChildStgKulturalisFunkciokAranya findOneByMuzeumiLatogatokSzama(int $muzeumi\_latogatok\_szama) Return the first ChildStgKulturalisFunkciokAranya filtered by the muzeumi\_latogatok\_szama column

\*

\* @method ChildStgKulturalisFunkciokAranya[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgKulturalisFunkciokAranya objects based on current ModelCriteria

\* @method ChildStgKulturalisFunkciokAranya[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgKulturalisFunkciokAranya objects filtered by the telepules\_nev column

\* @method ChildStgKulturalisFunkciokAranya[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgKulturalisFunkciokAranya objects filtered by the telepules\_KSHKOD column

\* @method ChildStgKulturalisFunkciokAranya[]|ObjectCollection findByKulturalisRendezvenyekenResztvevokSzama(int $kulturalis\_rendezvenyeken\_resztvevok\_szama) Return ChildStgKulturalisFunkciokAranya objects filtered by the kulturalis\_rendezvenyeken\_resztvevok\_szama column

\* @method ChildStgKulturalisFunkciokAranya[]|ObjectCollection findByMozilatogatasokSzama(int $mozilatogatasok\_szama) Return ChildStgKulturalisFunkciokAranya objects filtered by the mozilatogatasok\_szama column

\* @method ChildStgKulturalisFunkciokAranya[]|ObjectCollection findByMuzeumiLatogatokSzama(int $muzeumi\_latogatok\_szama) Return ChildStgKulturalisFunkciokAranya objects filtered by the muzeumi\_latogatok\_szama column

\* @method ChildStgKulturalisFunkciokAranya[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgKulturalisFunkciokAranyaQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgKulturalisFunkciokAranyaQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgKulturalisFunkciokAranya', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgKulturalisFunkciokAranyaQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgKulturalisFunkciokAranyaQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgKulturalisFunkciokAranyaQuery) {

return $criteria;

}

$query = new ChildStgKulturalisFunkciokAranyaQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgKulturalisFunkciokAranya|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgKulturalisFunkciokAranya object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgKulturalisFunkciokAranya object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgKulturalisFunkciokAranyaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgKulturalisFunkciokAranya object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgKulturalisFunkciokAranyaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgKulturalisFunkciokAranya object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKulturalisFunkciokAranyaQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKulturalisFunkciokAranyaTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKulturalisFunkciokAranyaQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgKulturalisFunkciokAranyaTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the kulturalis\_rendezvenyeken\_resztvevok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByKulturalisRendezvenyekenResztvevokSzama(1234); // WHERE kulturalis\_rendezvenyeken\_resztvevok\_szama = 1234

\* $query->filterByKulturalisRendezvenyekenResztvevokSzama(array(12, 34)); // WHERE kulturalis\_rendezvenyeken\_resztvevok\_szama IN (12, 34)

\* $query->filterByKulturalisRendezvenyekenResztvevokSzama(array('min' => 12)); // WHERE kulturalis\_rendezvenyeken\_resztvevok\_szama > 12

\* </code>

\*

\* @param mixed $kulturalisRendezvenyekenResztvevokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKulturalisFunkciokAranyaQuery The current query, for fluid interface

\*/

public function filterByKulturalisRendezvenyekenResztvevokSzama($kulturalisRendezvenyekenResztvevokSzama = null, $comparison = null)

{

if (is\_array($kulturalisRendezvenyekenResztvevokSzama)) {

$useMinMax = false;

if (isset($kulturalisRendezvenyekenResztvevokSzama['min'])) {

$this->addUsingAlias(StgKulturalisFunkciokAranyaTableMap::COL\_KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA, $kulturalisRendezvenyekenResztvevokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($kulturalisRendezvenyekenResztvevokSzama['max'])) {

$this->addUsingAlias(StgKulturalisFunkciokAranyaTableMap::COL\_KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA, $kulturalisRendezvenyekenResztvevokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKulturalisFunkciokAranyaTableMap::COL\_KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA, $kulturalisRendezvenyekenResztvevokSzama, $comparison);

}

/\*\*

\* Filter the query on the mozilatogatasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByMozilatogatasokSzama(1234); // WHERE mozilatogatasok\_szama = 1234

\* $query->filterByMozilatogatasokSzama(array(12, 34)); // WHERE mozilatogatasok\_szama IN (12, 34)

\* $query->filterByMozilatogatasokSzama(array('min' => 12)); // WHERE mozilatogatasok\_szama > 12

\* </code>

\*

\* @param mixed $mozilatogatasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKulturalisFunkciokAranyaQuery The current query, for fluid interface

\*/

public function filterByMozilatogatasokSzama($mozilatogatasokSzama = null, $comparison = null)

{

if (is\_array($mozilatogatasokSzama)) {

$useMinMax = false;

if (isset($mozilatogatasokSzama['min'])) {

$this->addUsingAlias(StgKulturalisFunkciokAranyaTableMap::COL\_MOZILATOGATASOK\_SZAMA, $mozilatogatasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($mozilatogatasokSzama['max'])) {

$this->addUsingAlias(StgKulturalisFunkciokAranyaTableMap::COL\_MOZILATOGATASOK\_SZAMA, $mozilatogatasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKulturalisFunkciokAranyaTableMap::COL\_MOZILATOGATASOK\_SZAMA, $mozilatogatasokSzama, $comparison);

}

/\*\*

\* Filter the query on the muzeumi\_latogatok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByMuzeumiLatogatokSzama(1234); // WHERE muzeumi\_latogatok\_szama = 1234

\* $query->filterByMuzeumiLatogatokSzama(array(12, 34)); // WHERE muzeumi\_latogatok\_szama IN (12, 34)

\* $query->filterByMuzeumiLatogatokSzama(array('min' => 12)); // WHERE muzeumi\_latogatok\_szama > 12

\* </code>

\*

\* @param mixed $muzeumiLatogatokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgKulturalisFunkciokAranyaQuery The current query, for fluid interface

\*/

public function filterByMuzeumiLatogatokSzama($muzeumiLatogatokSzama = null, $comparison = null)

{

if (is\_array($muzeumiLatogatokSzama)) {

$useMinMax = false;

if (isset($muzeumiLatogatokSzama['min'])) {

$this->addUsingAlias(StgKulturalisFunkciokAranyaTableMap::COL\_MUZEUMI\_LATOGATOK\_SZAMA, $muzeumiLatogatokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($muzeumiLatogatokSzama['max'])) {

$this->addUsingAlias(StgKulturalisFunkciokAranyaTableMap::COL\_MUZEUMI\_LATOGATOK\_SZAMA, $muzeumiLatogatokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgKulturalisFunkciokAranyaTableMap::COL\_MUZEUMI\_LATOGATOK\_SZAMA, $muzeumiLatogatokSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgKulturalisFunkciokAranya $stgKulturalisFunkciokAranya Object to remove from the list of results

\*

\* @return $this|ChildStgKulturalisFunkciokAranyaQuery The current query, for fluid interface

\*/

public function prune($stgKulturalisFunkciokAranya = null)

{

if ($stgKulturalisFunkciokAranya) {

throw new LogicException('StgKulturalisFunkciokAranya object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_kulturalis\_funkciok\_aranya table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKulturalisFunkciokAranyaTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgKulturalisFunkciokAranyaTableMap::clearInstancePool();

StgKulturalisFunkciokAranyaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKulturalisFunkciokAranyaTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgKulturalisFunkciokAranyaTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgKulturalisFunkciokAranyaTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgKulturalisFunkciokAranyaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgKulturalisFunkciokAranyaQuery

#### StgLakonepesseg.php

<?php

namespace Base;

use \StgLakonepessegQuery as ChildStgLakonepessegQuery;

use \Exception;

use \PDO;

use Map\StgLakonepessegTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgLakonepesseg implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgLakonepessegTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the nepesseg field.

\* @var int

\*/

protected $nepesseg;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgLakonepesseg object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgLakonepesseg</code> instance. If

\* <code>obj</code> is an instance of <code>StgLakonepesseg</code>, delegates to

\* <code>equals(StgLakonepesseg)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgLakonepesseg The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [nepesseg] column value.

\*

\* @return int

\*/

public function getNepesseg()

{

return $this->nepesseg;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgLakonepessegTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgLakonepessegTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgLakonepessegTableMap::translateFieldName('Nepesseg', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nepesseg = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 3; // 3 = StgLakonepessegTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgLakonepesseg'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgLakonepesseg The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgLakonepessegTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgLakonepesseg The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgLakonepessegTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [nepesseg] column.

\*

\* @param int $v new value

\* @return $this|\StgLakonepesseg The current object (for fluent API support)

\*/

public function setNepesseg($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nepesseg !== $v) {

$this->nepesseg = $v;

$this->modifiedColumns[StgLakonepessegTableMap::COL\_NEPESSEG] = true;

}

return $this;

} // setNepesseg()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgLakonepessegTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgLakonepessegQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgLakonepesseg::setDeleted()

\* @see StgLakonepesseg::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgLakonepessegTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgLakonepessegQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgLakonepessegTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgLakonepessegTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgLakonepessegTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgLakonepessegTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgLakonepessegTableMap::COL\_NEPESSEG)) {

$modifiedColumns[':p' . $index++] = 'NEPESSEG';

}

$sql = sprintf(

'INSERT INTO stg\_lakonepesseg (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'NEPESSEG':

$stmt->bindValue($identifier, $this->nepesseg, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgLakonepessegTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getNepesseg();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgLakonepesseg'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgLakonepesseg'][$this->getPrimaryKey()] = true;

$keys = StgLakonepessegTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getNepesseg(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgLakonepesseg

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgLakonepessegTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgLakonepesseg

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setNepesseg($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgLakonepessegTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setNepesseg($arr[$keys[2]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgLakonepesseg The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgLakonepessegTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgLakonepessegTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgLakonepessegTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgLakonepessegTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgLakonepessegTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgLakonepessegTableMap::COL\_NEPESSEG)) {

$criteria->add(StgLakonepessegTableMap::COL\_NEPESSEG, $this->nepesseg);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgLakonepesseg object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgLakonepesseg (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setNepesseg($this->getNepesseg());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgLakonepesseg Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->nepesseg = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgLakonepessegTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgLakonepessegQuery.php

<?php

namespace Base;

use \StgLakonepesseg as ChildStgLakonepesseg;

use \StgLakonepessegQuery as ChildStgLakonepessegQuery;

use \Exception;

use Map\StgLakonepessegTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_lakonepesseg' table.

\*

\*

\*

\* @method ChildStgLakonepessegQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgLakonepessegQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgLakonepessegQuery orderByNepesseg($order = Criteria::ASC) Order by the nepesseg column

\*

\* @method ChildStgLakonepessegQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgLakonepessegQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgLakonepessegQuery groupByNepesseg() Group by the nepesseg column

\*

\* @method ChildStgLakonepessegQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgLakonepessegQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgLakonepessegQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgLakonepesseg findOne(ConnectionInterface $con = null) Return the first ChildStgLakonepesseg matching the query

\* @method ChildStgLakonepesseg findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgLakonepesseg matching the query, or a new ChildStgLakonepesseg object populated from the query conditions when no match is found

\*

\* @method ChildStgLakonepesseg findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgLakonepesseg filtered by the telepules\_nev column

\* @method ChildStgLakonepesseg findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgLakonepesseg filtered by the telepules\_KSHKOD column

\* @method ChildStgLakonepesseg findOneByNepesseg(int $nepesseg) Return the first ChildStgLakonepesseg filtered by the nepesseg column

\*

\* @method ChildStgLakonepesseg[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgLakonepesseg objects based on current ModelCriteria

\* @method ChildStgLakonepesseg[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgLakonepesseg objects filtered by the telepules\_nev column

\* @method ChildStgLakonepesseg[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgLakonepesseg objects filtered by the telepules\_KSHKOD column

\* @method ChildStgLakonepesseg[]|ObjectCollection findByNepesseg(int $nepesseg) Return ChildStgLakonepesseg objects filtered by the nepesseg column

\* @method ChildStgLakonepesseg[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgLakonepessegQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgLakonepessegQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgLakonepesseg', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgLakonepessegQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgLakonepessegQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgLakonepessegQuery) {

return $criteria;

}

$query = new ChildStgLakonepessegQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgLakonepesseg|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgLakonepesseg object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgLakonepesseg object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgLakonepessegQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgLakonepesseg object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgLakonepessegQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgLakonepesseg object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgLakonepessegQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgLakonepessegTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgLakonepessegQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgLakonepessegTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the nepesseg column

\*

\* Example usage:

\* <code>

\* $query->filterByNepesseg(1234); // WHERE nepesseg = 1234

\* $query->filterByNepesseg(array(12, 34)); // WHERE nepesseg IN (12, 34)

\* $query->filterByNepesseg(array('min' => 12)); // WHERE nepesseg > 12

\* </code>

\*

\* @param mixed $nepesseg The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgLakonepessegQuery The current query, for fluid interface

\*/

public function filterByNepesseg($nepesseg = null, $comparison = null)

{

if (is\_array($nepesseg)) {

$useMinMax = false;

if (isset($nepesseg['min'])) {

$this->addUsingAlias(StgLakonepessegTableMap::COL\_NEPESSEG, $nepesseg['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nepesseg['max'])) {

$this->addUsingAlias(StgLakonepessegTableMap::COL\_NEPESSEG, $nepesseg['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgLakonepessegTableMap::COL\_NEPESSEG, $nepesseg, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgLakonepesseg $stgLakonepesseg Object to remove from the list of results

\*

\* @return $this|ChildStgLakonepessegQuery The current query, for fluid interface

\*/

public function prune($stgLakonepesseg = null)

{

if ($stgLakonepesseg) {

throw new LogicException('StgLakonepesseg object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_lakonepesseg table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgLakonepessegTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgLakonepessegTableMap::clearInstancePool();

StgLakonepessegTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgLakonepessegTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgLakonepessegTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgLakonepessegTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgLakonepessegTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgLakonepessegQuery

#### StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok.php

<?php

namespace Base;

use \StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery as ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery;

use \Exception;

use \PDO;

use Map\StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama field.

\* @var int

\*/

protected $mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama;

/\*\*

\* The value for the mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama field.

\* @var int

\*/

protected $mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok</code> instance. If

\* <code>obj</code> is an instance of <code>StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok</code>, delegates to

\* <code>equals(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama()

{

return $this->mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama;

}

/\*\*

\* Get the [mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama()

{

return $this->mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::translateFieldName('MasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::translateFieldName('MasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 4; // 4 = StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok The current object (for fluent API support)

\*/

public function setMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama !== $v) {

$this->mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama = $v;

$this->modifiedColumns[StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA] = true;

}

return $this;

} // setMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama()

/\*\*

\* Set the value of [mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok The current object (for fluent API support)

\*/

public function setMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama !== $v) {

$this->mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama = $v;

$this->modifiedColumns[StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA] = true;

}

return $this;

} // setMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok::setDeleted()

\* @see StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA';

}

if ($this->isColumnModified(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_mas\_telepulesrol\_bejaro\_altalanos\_es\_kozepiskolasok (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama, PDO::PARAM\_INT);

break;

case 'MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama();

break;

case 3:

return $this->getMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok'][$this->getPrimaryKey()] = true;

$keys = StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama(),

$keys[3] => $this->getMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama($value);

break;

case 3:

$this->setMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama($arr[$keys[3]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA)) {

$criteria->add(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA, $this->mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama);

}

if ($this->isColumnModified(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA)) {

$criteria->add(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $this->mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama($this->getMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama());

$copyObj->setMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama($this->getMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama = null;

$this->mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery.php

<?php

namespace Base;

use \StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok as ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok;

use \StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery as ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery;

use \Exception;

use Map\StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_mas\_telepulesrol\_bejaro\_altalanos\_es\_kozepiskolasok' table.

\*

\*

\*

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery orderByMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama($order = Criteria::ASC) Order by the mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama column

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery orderByMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama($order = Criteria::ASC) Order by the mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama column

\*

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery groupByMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama() Group by the mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama column

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery groupByMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama() Group by the mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama column

\*

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok findOne(ConnectionInterface $con = null) Return the first ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok matching the query

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok matching the query, or a new ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok object populated from the query conditions when no match is found

\*

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok filtered by the telepules\_nev column

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok filtered by the telepules\_KSHKOD column

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok findOneByMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama(int $mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama) Return the first ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok filtered by the mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama column

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok findOneByMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama(int $mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama) Return the first ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok filtered by the mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama column

\*

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok objects based on current ModelCriteria

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok objects filtered by the telepules\_nev column

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok objects filtered by the telepules\_KSHKOD column

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok[]|ObjectCollection findByMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama(int $mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama) Return ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok objects filtered by the mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama column

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok[]|ObjectCollection findByMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama(int $mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama) Return ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok objects filtered by the mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery) {

return $criteria;

}

$query = new ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama(1234); // WHERE mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama = 1234

\* $query->filterByMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama(array(12, 34)); // WHERE mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama IN (12, 34)

\* $query->filterByMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama(array('min' => 12)); // WHERE mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery The current query, for fluid interface

\*/

public function filterByMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama($masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama = null, $comparison = null)

{

if (is\_array($masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama)) {

$useMinMax = false;

if (isset($masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama['min'])) {

$this->addUsingAlias(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA, $masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama['max'])) {

$this->addUsingAlias(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA, $masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA, $masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama, $comparison);

}

/\*\*

\* Filter the query on the mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama(1234); // WHERE mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama = 1234

\* $query->filterByMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama(array(12, 34)); // WHERE mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama IN (12, 34)

\* $query->filterByMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama(array('min' => 12)); // WHERE mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery The current query, for fluid interface

\*/

public function filterByMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama($masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama = null, $comparison = null)

{

if (is\_array($masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama)) {

$useMinMax = false;

if (isset($masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama['min'])) {

$this->addUsingAlias(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama['max'])) {

$this->addUsingAlias(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasok $stgMasTelepulesrolBejaroAltalanosEsKozepiskolasok Object to remove from the list of results

\*

\* @return $this|ChildStgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery The current query, for fluid interface

\*/

public function prune($stgMasTelepulesrolBejaroAltalanosEsKozepiskolasok = null)

{

if ($stgMasTelepulesrolBejaroAltalanosEsKozepiskolasok) {

throw new LogicException('StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_mas\_telepulesrol\_bejaro\_altalanos\_es\_kozepiskolasok table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::clearInstancePool();

StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery

#### StgMukodoVallalkozasokSzama.php

<?php

namespace Base;

use \StgMukodoVallalkozasokSzamaQuery as ChildStgMukodoVallalkozasokSzamaQuery;

use \Exception;

use \PDO;

use Map\StgMukodoVallalkozasokSzamaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgMukodoVallalkozasokSzama implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgMukodoVallalkozasokSzamaTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the vallalkozasok\_szama field.

\* @var int

\*/

protected $vallalkozasok\_szama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgMukodoVallalkozasokSzama object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgMukodoVallalkozasokSzama</code> instance. If

\* <code>obj</code> is an instance of <code>StgMukodoVallalkozasokSzama</code>, delegates to

\* <code>equals(StgMukodoVallalkozasokSzama)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgMukodoVallalkozasokSzama The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [vallalkozasok\_szama] column value.

\*

\* @return int

\*/

public function getVallalkozasokSzama()

{

return $this->vallalkozasok\_szama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgMukodoVallalkozasokSzamaTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgMukodoVallalkozasokSzamaTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgMukodoVallalkozasokSzamaTableMap::translateFieldName('VallalkozasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->vallalkozasok\_szama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 3; // 3 = StgMukodoVallalkozasokSzamaTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgMukodoVallalkozasokSzama'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgMukodoVallalkozasokSzama The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgMukodoVallalkozasokSzamaTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgMukodoVallalkozasokSzama The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgMukodoVallalkozasokSzamaTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [vallalkozasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgMukodoVallalkozasokSzama The current object (for fluent API support)

\*/

public function setVallalkozasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->vallalkozasok\_szama !== $v) {

$this->vallalkozasok\_szama = $v;

$this->modifiedColumns[StgMukodoVallalkozasokSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA] = true;

}

return $this;

} // setVallalkozasokSzama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgMukodoVallalkozasokSzamaTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgMukodoVallalkozasokSzamaQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgMukodoVallalkozasokSzama::setDeleted()

\* @see StgMukodoVallalkozasokSzama::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMukodoVallalkozasokSzamaTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgMukodoVallalkozasokSzamaQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMukodoVallalkozasokSzamaTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgMukodoVallalkozasokSzamaTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'VALLALKOZASOK\_SZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_mukodo\_vallalkozasok\_szama (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'VALLALKOZASOK\_SZAMA':

$stmt->bindValue($identifier, $this->vallalkozasok\_szama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgMukodoVallalkozasokSzamaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getVallalkozasokSzama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgMukodoVallalkozasokSzama'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgMukodoVallalkozasokSzama'][$this->getPrimaryKey()] = true;

$keys = StgMukodoVallalkozasokSzamaTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getVallalkozasokSzama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgMukodoVallalkozasokSzama

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgMukodoVallalkozasokSzamaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgMukodoVallalkozasokSzama

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setVallalkozasokSzama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgMukodoVallalkozasokSzamaTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setVallalkozasokSzama($arr[$keys[2]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgMukodoVallalkozasokSzama The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgMukodoVallalkozasokSzamaTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgMukodoVallalkozasokSzamaTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgMukodoVallalkozasokSzamaTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA)) {

$criteria->add(StgMukodoVallalkozasokSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA, $this->vallalkozasok\_szama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgMukodoVallalkozasokSzama object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgMukodoVallalkozasokSzama (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setVallalkozasokSzama($this->getVallalkozasokSzama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgMukodoVallalkozasokSzama Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->vallalkozasok\_szama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgMukodoVallalkozasokSzamaTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgMukodoVallalkozasokSzamaAKiemeltIparagakban.php

<?php

namespace Base;

use \StgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery as ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery;

use \Exception;

use \PDO;

use Map\StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgMukodoVallalkozasokSzamaAKiemeltIparagakban implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the feldolgozoiparban\_mukodo\_vallalkozasok\_szama field.

\* @var int

\*/

protected $feldolgozoiparban\_mukodo\_vallalkozasok\_szama;

/\*\*

\* The value for the kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama field.

\* @var int

\*/

protected $kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama;

/\*\*

\* The value for the szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama field.

\* @var int

\*/

protected $szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama;

/\*\*

\* The value for the adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama field.

\* @var int

\*/

protected $adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama;

/\*\*

\* The value for the egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama field.

\* @var int

\*/

protected $egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgMukodoVallalkozasokSzamaAKiemeltIparagakban object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgMukodoVallalkozasokSzamaAKiemeltIparagakban</code> instance. If

\* <code>obj</code> is an instance of <code>StgMukodoVallalkozasokSzamaAKiemeltIparagakban</code>, delegates to

\* <code>equals(StgMukodoVallalkozasokSzamaAKiemeltIparagakban)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgMukodoVallalkozasokSzamaAKiemeltIparagakban The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [feldolgozoiparban\_mukodo\_vallalkozasok\_szama] column value.

\*

\* @return int

\*/

public function getFeldolgozoiparbanMukodoVallalkozasokSzama()

{

return $this->feldolgozoiparban\_mukodo\_vallalkozasok\_szama;

}

/\*\*

\* Get the [kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama] column value.

\*

\* @return int

\*/

public function getKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama()

{

return $this->kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama;

}

/\*\*

\* Get the [szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama] column value.

\*

\* @return int

\*/

public function getSzallitasbanRaktarozasbanMukodoVallalkozasokSzama()

{

return $this->szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama;

}

/\*\*

\* Get the [adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama] column value.

\*

\* @return int

\*/

public function getAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama()

{

return $this->adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama;

}

/\*\*

\* Get the [egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama] column value.

\*

\* @return int

\*/

public function getEgyebSzolgaltatasbanMukodoVallalkozasokSzama()

{

return $this->egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::translateFieldName('FeldolgozoiparbanMukodoVallalkozasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->feldolgozoiparban\_mukodo\_vallalkozasok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::translateFieldName('KereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::translateFieldName('SzallitasbanRaktarozasbanMukodoVallalkozasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 5 + $startcol : StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::translateFieldName('AdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 6 + $startcol : StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::translateFieldName('EgyebSzolgaltatasbanMukodoVallalkozasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 7; // 7 = StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgMukodoVallalkozasokSzamaAKiemeltIparagakban'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgMukodoVallalkozasokSzamaAKiemeltIparagakban The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgMukodoVallalkozasokSzamaAKiemeltIparagakban The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [feldolgozoiparban\_mukodo\_vallalkozasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgMukodoVallalkozasokSzamaAKiemeltIparagakban The current object (for fluent API support)

\*/

public function setFeldolgozoiparbanMukodoVallalkozasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->feldolgozoiparban\_mukodo\_vallalkozasok\_szama !== $v) {

$this->feldolgozoiparban\_mukodo\_vallalkozasok\_szama = $v;

$this->modifiedColumns[StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA] = true;

}

return $this;

} // setFeldolgozoiparbanMukodoVallalkozasokSzama()

/\*\*

\* Set the value of [kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgMukodoVallalkozasokSzamaAKiemeltIparagakban The current object (for fluent API support)

\*/

public function setKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama !== $v) {

$this->kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama = $v;

$this->modifiedColumns[StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA] = true;

}

return $this;

} // setKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama()

/\*\*

\* Set the value of [szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgMukodoVallalkozasokSzamaAKiemeltIparagakban The current object (for fluent API support)

\*/

public function setSzallitasbanRaktarozasbanMukodoVallalkozasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama !== $v) {

$this->szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama = $v;

$this->modifiedColumns[StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA] = true;

}

return $this;

} // setSzallitasbanRaktarozasbanMukodoVallalkozasokSzama()

/\*\*

\* Set the value of [adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgMukodoVallalkozasokSzamaAKiemeltIparagakban The current object (for fluent API support)

\*/

public function setAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama !== $v) {

$this->adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama = $v;

$this->modifiedColumns[StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA] = true;

}

return $this;

} // setAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama()

/\*\*

\* Set the value of [egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgMukodoVallalkozasokSzamaAKiemeltIparagakban The current object (for fluent API support)

\*/

public function setEgyebSzolgaltatasbanMukodoVallalkozasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama !== $v) {

$this->egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama = $v;

$this->modifiedColumns[StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA] = true;

}

return $this;

} // setEgyebSzolgaltatasbanMukodoVallalkozasokSzama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgMukodoVallalkozasokSzamaAKiemeltIparagakban::setDeleted()

\* @see StgMukodoVallalkozasokSzamaAKiemeltIparagakban::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA';

}

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA';

}

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA';

}

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA';

}

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA':

$stmt->bindValue($identifier, $this->feldolgozoiparban\_mukodo\_vallalkozasok\_szama, PDO::PARAM\_INT);

break;

case 'KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA':

$stmt->bindValue($identifier, $this->kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama, PDO::PARAM\_INT);

break;

case 'SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA':

$stmt->bindValue($identifier, $this->szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama, PDO::PARAM\_INT);

break;

case 'ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA':

$stmt->bindValue($identifier, $this->adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama, PDO::PARAM\_INT);

break;

case 'EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA':

$stmt->bindValue($identifier, $this->egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getFeldolgozoiparbanMukodoVallalkozasokSzama();

break;

case 3:

return $this->getKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama();

break;

case 4:

return $this->getSzallitasbanRaktarozasbanMukodoVallalkozasokSzama();

break;

case 5:

return $this->getAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama();

break;

case 6:

return $this->getEgyebSzolgaltatasbanMukodoVallalkozasokSzama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgMukodoVallalkozasokSzamaAKiemeltIparagakban'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgMukodoVallalkozasokSzamaAKiemeltIparagakban'][$this->getPrimaryKey()] = true;

$keys = StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getFeldolgozoiparbanMukodoVallalkozasokSzama(),

$keys[3] => $this->getKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama(),

$keys[4] => $this->getSzallitasbanRaktarozasbanMukodoVallalkozasokSzama(),

$keys[5] => $this->getAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama(),

$keys[6] => $this->getEgyebSzolgaltatasbanMukodoVallalkozasokSzama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgMukodoVallalkozasokSzamaAKiemeltIparagakban

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgMukodoVallalkozasokSzamaAKiemeltIparagakban

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setFeldolgozoiparbanMukodoVallalkozasokSzama($value);

break;

case 3:

$this->setKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama($value);

break;

case 4:

$this->setSzallitasbanRaktarozasbanMukodoVallalkozasokSzama($value);

break;

case 5:

$this->setAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama($value);

break;

case 6:

$this->setEgyebSzolgaltatasbanMukodoVallalkozasokSzama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setFeldolgozoiparbanMukodoVallalkozasokSzama($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setSzallitasbanRaktarozasbanMukodoVallalkozasokSzama($arr[$keys[4]]);

}

if (array\_key\_exists($keys[5], $arr)) {

$this->setAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama($arr[$keys[5]]);

}

if (array\_key\_exists($keys[6], $arr)) {

$this->setEgyebSzolgaltatasbanMukodoVallalkozasokSzama($arr[$keys[6]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgMukodoVallalkozasokSzamaAKiemeltIparagakban The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA)) {

$criteria->add(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $this->feldolgozoiparban\_mukodo\_vallalkozasok\_szama);

}

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA)) {

$criteria->add(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $this->kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama);

}

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA)) {

$criteria->add(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $this->szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama);

}

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA)) {

$criteria->add(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $this->adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama);

}

if ($this->isColumnModified(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA)) {

$criteria->add(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $this->egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgMukodoVallalkozasokSzamaAKiemeltIparagakban object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgMukodoVallalkozasokSzamaAKiemeltIparagakban (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setFeldolgozoiparbanMukodoVallalkozasokSzama($this->getFeldolgozoiparbanMukodoVallalkozasokSzama());

$copyObj->setKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama($this->getKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama());

$copyObj->setSzallitasbanRaktarozasbanMukodoVallalkozasokSzama($this->getSzallitasbanRaktarozasbanMukodoVallalkozasokSzama());

$copyObj->setAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama($this->getAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama());

$copyObj->setEgyebSzolgaltatasbanMukodoVallalkozasokSzama($this->getEgyebSzolgaltatasbanMukodoVallalkozasokSzama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgMukodoVallalkozasokSzamaAKiemeltIparagakban Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->feldolgozoiparban\_mukodo\_vallalkozasok\_szama = null;

$this->kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama = null;

$this->szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama = null;

$this->adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama = null;

$this->egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery.php

<?php

namespace Base;

use \StgMukodoVallalkozasokSzamaAKiemeltIparagakban as ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban;

use \StgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery as ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery;

use \Exception;

use Map\StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban' table.

\*

\*

\*

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery orderByFeldolgozoiparbanMukodoVallalkozasokSzama($order = Criteria::ASC) Order by the feldolgozoiparban\_mukodo\_vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery orderByKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama($order = Criteria::ASC) Order by the kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery orderBySzallitasbanRaktarozasbanMukodoVallalkozasokSzama($order = Criteria::ASC) Order by the szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery orderByAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama($order = Criteria::ASC) Order by the adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery orderByEgyebSzolgaltatasbanMukodoVallalkozasokSzama($order = Criteria::ASC) Order by the egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama column

\*

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery groupByFeldolgozoiparbanMukodoVallalkozasokSzama() Group by the feldolgozoiparban\_mukodo\_vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery groupByKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama() Group by the kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery groupBySzallitasbanRaktarozasbanMukodoVallalkozasokSzama() Group by the szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery groupByAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama() Group by the adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery groupByEgyebSzolgaltatasbanMukodoVallalkozasokSzama() Group by the egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama column

\*

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban findOne(ConnectionInterface $con = null) Return the first ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban matching the query

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban matching the query, or a new ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban object populated from the query conditions when no match is found

\*

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban filtered by the telepules\_nev column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban filtered by the telepules\_KSHKOD column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban findOneByFeldolgozoiparbanMukodoVallalkozasokSzama(int $feldolgozoiparban\_mukodo\_vallalkozasok\_szama) Return the first ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban filtered by the feldolgozoiparban\_mukodo\_vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban findOneByKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama(int $kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama) Return the first ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban filtered by the kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban findOneBySzallitasbanRaktarozasbanMukodoVallalkozasokSzama(int $szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama) Return the first ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban filtered by the szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban findOneByAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama(int $adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama) Return the first ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban filtered by the adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban findOneByEgyebSzolgaltatasbanMukodoVallalkozasokSzama(int $egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama) Return the first ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban filtered by the egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama column

\*

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban objects based on current ModelCriteria

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban objects filtered by the telepules\_nev column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban objects filtered by the telepules\_KSHKOD column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban[]|ObjectCollection findByFeldolgozoiparbanMukodoVallalkozasokSzama(int $feldolgozoiparban\_mukodo\_vallalkozasok\_szama) Return ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban objects filtered by the feldolgozoiparban\_mukodo\_vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban[]|ObjectCollection findByKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama(int $kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama) Return ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban objects filtered by the kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban[]|ObjectCollection findBySzallitasbanRaktarozasbanMukodoVallalkozasokSzama(int $szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama) Return ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban objects filtered by the szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban[]|ObjectCollection findByAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama(int $adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama) Return ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban objects filtered by the adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban[]|ObjectCollection findByEgyebSzolgaltatasbanMukodoVallalkozasokSzama(int $egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama) Return ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban objects filtered by the egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgMukodoVallalkozasokSzamaAKiemeltIparagakban', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery) {

return $criteria;

}

$query = new ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgMukodoVallalkozasokSzamaAKiemeltIparagakban object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgMukodoVallalkozasokSzamaAKiemeltIparagakban object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgMukodoVallalkozasokSzamaAKiemeltIparagakban object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgMukodoVallalkozasokSzamaAKiemeltIparagakban object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the feldolgozoiparban\_mukodo\_vallalkozasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByFeldolgozoiparbanMukodoVallalkozasokSzama(1234); // WHERE feldolgozoiparban\_mukodo\_vallalkozasok\_szama = 1234

\* $query->filterByFeldolgozoiparbanMukodoVallalkozasokSzama(array(12, 34)); // WHERE feldolgozoiparban\_mukodo\_vallalkozasok\_szama IN (12, 34)

\* $query->filterByFeldolgozoiparbanMukodoVallalkozasokSzama(array('min' => 12)); // WHERE feldolgozoiparban\_mukodo\_vallalkozasok\_szama > 12

\* </code>

\*

\* @param mixed $feldolgozoiparbanMukodoVallalkozasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery The current query, for fluid interface

\*/

public function filterByFeldolgozoiparbanMukodoVallalkozasokSzama($feldolgozoiparbanMukodoVallalkozasokSzama = null, $comparison = null)

{

if (is\_array($feldolgozoiparbanMukodoVallalkozasokSzama)) {

$useMinMax = false;

if (isset($feldolgozoiparbanMukodoVallalkozasokSzama['min'])) {

$this->addUsingAlias(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $feldolgozoiparbanMukodoVallalkozasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($feldolgozoiparbanMukodoVallalkozasokSzama['max'])) {

$this->addUsingAlias(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $feldolgozoiparbanMukodoVallalkozasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $feldolgozoiparbanMukodoVallalkozasokSzama, $comparison);

}

/\*\*

\* Filter the query on the kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama(1234); // WHERE kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama = 1234

\* $query->filterByKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama(array(12, 34)); // WHERE kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama IN (12, 34)

\* $query->filterByKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama(array('min' => 12)); // WHERE kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama > 12

\* </code>

\*

\* @param mixed $kereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery The current query, for fluid interface

\*/

public function filterByKereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama($kereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama = null, $comparison = null)

{

if (is\_array($kereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama)) {

$useMinMax = false;

if (isset($kereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama['min'])) {

$this->addUsingAlias(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $kereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($kereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama['max'])) {

$this->addUsingAlias(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $kereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $kereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama, $comparison);

}

/\*\*

\* Filter the query on the szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterBySzallitasbanRaktarozasbanMukodoVallalkozasokSzama(1234); // WHERE szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama = 1234

\* $query->filterBySzallitasbanRaktarozasbanMukodoVallalkozasokSzama(array(12, 34)); // WHERE szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama IN (12, 34)

\* $query->filterBySzallitasbanRaktarozasbanMukodoVallalkozasokSzama(array('min' => 12)); // WHERE szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama > 12

\* </code>

\*

\* @param mixed $szallitasbanRaktarozasbanMukodoVallalkozasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery The current query, for fluid interface

\*/

public function filterBySzallitasbanRaktarozasbanMukodoVallalkozasokSzama($szallitasbanRaktarozasbanMukodoVallalkozasokSzama = null, $comparison = null)

{

if (is\_array($szallitasbanRaktarozasbanMukodoVallalkozasokSzama)) {

$useMinMax = false;

if (isset($szallitasbanRaktarozasbanMukodoVallalkozasokSzama['min'])) {

$this->addUsingAlias(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $szallitasbanRaktarozasbanMukodoVallalkozasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($szallitasbanRaktarozasbanMukodoVallalkozasokSzama['max'])) {

$this->addUsingAlias(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $szallitasbanRaktarozasbanMukodoVallalkozasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $szallitasbanRaktarozasbanMukodoVallalkozasokSzama, $comparison);

}

/\*\*

\* Filter the query on the adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama(1234); // WHERE adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama = 1234

\* $query->filterByAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama(array(12, 34)); // WHERE adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama IN (12, 34)

\* $query->filterByAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama(array('min' => 12)); // WHERE adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama > 12

\* </code>

\*

\* @param mixed $adminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery The current query, for fluid interface

\*/

public function filterByAdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama($adminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama = null, $comparison = null)

{

if (is\_array($adminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama)) {

$useMinMax = false;

if (isset($adminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama['min'])) {

$this->addUsingAlias(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $adminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($adminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama['max'])) {

$this->addUsingAlias(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $adminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $adminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama, $comparison);

}

/\*\*

\* Filter the query on the egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByEgyebSzolgaltatasbanMukodoVallalkozasokSzama(1234); // WHERE egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama = 1234

\* $query->filterByEgyebSzolgaltatasbanMukodoVallalkozasokSzama(array(12, 34)); // WHERE egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama IN (12, 34)

\* $query->filterByEgyebSzolgaltatasbanMukodoVallalkozasokSzama(array('min' => 12)); // WHERE egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama > 12

\* </code>

\*

\* @param mixed $egyebSzolgaltatasbanMukodoVallalkozasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery The current query, for fluid interface

\*/

public function filterByEgyebSzolgaltatasbanMukodoVallalkozasokSzama($egyebSzolgaltatasbanMukodoVallalkozasokSzama = null, $comparison = null)

{

if (is\_array($egyebSzolgaltatasbanMukodoVallalkozasokSzama)) {

$useMinMax = false;

if (isset($egyebSzolgaltatasbanMukodoVallalkozasokSzama['min'])) {

$this->addUsingAlias(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $egyebSzolgaltatasbanMukodoVallalkozasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($egyebSzolgaltatasbanMukodoVallalkozasokSzama['max'])) {

$this->addUsingAlias(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $egyebSzolgaltatasbanMukodoVallalkozasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, $egyebSzolgaltatasbanMukodoVallalkozasokSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakban $stgMukodoVallalkozasokSzamaAKiemeltIparagakban Object to remove from the list of results

\*

\* @return $this|ChildStgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery The current query, for fluid interface

\*/

public function prune($stgMukodoVallalkozasokSzamaAKiemeltIparagakban = null)

{

if ($stgMukodoVallalkozasokSzamaAKiemeltIparagakban) {

throw new LogicException('StgMukodoVallalkozasokSzamaAKiemeltIparagakban object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::clearInstancePool();

StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery

#### StgMukodoVallalkozasokSzamaQuery.php

<?php

namespace Base;

use \StgMukodoVallalkozasokSzama as ChildStgMukodoVallalkozasokSzama;

use \StgMukodoVallalkozasokSzamaQuery as ChildStgMukodoVallalkozasokSzamaQuery;

use \Exception;

use Map\StgMukodoVallalkozasokSzamaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_mukodo\_vallalkozasok\_szama' table.

\*

\*

\*

\* @method ChildStgMukodoVallalkozasokSzamaQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgMukodoVallalkozasokSzamaQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgMukodoVallalkozasokSzamaQuery orderByVallalkozasokSzama($order = Criteria::ASC) Order by the vallalkozasok\_szama column

\*

\* @method ChildStgMukodoVallalkozasokSzamaQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgMukodoVallalkozasokSzamaQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgMukodoVallalkozasokSzamaQuery groupByVallalkozasokSzama() Group by the vallalkozasok\_szama column

\*

\* @method ChildStgMukodoVallalkozasokSzamaQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgMukodoVallalkozasokSzamaQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgMukodoVallalkozasokSzamaQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgMukodoVallalkozasokSzama findOne(ConnectionInterface $con = null) Return the first ChildStgMukodoVallalkozasokSzama matching the query

\* @method ChildStgMukodoVallalkozasokSzama findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgMukodoVallalkozasokSzama matching the query, or a new ChildStgMukodoVallalkozasokSzama object populated from the query conditions when no match is found

\*

\* @method ChildStgMukodoVallalkozasokSzama findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgMukodoVallalkozasokSzama filtered by the telepules\_nev column

\* @method ChildStgMukodoVallalkozasokSzama findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgMukodoVallalkozasokSzama filtered by the telepules\_KSHKOD column

\* @method ChildStgMukodoVallalkozasokSzama findOneByVallalkozasokSzama(int $vallalkozasok\_szama) Return the first ChildStgMukodoVallalkozasokSzama filtered by the vallalkozasok\_szama column

\*

\* @method ChildStgMukodoVallalkozasokSzama[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgMukodoVallalkozasokSzama objects based on current ModelCriteria

\* @method ChildStgMukodoVallalkozasokSzama[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgMukodoVallalkozasokSzama objects filtered by the telepules\_nev column

\* @method ChildStgMukodoVallalkozasokSzama[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgMukodoVallalkozasokSzama objects filtered by the telepules\_KSHKOD column

\* @method ChildStgMukodoVallalkozasokSzama[]|ObjectCollection findByVallalkozasokSzama(int $vallalkozasok\_szama) Return ChildStgMukodoVallalkozasokSzama objects filtered by the vallalkozasok\_szama column

\* @method ChildStgMukodoVallalkozasokSzama[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgMukodoVallalkozasokSzamaQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgMukodoVallalkozasokSzamaQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgMukodoVallalkozasokSzama', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgMukodoVallalkozasokSzamaQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgMukodoVallalkozasokSzamaQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgMukodoVallalkozasokSzamaQuery) {

return $criteria;

}

$query = new ChildStgMukodoVallalkozasokSzamaQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgMukodoVallalkozasokSzama|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgMukodoVallalkozasokSzama object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgMukodoVallalkozasokSzama object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgMukodoVallalkozasokSzamaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgMukodoVallalkozasokSzama object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgMukodoVallalkozasokSzamaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgMukodoVallalkozasokSzama object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgMukodoVallalkozasokSzamaQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgMukodoVallalkozasokSzamaTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgMukodoVallalkozasokSzamaQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgMukodoVallalkozasokSzamaTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the vallalkozasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByVallalkozasokSzama(1234); // WHERE vallalkozasok\_szama = 1234

\* $query->filterByVallalkozasokSzama(array(12, 34)); // WHERE vallalkozasok\_szama IN (12, 34)

\* $query->filterByVallalkozasokSzama(array('min' => 12)); // WHERE vallalkozasok\_szama > 12

\* </code>

\*

\* @param mixed $vallalkozasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgMukodoVallalkozasokSzamaQuery The current query, for fluid interface

\*/

public function filterByVallalkozasokSzama($vallalkozasokSzama = null, $comparison = null)

{

if (is\_array($vallalkozasokSzama)) {

$useMinMax = false;

if (isset($vallalkozasokSzama['min'])) {

$this->addUsingAlias(StgMukodoVallalkozasokSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA, $vallalkozasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($vallalkozasokSzama['max'])) {

$this->addUsingAlias(StgMukodoVallalkozasokSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA, $vallalkozasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgMukodoVallalkozasokSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA, $vallalkozasokSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgMukodoVallalkozasokSzama $stgMukodoVallalkozasokSzama Object to remove from the list of results

\*

\* @return $this|ChildStgMukodoVallalkozasokSzamaQuery The current query, for fluid interface

\*/

public function prune($stgMukodoVallalkozasokSzama = null)

{

if ($stgMukodoVallalkozasokSzama) {

throw new LogicException('StgMukodoVallalkozasokSzama object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_mukodo\_vallalkozasok\_szama table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMukodoVallalkozasokSzamaTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgMukodoVallalkozasokSzamaTableMap::clearInstancePool();

StgMukodoVallalkozasokSzamaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMukodoVallalkozasokSzamaTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgMukodoVallalkozasokSzamaTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgMukodoVallalkozasokSzamaTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgMukodoVallalkozasokSzamaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgMukodoVallalkozasokSzamaQuery

#### StgMunkakepesKoruNepesseg.php

<?php

namespace Base;

use \StgMunkakepesKoruNepessegQuery as ChildStgMunkakepesKoruNepessegQuery;

use \Exception;

use \PDO;

use Map\StgMunkakepesKoruNepessegTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgMunkakepesKoruNepesseg implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgMunkakepesKoruNepessegTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the nepesseg field.

\* @var string

\*/

protected $nepesseg;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgMunkakepesKoruNepesseg object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgMunkakepesKoruNepesseg</code> instance. If

\* <code>obj</code> is an instance of <code>StgMunkakepesKoruNepesseg</code>, delegates to

\* <code>equals(StgMunkakepesKoruNepesseg)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgMunkakepesKoruNepesseg The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [nepesseg] column value.

\*

\* @return string

\*/

public function getNepesseg()

{

return $this->nepesseg;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgMunkakepesKoruNepessegTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgMunkakepesKoruNepessegTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgMunkakepesKoruNepessegTableMap::translateFieldName('Nepesseg', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nepesseg = (null !== $col) ? (string) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 3; // 3 = StgMunkakepesKoruNepessegTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgMunkakepesKoruNepesseg'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgMunkakepesKoruNepesseg The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgMunkakepesKoruNepessegTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgMunkakepesKoruNepesseg The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgMunkakepesKoruNepessegTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [nepesseg] column.

\*

\* @param string $v new value

\* @return $this|\StgMunkakepesKoruNepesseg The current object (for fluent API support)

\*/

public function setNepesseg($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->nepesseg !== $v) {

$this->nepesseg = $v;

$this->modifiedColumns[StgMunkakepesKoruNepessegTableMap::COL\_NEPESSEG] = true;

}

return $this;

} // setNepesseg()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgMunkakepesKoruNepessegTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgMunkakepesKoruNepessegQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgMunkakepesKoruNepesseg::setDeleted()

\* @see StgMunkakepesKoruNepesseg::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMunkakepesKoruNepessegTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgMunkakepesKoruNepessegQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMunkakepesKoruNepessegTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgMunkakepesKoruNepessegTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgMunkakepesKoruNepessegTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgMunkakepesKoruNepessegTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgMunkakepesKoruNepessegTableMap::COL\_NEPESSEG)) {

$modifiedColumns[':p' . $index++] = 'NEPESSEG';

}

$sql = sprintf(

'INSERT INTO stg\_munkakepes\_koru\_nepesseg (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'NEPESSEG':

$stmt->bindValue($identifier, $this->nepesseg, PDO::PARAM\_STR);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgMunkakepesKoruNepessegTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getNepesseg();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgMunkakepesKoruNepesseg'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgMunkakepesKoruNepesseg'][$this->getPrimaryKey()] = true;

$keys = StgMunkakepesKoruNepessegTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getNepesseg(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgMunkakepesKoruNepesseg

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgMunkakepesKoruNepessegTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgMunkakepesKoruNepesseg

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setNepesseg($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgMunkakepesKoruNepessegTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setNepesseg($arr[$keys[2]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgMunkakepesKoruNepesseg The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgMunkakepesKoruNepessegTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgMunkakepesKoruNepessegTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgMunkakepesKoruNepessegTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgMunkakepesKoruNepessegTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgMunkakepesKoruNepessegTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgMunkakepesKoruNepessegTableMap::COL\_NEPESSEG)) {

$criteria->add(StgMunkakepesKoruNepessegTableMap::COL\_NEPESSEG, $this->nepesseg);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgMunkakepesKoruNepesseg object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgMunkakepesKoruNepesseg (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setNepesseg($this->getNepesseg());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgMunkakepesKoruNepesseg Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->nepesseg = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgMunkakepesKoruNepessegTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgMunkakepesKoruNepessegQuery.php

<?php

namespace Base;

use \StgMunkakepesKoruNepesseg as ChildStgMunkakepesKoruNepesseg;

use \StgMunkakepesKoruNepessegQuery as ChildStgMunkakepesKoruNepessegQuery;

use \Exception;

use Map\StgMunkakepesKoruNepessegTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_munkakepes\_koru\_nepesseg' table.

\*

\*

\*

\* @method ChildStgMunkakepesKoruNepessegQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgMunkakepesKoruNepessegQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgMunkakepesKoruNepessegQuery orderByNepesseg($order = Criteria::ASC) Order by the nepesseg column

\*

\* @method ChildStgMunkakepesKoruNepessegQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgMunkakepesKoruNepessegQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgMunkakepesKoruNepessegQuery groupByNepesseg() Group by the nepesseg column

\*

\* @method ChildStgMunkakepesKoruNepessegQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgMunkakepesKoruNepessegQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgMunkakepesKoruNepessegQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgMunkakepesKoruNepesseg findOne(ConnectionInterface $con = null) Return the first ChildStgMunkakepesKoruNepesseg matching the query

\* @method ChildStgMunkakepesKoruNepesseg findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgMunkakepesKoruNepesseg matching the query, or a new ChildStgMunkakepesKoruNepesseg object populated from the query conditions when no match is found

\*

\* @method ChildStgMunkakepesKoruNepesseg findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgMunkakepesKoruNepesseg filtered by the telepules\_nev column

\* @method ChildStgMunkakepesKoruNepesseg findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgMunkakepesKoruNepesseg filtered by the telepules\_KSHKOD column

\* @method ChildStgMunkakepesKoruNepesseg findOneByNepesseg(string $nepesseg) Return the first ChildStgMunkakepesKoruNepesseg filtered by the nepesseg column

\*

\* @method ChildStgMunkakepesKoruNepesseg[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgMunkakepesKoruNepesseg objects based on current ModelCriteria

\* @method ChildStgMunkakepesKoruNepesseg[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgMunkakepesKoruNepesseg objects filtered by the telepules\_nev column

\* @method ChildStgMunkakepesKoruNepesseg[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgMunkakepesKoruNepesseg objects filtered by the telepules\_KSHKOD column

\* @method ChildStgMunkakepesKoruNepesseg[]|ObjectCollection findByNepesseg(string $nepesseg) Return ChildStgMunkakepesKoruNepesseg objects filtered by the nepesseg column

\* @method ChildStgMunkakepesKoruNepesseg[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgMunkakepesKoruNepessegQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgMunkakepesKoruNepessegQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgMunkakepesKoruNepesseg', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgMunkakepesKoruNepessegQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgMunkakepesKoruNepessegQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgMunkakepesKoruNepessegQuery) {

return $criteria;

}

$query = new ChildStgMunkakepesKoruNepessegQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgMunkakepesKoruNepesseg|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgMunkakepesKoruNepesseg object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgMunkakepesKoruNepesseg object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgMunkakepesKoruNepessegQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgMunkakepesKoruNepesseg object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgMunkakepesKoruNepessegQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgMunkakepesKoruNepesseg object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgMunkakepesKoruNepessegQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgMunkakepesKoruNepessegTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgMunkakepesKoruNepessegQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgMunkakepesKoruNepessegTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the nepesseg column

\*

\* Example usage:

\* <code>

\* $query->filterByNepesseg('fooValue'); // WHERE nepesseg = 'fooValue'

\* $query->filterByNepesseg('%fooValue%'); // WHERE nepesseg LIKE '%fooValue%'

\* </code>

\*

\* @param string $nepesseg The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgMunkakepesKoruNepessegQuery The current query, for fluid interface

\*/

public function filterByNepesseg($nepesseg = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($nepesseg)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $nepesseg)) {

$nepesseg = str\_replace('\*', '%', $nepesseg);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgMunkakepesKoruNepessegTableMap::COL\_NEPESSEG, $nepesseg, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgMunkakepesKoruNepesseg $stgMunkakepesKoruNepesseg Object to remove from the list of results

\*

\* @return $this|ChildStgMunkakepesKoruNepessegQuery The current query, for fluid interface

\*/

public function prune($stgMunkakepesKoruNepesseg = null)

{

if ($stgMunkakepesKoruNepesseg) {

throw new LogicException('StgMunkakepesKoruNepesseg object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_munkakepes\_koru\_nepesseg table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMunkakepesKoruNepessegTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgMunkakepesKoruNepessegTableMap::clearInstancePool();

StgMunkakepesKoruNepessegTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMunkakepesKoruNepessegTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgMunkakepesKoruNepessegTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgMunkakepesKoruNepessegTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgMunkakepesKoruNepessegTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgMunkakepesKoruNepessegQuery

#### StgNemzetkoziTranzitforgalom.php

<?php

namespace Base;

use \StgNemzetkoziTranzitforgalomQuery as ChildStgNemzetkoziTranzitforgalomQuery;

use \Exception;

use \PDO;

use Map\StgNemzetkoziTranzitforgalomTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgNemzetkoziTranzitforgalom implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgNemzetkoziTranzitforgalomTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the hataratlepok\_szama field.

\* @var int

\*/

protected $hataratlepok\_szama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgNemzetkoziTranzitforgalom object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgNemzetkoziTranzitforgalom</code> instance. If

\* <code>obj</code> is an instance of <code>StgNemzetkoziTranzitforgalom</code>, delegates to

\* <code>equals(StgNemzetkoziTranzitforgalom)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgNemzetkoziTranzitforgalom The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [hataratlepok\_szama] column value.

\*

\* @return int

\*/

public function getHataratlepokSzama()

{

return $this->hataratlepok\_szama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgNemzetkoziTranzitforgalomTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgNemzetkoziTranzitforgalomTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgNemzetkoziTranzitforgalomTableMap::translateFieldName('HataratlepokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->hataratlepok\_szama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 3; // 3 = StgNemzetkoziTranzitforgalomTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgNemzetkoziTranzitforgalom'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgNemzetkoziTranzitforgalom The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgNemzetkoziTranzitforgalomTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgNemzetkoziTranzitforgalom The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgNemzetkoziTranzitforgalomTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [hataratlepok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgNemzetkoziTranzitforgalom The current object (for fluent API support)

\*/

public function setHataratlepokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->hataratlepok\_szama !== $v) {

$this->hataratlepok\_szama = $v;

$this->modifiedColumns[StgNemzetkoziTranzitforgalomTableMap::COL\_HATARATLEPOK\_SZAMA] = true;

}

return $this;

} // setHataratlepokSzama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgNemzetkoziTranzitforgalomTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgNemzetkoziTranzitforgalomQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgNemzetkoziTranzitforgalom::setDeleted()

\* @see StgNemzetkoziTranzitforgalom::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNemzetkoziTranzitforgalomTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgNemzetkoziTranzitforgalomQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNemzetkoziTranzitforgalomTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgNemzetkoziTranzitforgalomTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgNemzetkoziTranzitforgalomTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgNemzetkoziTranzitforgalomTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgNemzetkoziTranzitforgalomTableMap::COL\_HATARATLEPOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'HATARATLEPOK\_SZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_nemzetkozi\_tranzitforgalom (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'HATARATLEPOK\_SZAMA':

$stmt->bindValue($identifier, $this->hataratlepok\_szama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgNemzetkoziTranzitforgalomTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getHataratlepokSzama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgNemzetkoziTranzitforgalom'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgNemzetkoziTranzitforgalom'][$this->getPrimaryKey()] = true;

$keys = StgNemzetkoziTranzitforgalomTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getHataratlepokSzama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgNemzetkoziTranzitforgalom

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgNemzetkoziTranzitforgalomTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgNemzetkoziTranzitforgalom

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setHataratlepokSzama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgNemzetkoziTranzitforgalomTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setHataratlepokSzama($arr[$keys[2]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgNemzetkoziTranzitforgalom The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgNemzetkoziTranzitforgalomTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgNemzetkoziTranzitforgalomTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgNemzetkoziTranzitforgalomTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgNemzetkoziTranzitforgalomTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgNemzetkoziTranzitforgalomTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgNemzetkoziTranzitforgalomTableMap::COL\_HATARATLEPOK\_SZAMA)) {

$criteria->add(StgNemzetkoziTranzitforgalomTableMap::COL\_HATARATLEPOK\_SZAMA, $this->hataratlepok\_szama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgNemzetkoziTranzitforgalom object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgNemzetkoziTranzitforgalom (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setHataratlepokSzama($this->getHataratlepokSzama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgNemzetkoziTranzitforgalom Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->hataratlepok\_szama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgNemzetkoziTranzitforgalomTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgNemzetkoziTranzitforgalomQuery.php

<?php

namespace Base;

use \StgNemzetkoziTranzitforgalom as ChildStgNemzetkoziTranzitforgalom;

use \StgNemzetkoziTranzitforgalomQuery as ChildStgNemzetkoziTranzitforgalomQuery;

use \Exception;

use Map\StgNemzetkoziTranzitforgalomTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_nemzetkozi\_tranzitforgalom' table.

\*

\*

\*

\* @method ChildStgNemzetkoziTranzitforgalomQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgNemzetkoziTranzitforgalomQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgNemzetkoziTranzitforgalomQuery orderByHataratlepokSzama($order = Criteria::ASC) Order by the hataratlepok\_szama column

\*

\* @method ChildStgNemzetkoziTranzitforgalomQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgNemzetkoziTranzitforgalomQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgNemzetkoziTranzitforgalomQuery groupByHataratlepokSzama() Group by the hataratlepok\_szama column

\*

\* @method ChildStgNemzetkoziTranzitforgalomQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgNemzetkoziTranzitforgalomQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgNemzetkoziTranzitforgalomQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgNemzetkoziTranzitforgalom findOne(ConnectionInterface $con = null) Return the first ChildStgNemzetkoziTranzitforgalom matching the query

\* @method ChildStgNemzetkoziTranzitforgalom findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgNemzetkoziTranzitforgalom matching the query, or a new ChildStgNemzetkoziTranzitforgalom object populated from the query conditions when no match is found

\*

\* @method ChildStgNemzetkoziTranzitforgalom findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgNemzetkoziTranzitforgalom filtered by the telepules\_nev column

\* @method ChildStgNemzetkoziTranzitforgalom findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgNemzetkoziTranzitforgalom filtered by the telepules\_KSHKOD column

\* @method ChildStgNemzetkoziTranzitforgalom findOneByHataratlepokSzama(int $hataratlepok\_szama) Return the first ChildStgNemzetkoziTranzitforgalom filtered by the hataratlepok\_szama column

\*

\* @method ChildStgNemzetkoziTranzitforgalom[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgNemzetkoziTranzitforgalom objects based on current ModelCriteria

\* @method ChildStgNemzetkoziTranzitforgalom[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgNemzetkoziTranzitforgalom objects filtered by the telepules\_nev column

\* @method ChildStgNemzetkoziTranzitforgalom[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgNemzetkoziTranzitforgalom objects filtered by the telepules\_KSHKOD column

\* @method ChildStgNemzetkoziTranzitforgalom[]|ObjectCollection findByHataratlepokSzama(int $hataratlepok\_szama) Return ChildStgNemzetkoziTranzitforgalom objects filtered by the hataratlepok\_szama column

\* @method ChildStgNemzetkoziTranzitforgalom[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgNemzetkoziTranzitforgalomQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgNemzetkoziTranzitforgalomQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgNemzetkoziTranzitforgalom', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgNemzetkoziTranzitforgalomQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgNemzetkoziTranzitforgalomQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgNemzetkoziTranzitforgalomQuery) {

return $criteria;

}

$query = new ChildStgNemzetkoziTranzitforgalomQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgNemzetkoziTranzitforgalom|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgNemzetkoziTranzitforgalom object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgNemzetkoziTranzitforgalom object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgNemzetkoziTranzitforgalomQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgNemzetkoziTranzitforgalom object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgNemzetkoziTranzitforgalomQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgNemzetkoziTranzitforgalom object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNemzetkoziTranzitforgalomQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgNemzetkoziTranzitforgalomTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNemzetkoziTranzitforgalomQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgNemzetkoziTranzitforgalomTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the hataratlepok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByHataratlepokSzama(1234); // WHERE hataratlepok\_szama = 1234

\* $query->filterByHataratlepokSzama(array(12, 34)); // WHERE hataratlepok\_szama IN (12, 34)

\* $query->filterByHataratlepokSzama(array('min' => 12)); // WHERE hataratlepok\_szama > 12

\* </code>

\*

\* @param mixed $hataratlepokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNemzetkoziTranzitforgalomQuery The current query, for fluid interface

\*/

public function filterByHataratlepokSzama($hataratlepokSzama = null, $comparison = null)

{

if (is\_array($hataratlepokSzama)) {

$useMinMax = false;

if (isset($hataratlepokSzama['min'])) {

$this->addUsingAlias(StgNemzetkoziTranzitforgalomTableMap::COL\_HATARATLEPOK\_SZAMA, $hataratlepokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($hataratlepokSzama['max'])) {

$this->addUsingAlias(StgNemzetkoziTranzitforgalomTableMap::COL\_HATARATLEPOK\_SZAMA, $hataratlepokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNemzetkoziTranzitforgalomTableMap::COL\_HATARATLEPOK\_SZAMA, $hataratlepokSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgNemzetkoziTranzitforgalom $stgNemzetkoziTranzitforgalom Object to remove from the list of results

\*

\* @return $this|ChildStgNemzetkoziTranzitforgalomQuery The current query, for fluid interface

\*/

public function prune($stgNemzetkoziTranzitforgalom = null)

{

if ($stgNemzetkoziTranzitforgalom) {

throw new LogicException('StgNemzetkoziTranzitforgalom object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_nemzetkozi\_tranzitforgalom table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNemzetkoziTranzitforgalomTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgNemzetkoziTranzitforgalomTableMap::clearInstancePool();

StgNemzetkoziTranzitforgalomTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNemzetkoziTranzitforgalomTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgNemzetkoziTranzitforgalomTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgNemzetkoziTranzitforgalomTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgNemzetkoziTranzitforgalomTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgNemzetkoziTranzitforgalomQuery

#### StgNodeAttribs.php

<?php

namespace Base;

use \StgNodeAttribsQuery as ChildStgNodeAttribsQuery;

use \Exception;

use \PDO;

use Map\StgNodeAttribsTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgNodeAttribs implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgNodeAttribsTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the shape\_id field.

\* @var int

\*/

protected $shape\_id;

/\*\*

\* The value for the kszam field.

\* @var string

\*/

protected $kszam;

/\*\*

\* The value for the pkod field.

\* @var int

\*/

protected $pkod;

/\*\*

\* The value for the kkod field.

\* @var string

\*/

protected $kkod;

/\*\*

\* The value for the vvkod field.

\* @var string

\*/

protected $vvkod;

/\*\*

\* The value for the kszelv field.

\* @var string

\*/

protected $kszelv;

/\*\*

\* The value for the vszelv field.

\* @var string

\*/

protected $vszelv;

/\*\*

\* The value for the rshossz field.

\* @var double

\*/

protected $rshossz;

/\*\*

\* The value for the anf field.

\* @var int

\*/

protected $anf;

/\*\*

\* The value for the anet field.

\* @var int

\*/

protected $anet;

/\*\*

\* The value for the mof field.

\* @var int

\*/

protected $mof;

/\*\*

\* The value for the ongj field.

\* @var int

\*/

protected $ongj;

/\*\*

\* The value for the oj field.

\* @var int

\*/

protected $oj;

/\*\*

\* The value for the omot field.

\* @var int

\*/

protected $omot;

/\*\*

\* The value for the ev field.

\* @var int

\*/

protected $ev;

/\*\*

\* The value for the asz field.

\* @var int

\*/

protected $asz;

/\*\*

\* The value for the buszcs field.

\* @var int

\*/

protected $buszcs;

/\*\*

\* The value for the busze field.

\* @var int

\*/

protected $busze;

/\*\*

\* The value for the obusz field.

\* @var int

\*/

protected $obusz;

/\*\*

\* The value for the nyszer field.

\* @var int

\*/

protected $nyszer;

/\*\*

\* The value for the potktgk field.

\* @var int

\*/

protected $potktgk;

/\*\*

\* The value for the ktgk field.

\* @var int

\*/

protected $ktgk;

/\*\*

\* The value for the ntgk field.

\* @var int

\*/

protected $ntgk;

/\*\*

\* The value for the kntgk field.

\* @var int

\*/

protected $kntgk;

/\*\*

\* The value for the otgk field.

\* @var int

\*/

protected $otgk;

/\*\*

\* The value for the szgk field.

\* @var int

\*/

protected $szgk;

/\*\*

\* The value for the oszgk field.

\* @var int

\*/

protected $oszgk;

/\*\*

\* The value for the mkp field.

\* @var int

\*/

protected $mkp;

/\*\*

\* The value for the kpf field.

\* @var int

\*/

protected $kpf;

/\*\*

\* The value for the lassu field.

\* @var int

\*/

protected $lassu;

/\*\*

\* The value for the spec field.

\* @var int

\*/

protected $spec;

/\*\*

\* The value for the fmegb field.

\* @var string

\*/

protected $fmegb;

/\*\*

\* The value for the adatforr field.

\* @var string

\*/

protected $adatforr;

/\*\*

\* The value for the szamlnap field.

\* @var int

\*/

protected $szamlnap;

/\*\*

\* The value for the jelleg1 field.

\* @var string

\*/

protected $jelleg1;

/\*\*

\* The value for the jelleg2 field.

\* @var string

\*/

protected $jelleg2;

/\*\*

\* The value for the fmegj field.

\* @var string

\*/

protected $fmegj;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgNodeAttribs object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgNodeAttribs</code> instance. If

\* <code>obj</code> is an instance of <code>StgNodeAttribs</code>, delegates to

\* <code>equals(StgNodeAttribs)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgNodeAttribs The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [shape\_id] column value.

\*

\* @return int

\*/

public function getShapeId()

{

return $this->shape\_id;

}

/\*\*

\* Get the [kszam] column value.

\*

\* @return string

\*/

public function getKszam()

{

return $this->kszam;

}

/\*\*

\* Get the [pkod] column value.

\*

\* @return int

\*/

public function getPkod()

{

return $this->pkod;

}

/\*\*

\* Get the [kkod] column value.

\*

\* @return string

\*/

public function getKkod()

{

return $this->kkod;

}

/\*\*

\* Get the [vvkod] column value.

\*

\* @return string

\*/

public function getVvkod()

{

return $this->vvkod;

}

/\*\*

\* Get the [kszelv] column value.

\*

\* @return string

\*/

public function getKszelv()

{

return $this->kszelv;

}

/\*\*

\* Get the [vszelv] column value.

\*

\* @return string

\*/

public function getVszelv()

{

return $this->vszelv;

}

/\*\*

\* Get the [rshossz] column value.

\*

\* @return double

\*/

public function getRshossz()

{

return $this->rshossz;

}

/\*\*

\* Get the [anf] column value.

\*

\* @return int

\*/

public function getAnf()

{

return $this->anf;

}

/\*\*

\* Get the [anet] column value.

\*

\* @return int

\*/

public function getAnet()

{

return $this->anet;

}

/\*\*

\* Get the [mof] column value.

\*

\* @return int

\*/

public function getMof()

{

return $this->mof;

}

/\*\*

\* Get the [ongj] column value.

\*

\* @return int

\*/

public function getOngj()

{

return $this->ongj;

}

/\*\*

\* Get the [oj] column value.

\*

\* @return int

\*/

public function getOj()

{

return $this->oj;

}

/\*\*

\* Get the [omot] column value.

\*

\* @return int

\*/

public function getOmot()

{

return $this->omot;

}

/\*\*

\* Get the [ev] column value.

\*

\* @return int

\*/

public function getEv()

{

return $this->ev;

}

/\*\*

\* Get the [asz] column value.

\*

\* @return int

\*/

public function getAsz()

{

return $this->asz;

}

/\*\*

\* Get the [buszcs] column value.

\*

\* @return int

\*/

public function getBuszcs()

{

return $this->buszcs;

}

/\*\*

\* Get the [busze] column value.

\*

\* @return int

\*/

public function getBusze()

{

return $this->busze;

}

/\*\*

\* Get the [obusz] column value.

\*

\* @return int

\*/

public function getObusz()

{

return $this->obusz;

}

/\*\*

\* Get the [nyszer] column value.

\*

\* @return int

\*/

public function getNyszer()

{

return $this->nyszer;

}

/\*\*

\* Get the [potktgk] column value.

\*

\* @return int

\*/

public function getPotktgk()

{

return $this->potktgk;

}

/\*\*

\* Get the [ktgk] column value.

\*

\* @return int

\*/

public function getKtgk()

{

return $this->ktgk;

}

/\*\*

\* Get the [ntgk] column value.

\*

\* @return int

\*/

public function getNtgk()

{

return $this->ntgk;

}

/\*\*

\* Get the [kntgk] column value.

\*

\* @return int

\*/

public function getKntgk()

{

return $this->kntgk;

}

/\*\*

\* Get the [otgk] column value.

\*

\* @return int

\*/

public function getOtgk()

{

return $this->otgk;

}

/\*\*

\* Get the [szgk] column value.

\*

\* @return int

\*/

public function getSzgk()

{

return $this->szgk;

}

/\*\*

\* Get the [oszgk] column value.

\*

\* @return int

\*/

public function getOszgk()

{

return $this->oszgk;

}

/\*\*

\* Get the [mkp] column value.

\*

\* @return int

\*/

public function getMkp()

{

return $this->mkp;

}

/\*\*

\* Get the [kpf] column value.

\*

\* @return int

\*/

public function getKpf()

{

return $this->kpf;

}

/\*\*

\* Get the [lassu] column value.

\*

\* @return int

\*/

public function getLassu()

{

return $this->lassu;

}

/\*\*

\* Get the [spec] column value.

\*

\* @return int

\*/

public function getSpec()

{

return $this->spec;

}

/\*\*

\* Get the [fmegb] column value.

\*

\* @return string

\*/

public function getFmegb()

{

return $this->fmegb;

}

/\*\*

\* Get the [adatforr] column value.

\*

\* @return string

\*/

public function getAdatforr()

{

return $this->adatforr;

}

/\*\*

\* Get the [szamlnap] column value.

\*

\* @return int

\*/

public function getSzamlnap()

{

return $this->szamlnap;

}

/\*\*

\* Get the [jelleg1] column value.

\*

\* @return string

\*/

public function getJelleg1()

{

return $this->jelleg1;

}

/\*\*

\* Get the [jelleg2] column value.

\*

\* @return string

\*/

public function getJelleg2()

{

return $this->jelleg2;

}

/\*\*

\* Get the [fmegj] column value.

\*

\* @return string

\*/

public function getFmegj()

{

return $this->fmegj;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgNodeAttribsTableMap::translateFieldName('ShapeId', TableMap::TYPE\_PHPNAME, $indexType)];

$this->shape\_id = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgNodeAttribsTableMap::translateFieldName('Kszam', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kszam = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgNodeAttribsTableMap::translateFieldName('Pkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->pkod = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgNodeAttribsTableMap::translateFieldName('Kkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgNodeAttribsTableMap::translateFieldName('Vvkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->vvkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 5 + $startcol : StgNodeAttribsTableMap::translateFieldName('Kszelv', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kszelv = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 6 + $startcol : StgNodeAttribsTableMap::translateFieldName('Vszelv', TableMap::TYPE\_PHPNAME, $indexType)];

$this->vszelv = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 7 + $startcol : StgNodeAttribsTableMap::translateFieldName('Rshossz', TableMap::TYPE\_PHPNAME, $indexType)];

$this->rshossz = (null !== $col) ? (double) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 8 + $startcol : StgNodeAttribsTableMap::translateFieldName('Anf', TableMap::TYPE\_PHPNAME, $indexType)];

$this->anf = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 9 + $startcol : StgNodeAttribsTableMap::translateFieldName('Anet', TableMap::TYPE\_PHPNAME, $indexType)];

$this->anet = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 10 + $startcol : StgNodeAttribsTableMap::translateFieldName('Mof', TableMap::TYPE\_PHPNAME, $indexType)];

$this->mof = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 11 + $startcol : StgNodeAttribsTableMap::translateFieldName('Ongj', TableMap::TYPE\_PHPNAME, $indexType)];

$this->ongj = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 12 + $startcol : StgNodeAttribsTableMap::translateFieldName('Oj', TableMap::TYPE\_PHPNAME, $indexType)];

$this->oj = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 13 + $startcol : StgNodeAttribsTableMap::translateFieldName('Omot', TableMap::TYPE\_PHPNAME, $indexType)];

$this->omot = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 14 + $startcol : StgNodeAttribsTableMap::translateFieldName('Ev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->ev = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 15 + $startcol : StgNodeAttribsTableMap::translateFieldName('Asz', TableMap::TYPE\_PHPNAME, $indexType)];

$this->asz = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 16 + $startcol : StgNodeAttribsTableMap::translateFieldName('Buszcs', TableMap::TYPE\_PHPNAME, $indexType)];

$this->buszcs = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 17 + $startcol : StgNodeAttribsTableMap::translateFieldName('Busze', TableMap::TYPE\_PHPNAME, $indexType)];

$this->busze = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 18 + $startcol : StgNodeAttribsTableMap::translateFieldName('Obusz', TableMap::TYPE\_PHPNAME, $indexType)];

$this->obusz = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 19 + $startcol : StgNodeAttribsTableMap::translateFieldName('Nyszer', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nyszer = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 20 + $startcol : StgNodeAttribsTableMap::translateFieldName('Potktgk', TableMap::TYPE\_PHPNAME, $indexType)];

$this->potktgk = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 21 + $startcol : StgNodeAttribsTableMap::translateFieldName('Ktgk', TableMap::TYPE\_PHPNAME, $indexType)];

$this->ktgk = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 22 + $startcol : StgNodeAttribsTableMap::translateFieldName('Ntgk', TableMap::TYPE\_PHPNAME, $indexType)];

$this->ntgk = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 23 + $startcol : StgNodeAttribsTableMap::translateFieldName('Kntgk', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kntgk = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 24 + $startcol : StgNodeAttribsTableMap::translateFieldName('Otgk', TableMap::TYPE\_PHPNAME, $indexType)];

$this->otgk = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 25 + $startcol : StgNodeAttribsTableMap::translateFieldName('Szgk', TableMap::TYPE\_PHPNAME, $indexType)];

$this->szgk = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 26 + $startcol : StgNodeAttribsTableMap::translateFieldName('Oszgk', TableMap::TYPE\_PHPNAME, $indexType)];

$this->oszgk = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 27 + $startcol : StgNodeAttribsTableMap::translateFieldName('Mkp', TableMap::TYPE\_PHPNAME, $indexType)];

$this->mkp = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 28 + $startcol : StgNodeAttribsTableMap::translateFieldName('Kpf', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kpf = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 29 + $startcol : StgNodeAttribsTableMap::translateFieldName('Lassu', TableMap::TYPE\_PHPNAME, $indexType)];

$this->lassu = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 30 + $startcol : StgNodeAttribsTableMap::translateFieldName('Spec', TableMap::TYPE\_PHPNAME, $indexType)];

$this->spec = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 31 + $startcol : StgNodeAttribsTableMap::translateFieldName('Fmegb', TableMap::TYPE\_PHPNAME, $indexType)];

$this->fmegb = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 32 + $startcol : StgNodeAttribsTableMap::translateFieldName('Adatforr', TableMap::TYPE\_PHPNAME, $indexType)];

$this->adatforr = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 33 + $startcol : StgNodeAttribsTableMap::translateFieldName('Szamlnap', TableMap::TYPE\_PHPNAME, $indexType)];

$this->szamlnap = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 34 + $startcol : StgNodeAttribsTableMap::translateFieldName('Jelleg1', TableMap::TYPE\_PHPNAME, $indexType)];

$this->jelleg1 = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 35 + $startcol : StgNodeAttribsTableMap::translateFieldName('Jelleg2', TableMap::TYPE\_PHPNAME, $indexType)];

$this->jelleg2 = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 36 + $startcol : StgNodeAttribsTableMap::translateFieldName('Fmegj', TableMap::TYPE\_PHPNAME, $indexType)];

$this->fmegj = (null !== $col) ? (string) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 37; // 37 = StgNodeAttribsTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgNodeAttribs'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [shape\_id] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setShapeId($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->shape\_id !== $v) {

$this->shape\_id = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_SHAPE\_ID] = true;

}

return $this;

} // setShapeId()

/\*\*

\* Set the value of [kszam] column.

\*

\* @param string $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setKszam($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->kszam !== $v) {

$this->kszam = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_KSZAM] = true;

}

return $this;

} // setKszam()

/\*\*

\* Set the value of [pkod] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setPkod($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->pkod !== $v) {

$this->pkod = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_PKOD] = true;

}

return $this;

} // setPkod()

/\*\*

\* Set the value of [kkod] column.

\*

\* @param string $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setKkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->kkod !== $v) {

$this->kkod = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_KKOD] = true;

}

return $this;

} // setKkod()

/\*\*

\* Set the value of [vvkod] column.

\*

\* @param string $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setVvkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->vvkod !== $v) {

$this->vvkod = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_VVKOD] = true;

}

return $this;

} // setVvkod()

/\*\*

\* Set the value of [kszelv] column.

\*

\* @param string $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setKszelv($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->kszelv !== $v) {

$this->kszelv = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_KSZELV] = true;

}

return $this;

} // setKszelv()

/\*\*

\* Set the value of [vszelv] column.

\*

\* @param string $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setVszelv($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->vszelv !== $v) {

$this->vszelv = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_VSZELV] = true;

}

return $this;

} // setVszelv()

/\*\*

\* Set the value of [rshossz] column.

\*

\* @param double $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setRshossz($v)

{

if ($v !== null) {

$v = (double) $v;

}

if ($this->rshossz !== $v) {

$this->rshossz = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_RSHOSSZ] = true;

}

return $this;

} // setRshossz()

/\*\*

\* Set the value of [anf] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setAnf($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->anf !== $v) {

$this->anf = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_ANF] = true;

}

return $this;

} // setAnf()

/\*\*

\* Set the value of [anet] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setAnet($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->anet !== $v) {

$this->anet = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_ANET] = true;

}

return $this;

} // setAnet()

/\*\*

\* Set the value of [mof] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setMof($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->mof !== $v) {

$this->mof = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_MOF] = true;

}

return $this;

} // setMof()

/\*\*

\* Set the value of [ongj] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setOngj($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->ongj !== $v) {

$this->ongj = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_ONGJ] = true;

}

return $this;

} // setOngj()

/\*\*

\* Set the value of [oj] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setOj($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->oj !== $v) {

$this->oj = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_OJ] = true;

}

return $this;

} // setOj()

/\*\*

\* Set the value of [omot] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setOmot($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->omot !== $v) {

$this->omot = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_OMOT] = true;

}

return $this;

} // setOmot()

/\*\*

\* Set the value of [ev] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setEv($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->ev !== $v) {

$this->ev = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_EV] = true;

}

return $this;

} // setEv()

/\*\*

\* Set the value of [asz] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setAsz($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->asz !== $v) {

$this->asz = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_ASZ] = true;

}

return $this;

} // setAsz()

/\*\*

\* Set the value of [buszcs] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setBuszcs($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->buszcs !== $v) {

$this->buszcs = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_BUSZCS] = true;

}

return $this;

} // setBuszcs()

/\*\*

\* Set the value of [busze] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setBusze($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->busze !== $v) {

$this->busze = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_BUSZE] = true;

}

return $this;

} // setBusze()

/\*\*

\* Set the value of [obusz] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setObusz($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->obusz !== $v) {

$this->obusz = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_OBUSZ] = true;

}

return $this;

} // setObusz()

/\*\*

\* Set the value of [nyszer] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setNyszer($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nyszer !== $v) {

$this->nyszer = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_NYSZER] = true;

}

return $this;

} // setNyszer()

/\*\*

\* Set the value of [potktgk] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setPotktgk($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->potktgk !== $v) {

$this->potktgk = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_POTKTGK] = true;

}

return $this;

} // setPotktgk()

/\*\*

\* Set the value of [ktgk] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setKtgk($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->ktgk !== $v) {

$this->ktgk = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_KTGK] = true;

}

return $this;

} // setKtgk()

/\*\*

\* Set the value of [ntgk] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setNtgk($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->ntgk !== $v) {

$this->ntgk = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_NTGK] = true;

}

return $this;

} // setNtgk()

/\*\*

\* Set the value of [kntgk] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setKntgk($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->kntgk !== $v) {

$this->kntgk = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_KNTGK] = true;

}

return $this;

} // setKntgk()

/\*\*

\* Set the value of [otgk] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setOtgk($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->otgk !== $v) {

$this->otgk = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_OTGK] = true;

}

return $this;

} // setOtgk()

/\*\*

\* Set the value of [szgk] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setSzgk($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->szgk !== $v) {

$this->szgk = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_SZGK] = true;

}

return $this;

} // setSzgk()

/\*\*

\* Set the value of [oszgk] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setOszgk($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->oszgk !== $v) {

$this->oszgk = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_OSZGK] = true;

}

return $this;

} // setOszgk()

/\*\*

\* Set the value of [mkp] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setMkp($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->mkp !== $v) {

$this->mkp = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_MKP] = true;

}

return $this;

} // setMkp()

/\*\*

\* Set the value of [kpf] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setKpf($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->kpf !== $v) {

$this->kpf = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_KPF] = true;

}

return $this;

} // setKpf()

/\*\*

\* Set the value of [lassu] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setLassu($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->lassu !== $v) {

$this->lassu = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_LASSU] = true;

}

return $this;

} // setLassu()

/\*\*

\* Set the value of [spec] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setSpec($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->spec !== $v) {

$this->spec = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_SPEC] = true;

}

return $this;

} // setSpec()

/\*\*

\* Set the value of [fmegb] column.

\*

\* @param string $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setFmegb($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->fmegb !== $v) {

$this->fmegb = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_FMEGB] = true;

}

return $this;

} // setFmegb()

/\*\*

\* Set the value of [adatforr] column.

\*

\* @param string $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setAdatforr($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->adatforr !== $v) {

$this->adatforr = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_ADATFORR] = true;

}

return $this;

} // setAdatforr()

/\*\*

\* Set the value of [szamlnap] column.

\*

\* @param int $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setSzamlnap($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->szamlnap !== $v) {

$this->szamlnap = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_SZAMLNAP] = true;

}

return $this;

} // setSzamlnap()

/\*\*

\* Set the value of [jelleg1] column.

\*

\* @param string $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setJelleg1($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->jelleg1 !== $v) {

$this->jelleg1 = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_JELLEG1] = true;

}

return $this;

} // setJelleg1()

/\*\*

\* Set the value of [jelleg2] column.

\*

\* @param string $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setJelleg2($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->jelleg2 !== $v) {

$this->jelleg2 = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_JELLEG2] = true;

}

return $this;

} // setJelleg2()

/\*\*

\* Set the value of [fmegj] column.

\*

\* @param string $v new value

\* @return $this|\StgNodeAttribs The current object (for fluent API support)

\*/

public function setFmegj($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->fmegj !== $v) {

$this->fmegj = $v;

$this->modifiedColumns[StgNodeAttribsTableMap::COL\_FMEGJ] = true;

}

return $this;

} // setFmegj()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgNodeAttribsTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgNodeAttribsQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgNodeAttribs::setDeleted()

\* @see StgNodeAttribs::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNodeAttribsTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgNodeAttribsQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNodeAttribsTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgNodeAttribsTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_SHAPE\_ID)) {

$modifiedColumns[':p' . $index++] = 'SHAPE\_ID';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_KSZAM)) {

$modifiedColumns[':p' . $index++] = 'KSZAM';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_PKOD)) {

$modifiedColumns[':p' . $index++] = 'PKOD';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_KKOD)) {

$modifiedColumns[':p' . $index++] = 'KKOD';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_VVKOD)) {

$modifiedColumns[':p' . $index++] = 'VVKOD';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_KSZELV)) {

$modifiedColumns[':p' . $index++] = 'KSZELV';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_VSZELV)) {

$modifiedColumns[':p' . $index++] = 'VSZELV';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_RSHOSSZ)) {

$modifiedColumns[':p' . $index++] = 'RSHOSSZ';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_ANF)) {

$modifiedColumns[':p' . $index++] = 'ANF';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_ANET)) {

$modifiedColumns[':p' . $index++] = 'ANET';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_MOF)) {

$modifiedColumns[':p' . $index++] = 'MOF';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_ONGJ)) {

$modifiedColumns[':p' . $index++] = 'ONGJ';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_OJ)) {

$modifiedColumns[':p' . $index++] = 'OJ';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_OMOT)) {

$modifiedColumns[':p' . $index++] = 'OMOT';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_EV)) {

$modifiedColumns[':p' . $index++] = 'EV';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_ASZ)) {

$modifiedColumns[':p' . $index++] = 'ASZ';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_BUSZCS)) {

$modifiedColumns[':p' . $index++] = 'BUSZCS';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_BUSZE)) {

$modifiedColumns[':p' . $index++] = 'BUSZE';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_OBUSZ)) {

$modifiedColumns[':p' . $index++] = 'OBUSZ';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_NYSZER)) {

$modifiedColumns[':p' . $index++] = 'NYSZER';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_POTKTGK)) {

$modifiedColumns[':p' . $index++] = 'POTKTGK';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_KTGK)) {

$modifiedColumns[':p' . $index++] = 'KTGK';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_NTGK)) {

$modifiedColumns[':p' . $index++] = 'NTGK';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_KNTGK)) {

$modifiedColumns[':p' . $index++] = 'KNTGK';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_OTGK)) {

$modifiedColumns[':p' . $index++] = 'OTGK';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_SZGK)) {

$modifiedColumns[':p' . $index++] = 'SZGK';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_OSZGK)) {

$modifiedColumns[':p' . $index++] = 'OSZGK';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_MKP)) {

$modifiedColumns[':p' . $index++] = 'MKP';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_KPF)) {

$modifiedColumns[':p' . $index++] = 'KPF';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_LASSU)) {

$modifiedColumns[':p' . $index++] = 'LASSU';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_SPEC)) {

$modifiedColumns[':p' . $index++] = 'SPEC';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_FMEGB)) {

$modifiedColumns[':p' . $index++] = 'FMEGB';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_ADATFORR)) {

$modifiedColumns[':p' . $index++] = 'ADATFORR';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_SZAMLNAP)) {

$modifiedColumns[':p' . $index++] = 'SZAMLNAP';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_JELLEG1)) {

$modifiedColumns[':p' . $index++] = 'JELLEG1';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_JELLEG2)) {

$modifiedColumns[':p' . $index++] = 'JELLEG2';

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_FMEGJ)) {

$modifiedColumns[':p' . $index++] = 'FMEGJ';

}

$sql = sprintf(

'INSERT INTO stg\_node\_attribs (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'SHAPE\_ID':

$stmt->bindValue($identifier, $this->shape\_id, PDO::PARAM\_INT);

break;

case 'KSZAM':

$stmt->bindValue($identifier, $this->kszam, PDO::PARAM\_STR);

break;

case 'PKOD':

$stmt->bindValue($identifier, $this->pkod, PDO::PARAM\_INT);

break;

case 'KKOD':

$stmt->bindValue($identifier, $this->kkod, PDO::PARAM\_STR);

break;

case 'VVKOD':

$stmt->bindValue($identifier, $this->vvkod, PDO::PARAM\_STR);

break;

case 'KSZELV':

$stmt->bindValue($identifier, $this->kszelv, PDO::PARAM\_STR);

break;

case 'VSZELV':

$stmt->bindValue($identifier, $this->vszelv, PDO::PARAM\_STR);

break;

case 'RSHOSSZ':

$stmt->bindValue($identifier, $this->rshossz, PDO::PARAM\_STR);

break;

case 'ANF':

$stmt->bindValue($identifier, $this->anf, PDO::PARAM\_INT);

break;

case 'ANET':

$stmt->bindValue($identifier, $this->anet, PDO::PARAM\_INT);

break;

case 'MOF':

$stmt->bindValue($identifier, $this->mof, PDO::PARAM\_INT);

break;

case 'ONGJ':

$stmt->bindValue($identifier, $this->ongj, PDO::PARAM\_INT);

break;

case 'OJ':

$stmt->bindValue($identifier, $this->oj, PDO::PARAM\_INT);

break;

case 'OMOT':

$stmt->bindValue($identifier, $this->omot, PDO::PARAM\_INT);

break;

case 'EV':

$stmt->bindValue($identifier, $this->ev, PDO::PARAM\_INT);

break;

case 'ASZ':

$stmt->bindValue($identifier, $this->asz, PDO::PARAM\_INT);

break;

case 'BUSZCS':

$stmt->bindValue($identifier, $this->buszcs, PDO::PARAM\_INT);

break;

case 'BUSZE':

$stmt->bindValue($identifier, $this->busze, PDO::PARAM\_INT);

break;

case 'OBUSZ':

$stmt->bindValue($identifier, $this->obusz, PDO::PARAM\_INT);

break;

case 'NYSZER':

$stmt->bindValue($identifier, $this->nyszer, PDO::PARAM\_INT);

break;

case 'POTKTGK':

$stmt->bindValue($identifier, $this->potktgk, PDO::PARAM\_INT);

break;

case 'KTGK':

$stmt->bindValue($identifier, $this->ktgk, PDO::PARAM\_INT);

break;

case 'NTGK':

$stmt->bindValue($identifier, $this->ntgk, PDO::PARAM\_INT);

break;

case 'KNTGK':

$stmt->bindValue($identifier, $this->kntgk, PDO::PARAM\_INT);

break;

case 'OTGK':

$stmt->bindValue($identifier, $this->otgk, PDO::PARAM\_INT);

break;

case 'SZGK':

$stmt->bindValue($identifier, $this->szgk, PDO::PARAM\_INT);

break;

case 'OSZGK':

$stmt->bindValue($identifier, $this->oszgk, PDO::PARAM\_INT);

break;

case 'MKP':

$stmt->bindValue($identifier, $this->mkp, PDO::PARAM\_INT);

break;

case 'KPF':

$stmt->bindValue($identifier, $this->kpf, PDO::PARAM\_INT);

break;

case 'LASSU':

$stmt->bindValue($identifier, $this->lassu, PDO::PARAM\_INT);

break;

case 'SPEC':

$stmt->bindValue($identifier, $this->spec, PDO::PARAM\_INT);

break;

case 'FMEGB':

$stmt->bindValue($identifier, $this->fmegb, PDO::PARAM\_STR);

break;

case 'ADATFORR':

$stmt->bindValue($identifier, $this->adatforr, PDO::PARAM\_STR);

break;

case 'SZAMLNAP':

$stmt->bindValue($identifier, $this->szamlnap, PDO::PARAM\_INT);

break;

case 'JELLEG1':

$stmt->bindValue($identifier, $this->jelleg1, PDO::PARAM\_STR);

break;

case 'JELLEG2':

$stmt->bindValue($identifier, $this->jelleg2, PDO::PARAM\_STR);

break;

case 'FMEGJ':

$stmt->bindValue($identifier, $this->fmegj, PDO::PARAM\_STR);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgNodeAttribsTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getShapeId();

break;

case 1:

return $this->getKszam();

break;

case 2:

return $this->getPkod();

break;

case 3:

return $this->getKkod();

break;

case 4:

return $this->getVvkod();

break;

case 5:

return $this->getKszelv();

break;

case 6:

return $this->getVszelv();

break;

case 7:

return $this->getRshossz();

break;

case 8:

return $this->getAnf();

break;

case 9:

return $this->getAnet();

break;

case 10:

return $this->getMof();

break;

case 11:

return $this->getOngj();

break;

case 12:

return $this->getOj();

break;

case 13:

return $this->getOmot();

break;

case 14:

return $this->getEv();

break;

case 15:

return $this->getAsz();

break;

case 16:

return $this->getBuszcs();

break;

case 17:

return $this->getBusze();

break;

case 18:

return $this->getObusz();

break;

case 19:

return $this->getNyszer();

break;

case 20:

return $this->getPotktgk();

break;

case 21:

return $this->getKtgk();

break;

case 22:

return $this->getNtgk();

break;

case 23:

return $this->getKntgk();

break;

case 24:

return $this->getOtgk();

break;

case 25:

return $this->getSzgk();

break;

case 26:

return $this->getOszgk();

break;

case 27:

return $this->getMkp();

break;

case 28:

return $this->getKpf();

break;

case 29:

return $this->getLassu();

break;

case 30:

return $this->getSpec();

break;

case 31:

return $this->getFmegb();

break;

case 32:

return $this->getAdatforr();

break;

case 33:

return $this->getSzamlnap();

break;

case 34:

return $this->getJelleg1();

break;

case 35:

return $this->getJelleg2();

break;

case 36:

return $this->getFmegj();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgNodeAttribs'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgNodeAttribs'][$this->getPrimaryKey()] = true;

$keys = StgNodeAttribsTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getShapeId(),

$keys[1] => $this->getKszam(),

$keys[2] => $this->getPkod(),

$keys[3] => $this->getKkod(),

$keys[4] => $this->getVvkod(),

$keys[5] => $this->getKszelv(),

$keys[6] => $this->getVszelv(),

$keys[7] => $this->getRshossz(),

$keys[8] => $this->getAnf(),

$keys[9] => $this->getAnet(),

$keys[10] => $this->getMof(),

$keys[11] => $this->getOngj(),

$keys[12] => $this->getOj(),

$keys[13] => $this->getOmot(),

$keys[14] => $this->getEv(),

$keys[15] => $this->getAsz(),

$keys[16] => $this->getBuszcs(),

$keys[17] => $this->getBusze(),

$keys[18] => $this->getObusz(),

$keys[19] => $this->getNyszer(),

$keys[20] => $this->getPotktgk(),

$keys[21] => $this->getKtgk(),

$keys[22] => $this->getNtgk(),

$keys[23] => $this->getKntgk(),

$keys[24] => $this->getOtgk(),

$keys[25] => $this->getSzgk(),

$keys[26] => $this->getOszgk(),

$keys[27] => $this->getMkp(),

$keys[28] => $this->getKpf(),

$keys[29] => $this->getLassu(),

$keys[30] => $this->getSpec(),

$keys[31] => $this->getFmegb(),

$keys[32] => $this->getAdatforr(),

$keys[33] => $this->getSzamlnap(),

$keys[34] => $this->getJelleg1(),

$keys[35] => $this->getJelleg2(),

$keys[36] => $this->getFmegj(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgNodeAttribs

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgNodeAttribsTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgNodeAttribs

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setShapeId($value);

break;

case 1:

$this->setKszam($value);

break;

case 2:

$this->setPkod($value);

break;

case 3:

$this->setKkod($value);

break;

case 4:

$this->setVvkod($value);

break;

case 5:

$this->setKszelv($value);

break;

case 6:

$this->setVszelv($value);

break;

case 7:

$this->setRshossz($value);

break;

case 8:

$this->setAnf($value);

break;

case 9:

$this->setAnet($value);

break;

case 10:

$this->setMof($value);

break;

case 11:

$this->setOngj($value);

break;

case 12:

$this->setOj($value);

break;

case 13:

$this->setOmot($value);

break;

case 14:

$this->setEv($value);

break;

case 15:

$this->setAsz($value);

break;

case 16:

$this->setBuszcs($value);

break;

case 17:

$this->setBusze($value);

break;

case 18:

$this->setObusz($value);

break;

case 19:

$this->setNyszer($value);

break;

case 20:

$this->setPotktgk($value);

break;

case 21:

$this->setKtgk($value);

break;

case 22:

$this->setNtgk($value);

break;

case 23:

$this->setKntgk($value);

break;

case 24:

$this->setOtgk($value);

break;

case 25:

$this->setSzgk($value);

break;

case 26:

$this->setOszgk($value);

break;

case 27:

$this->setMkp($value);

break;

case 28:

$this->setKpf($value);

break;

case 29:

$this->setLassu($value);

break;

case 30:

$this->setSpec($value);

break;

case 31:

$this->setFmegb($value);

break;

case 32:

$this->setAdatforr($value);

break;

case 33:

$this->setSzamlnap($value);

break;

case 34:

$this->setJelleg1($value);

break;

case 35:

$this->setJelleg2($value);

break;

case 36:

$this->setFmegj($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgNodeAttribsTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setShapeId($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setKszam($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setPkod($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setKkod($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setVvkod($arr[$keys[4]]);

}

if (array\_key\_exists($keys[5], $arr)) {

$this->setKszelv($arr[$keys[5]]);

}

if (array\_key\_exists($keys[6], $arr)) {

$this->setVszelv($arr[$keys[6]]);

}

if (array\_key\_exists($keys[7], $arr)) {

$this->setRshossz($arr[$keys[7]]);

}

if (array\_key\_exists($keys[8], $arr)) {

$this->setAnf($arr[$keys[8]]);

}

if (array\_key\_exists($keys[9], $arr)) {

$this->setAnet($arr[$keys[9]]);

}

if (array\_key\_exists($keys[10], $arr)) {

$this->setMof($arr[$keys[10]]);

}

if (array\_key\_exists($keys[11], $arr)) {

$this->setOngj($arr[$keys[11]]);

}

if (array\_key\_exists($keys[12], $arr)) {

$this->setOj($arr[$keys[12]]);

}

if (array\_key\_exists($keys[13], $arr)) {

$this->setOmot($arr[$keys[13]]);

}

if (array\_key\_exists($keys[14], $arr)) {

$this->setEv($arr[$keys[14]]);

}

if (array\_key\_exists($keys[15], $arr)) {

$this->setAsz($arr[$keys[15]]);

}

if (array\_key\_exists($keys[16], $arr)) {

$this->setBuszcs($arr[$keys[16]]);

}

if (array\_key\_exists($keys[17], $arr)) {

$this->setBusze($arr[$keys[17]]);

}

if (array\_key\_exists($keys[18], $arr)) {

$this->setObusz($arr[$keys[18]]);

}

if (array\_key\_exists($keys[19], $arr)) {

$this->setNyszer($arr[$keys[19]]);

}

if (array\_key\_exists($keys[20], $arr)) {

$this->setPotktgk($arr[$keys[20]]);

}

if (array\_key\_exists($keys[21], $arr)) {

$this->setKtgk($arr[$keys[21]]);

}

if (array\_key\_exists($keys[22], $arr)) {

$this->setNtgk($arr[$keys[22]]);

}

if (array\_key\_exists($keys[23], $arr)) {

$this->setKntgk($arr[$keys[23]]);

}

if (array\_key\_exists($keys[24], $arr)) {

$this->setOtgk($arr[$keys[24]]);

}

if (array\_key\_exists($keys[25], $arr)) {

$this->setSzgk($arr[$keys[25]]);

}

if (array\_key\_exists($keys[26], $arr)) {

$this->setOszgk($arr[$keys[26]]);

}

if (array\_key\_exists($keys[27], $arr)) {

$this->setMkp($arr[$keys[27]]);

}

if (array\_key\_exists($keys[28], $arr)) {

$this->setKpf($arr[$keys[28]]);

}

if (array\_key\_exists($keys[29], $arr)) {

$this->setLassu($arr[$keys[29]]);

}

if (array\_key\_exists($keys[30], $arr)) {

$this->setSpec($arr[$keys[30]]);

}

if (array\_key\_exists($keys[31], $arr)) {

$this->setFmegb($arr[$keys[31]]);

}

if (array\_key\_exists($keys[32], $arr)) {

$this->setAdatforr($arr[$keys[32]]);

}

if (array\_key\_exists($keys[33], $arr)) {

$this->setSzamlnap($arr[$keys[33]]);

}

if (array\_key\_exists($keys[34], $arr)) {

$this->setJelleg1($arr[$keys[34]]);

}

if (array\_key\_exists($keys[35], $arr)) {

$this->setJelleg2($arr[$keys[35]]);

}

if (array\_key\_exists($keys[36], $arr)) {

$this->setFmegj($arr[$keys[36]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgNodeAttribs The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgNodeAttribsTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_SHAPE\_ID)) {

$criteria->add(StgNodeAttribsTableMap::COL\_SHAPE\_ID, $this->shape\_id);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_KSZAM)) {

$criteria->add(StgNodeAttribsTableMap::COL\_KSZAM, $this->kszam);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_PKOD)) {

$criteria->add(StgNodeAttribsTableMap::COL\_PKOD, $this->pkod);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_KKOD)) {

$criteria->add(StgNodeAttribsTableMap::COL\_KKOD, $this->kkod);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_VVKOD)) {

$criteria->add(StgNodeAttribsTableMap::COL\_VVKOD, $this->vvkod);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_KSZELV)) {

$criteria->add(StgNodeAttribsTableMap::COL\_KSZELV, $this->kszelv);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_VSZELV)) {

$criteria->add(StgNodeAttribsTableMap::COL\_VSZELV, $this->vszelv);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_RSHOSSZ)) {

$criteria->add(StgNodeAttribsTableMap::COL\_RSHOSSZ, $this->rshossz);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_ANF)) {

$criteria->add(StgNodeAttribsTableMap::COL\_ANF, $this->anf);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_ANET)) {

$criteria->add(StgNodeAttribsTableMap::COL\_ANET, $this->anet);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_MOF)) {

$criteria->add(StgNodeAttribsTableMap::COL\_MOF, $this->mof);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_ONGJ)) {

$criteria->add(StgNodeAttribsTableMap::COL\_ONGJ, $this->ongj);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_OJ)) {

$criteria->add(StgNodeAttribsTableMap::COL\_OJ, $this->oj);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_OMOT)) {

$criteria->add(StgNodeAttribsTableMap::COL\_OMOT, $this->omot);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_EV)) {

$criteria->add(StgNodeAttribsTableMap::COL\_EV, $this->ev);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_ASZ)) {

$criteria->add(StgNodeAttribsTableMap::COL\_ASZ, $this->asz);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_BUSZCS)) {

$criteria->add(StgNodeAttribsTableMap::COL\_BUSZCS, $this->buszcs);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_BUSZE)) {

$criteria->add(StgNodeAttribsTableMap::COL\_BUSZE, $this->busze);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_OBUSZ)) {

$criteria->add(StgNodeAttribsTableMap::COL\_OBUSZ, $this->obusz);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_NYSZER)) {

$criteria->add(StgNodeAttribsTableMap::COL\_NYSZER, $this->nyszer);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_POTKTGK)) {

$criteria->add(StgNodeAttribsTableMap::COL\_POTKTGK, $this->potktgk);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_KTGK)) {

$criteria->add(StgNodeAttribsTableMap::COL\_KTGK, $this->ktgk);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_NTGK)) {

$criteria->add(StgNodeAttribsTableMap::COL\_NTGK, $this->ntgk);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_KNTGK)) {

$criteria->add(StgNodeAttribsTableMap::COL\_KNTGK, $this->kntgk);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_OTGK)) {

$criteria->add(StgNodeAttribsTableMap::COL\_OTGK, $this->otgk);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_SZGK)) {

$criteria->add(StgNodeAttribsTableMap::COL\_SZGK, $this->szgk);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_OSZGK)) {

$criteria->add(StgNodeAttribsTableMap::COL\_OSZGK, $this->oszgk);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_MKP)) {

$criteria->add(StgNodeAttribsTableMap::COL\_MKP, $this->mkp);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_KPF)) {

$criteria->add(StgNodeAttribsTableMap::COL\_KPF, $this->kpf);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_LASSU)) {

$criteria->add(StgNodeAttribsTableMap::COL\_LASSU, $this->lassu);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_SPEC)) {

$criteria->add(StgNodeAttribsTableMap::COL\_SPEC, $this->spec);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_FMEGB)) {

$criteria->add(StgNodeAttribsTableMap::COL\_FMEGB, $this->fmegb);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_ADATFORR)) {

$criteria->add(StgNodeAttribsTableMap::COL\_ADATFORR, $this->adatforr);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_SZAMLNAP)) {

$criteria->add(StgNodeAttribsTableMap::COL\_SZAMLNAP, $this->szamlnap);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_JELLEG1)) {

$criteria->add(StgNodeAttribsTableMap::COL\_JELLEG1, $this->jelleg1);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_JELLEG2)) {

$criteria->add(StgNodeAttribsTableMap::COL\_JELLEG2, $this->jelleg2);

}

if ($this->isColumnModified(StgNodeAttribsTableMap::COL\_FMEGJ)) {

$criteria->add(StgNodeAttribsTableMap::COL\_FMEGJ, $this->fmegj);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgNodeAttribs object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgNodeAttribs (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setShapeId($this->getShapeId());

$copyObj->setKszam($this->getKszam());

$copyObj->setPkod($this->getPkod());

$copyObj->setKkod($this->getKkod());

$copyObj->setVvkod($this->getVvkod());

$copyObj->setKszelv($this->getKszelv());

$copyObj->setVszelv($this->getVszelv());

$copyObj->setRshossz($this->getRshossz());

$copyObj->setAnf($this->getAnf());

$copyObj->setAnet($this->getAnet());

$copyObj->setMof($this->getMof());

$copyObj->setOngj($this->getOngj());

$copyObj->setOj($this->getOj());

$copyObj->setOmot($this->getOmot());

$copyObj->setEv($this->getEv());

$copyObj->setAsz($this->getAsz());

$copyObj->setBuszcs($this->getBuszcs());

$copyObj->setBusze($this->getBusze());

$copyObj->setObusz($this->getObusz());

$copyObj->setNyszer($this->getNyszer());

$copyObj->setPotktgk($this->getPotktgk());

$copyObj->setKtgk($this->getKtgk());

$copyObj->setNtgk($this->getNtgk());

$copyObj->setKntgk($this->getKntgk());

$copyObj->setOtgk($this->getOtgk());

$copyObj->setSzgk($this->getSzgk());

$copyObj->setOszgk($this->getOszgk());

$copyObj->setMkp($this->getMkp());

$copyObj->setKpf($this->getKpf());

$copyObj->setLassu($this->getLassu());

$copyObj->setSpec($this->getSpec());

$copyObj->setFmegb($this->getFmegb());

$copyObj->setAdatforr($this->getAdatforr());

$copyObj->setSzamlnap($this->getSzamlnap());

$copyObj->setJelleg1($this->getJelleg1());

$copyObj->setJelleg2($this->getJelleg2());

$copyObj->setFmegj($this->getFmegj());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgNodeAttribs Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->shape\_id = null;

$this->kszam = null;

$this->pkod = null;

$this->kkod = null;

$this->vvkod = null;

$this->kszelv = null;

$this->vszelv = null;

$this->rshossz = null;

$this->anf = null;

$this->anet = null;

$this->mof = null;

$this->ongj = null;

$this->oj = null;

$this->omot = null;

$this->ev = null;

$this->asz = null;

$this->buszcs = null;

$this->busze = null;

$this->obusz = null;

$this->nyszer = null;

$this->potktgk = null;

$this->ktgk = null;

$this->ntgk = null;

$this->kntgk = null;

$this->otgk = null;

$this->szgk = null;

$this->oszgk = null;

$this->mkp = null;

$this->kpf = null;

$this->lassu = null;

$this->spec = null;

$this->fmegb = null;

$this->adatforr = null;

$this->szamlnap = null;

$this->jelleg1 = null;

$this->jelleg2 = null;

$this->fmegj = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgNodeAttribsTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgNodeAttribsQuery.php

<?php

namespace Base;

use \StgNodeAttribs as ChildStgNodeAttribs;

use \StgNodeAttribsQuery as ChildStgNodeAttribsQuery;

use \Exception;

use Map\StgNodeAttribsTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_node\_attribs' table.

\*

\*

\*

\* @method ChildStgNodeAttribsQuery orderByShapeId($order = Criteria::ASC) Order by the shape\_id column

\* @method ChildStgNodeAttribsQuery orderByKszam($order = Criteria::ASC) Order by the kszam column

\* @method ChildStgNodeAttribsQuery orderByPkod($order = Criteria::ASC) Order by the pkod column

\* @method ChildStgNodeAttribsQuery orderByKkod($order = Criteria::ASC) Order by the kkod column

\* @method ChildStgNodeAttribsQuery orderByVvkod($order = Criteria::ASC) Order by the vvkod column

\* @method ChildStgNodeAttribsQuery orderByKszelv($order = Criteria::ASC) Order by the kszelv column

\* @method ChildStgNodeAttribsQuery orderByVszelv($order = Criteria::ASC) Order by the vszelv column

\* @method ChildStgNodeAttribsQuery orderByRshossz($order = Criteria::ASC) Order by the RSHOSSZ column

\* @method ChildStgNodeAttribsQuery orderByAnf($order = Criteria::ASC) Order by the ANF column

\* @method ChildStgNodeAttribsQuery orderByAnet($order = Criteria::ASC) Order by the ANET column

\* @method ChildStgNodeAttribsQuery orderByMof($order = Criteria::ASC) Order by the MOF column

\* @method ChildStgNodeAttribsQuery orderByOngj($order = Criteria::ASC) Order by the ONGJ column

\* @method ChildStgNodeAttribsQuery orderByOj($order = Criteria::ASC) Order by the OJ column

\* @method ChildStgNodeAttribsQuery orderByOmot($order = Criteria::ASC) Order by the OMOT column

\* @method ChildStgNodeAttribsQuery orderByEv($order = Criteria::ASC) Order by the EV column

\* @method ChildStgNodeAttribsQuery orderByAsz($order = Criteria::ASC) Order by the ASZ column

\* @method ChildStgNodeAttribsQuery orderByBuszcs($order = Criteria::ASC) Order by the BUSZCS column

\* @method ChildStgNodeAttribsQuery orderByBusze($order = Criteria::ASC) Order by the BUSZE column

\* @method ChildStgNodeAttribsQuery orderByObusz($order = Criteria::ASC) Order by the OBUSZ column

\* @method ChildStgNodeAttribsQuery orderByNyszer($order = Criteria::ASC) Order by the NYSZER column

\* @method ChildStgNodeAttribsQuery orderByPotktgk($order = Criteria::ASC) Order by the POTKTGK column

\* @method ChildStgNodeAttribsQuery orderByKtgk($order = Criteria::ASC) Order by the KTGK column

\* @method ChildStgNodeAttribsQuery orderByNtgk($order = Criteria::ASC) Order by the NTGK column

\* @method ChildStgNodeAttribsQuery orderByKntgk($order = Criteria::ASC) Order by the KNTGK column

\* @method ChildStgNodeAttribsQuery orderByOtgk($order = Criteria::ASC) Order by the OTGK column

\* @method ChildStgNodeAttribsQuery orderBySzgk($order = Criteria::ASC) Order by the SZGK column

\* @method ChildStgNodeAttribsQuery orderByOszgk($order = Criteria::ASC) Order by the OSZGK column

\* @method ChildStgNodeAttribsQuery orderByMkp($order = Criteria::ASC) Order by the MKP column

\* @method ChildStgNodeAttribsQuery orderByKpf($order = Criteria::ASC) Order by the KPF column

\* @method ChildStgNodeAttribsQuery orderByLassu($order = Criteria::ASC) Order by the LASSU column

\* @method ChildStgNodeAttribsQuery orderBySpec($order = Criteria::ASC) Order by the SPEC column

\* @method ChildStgNodeAttribsQuery orderByFmegb($order = Criteria::ASC) Order by the FMEGB column

\* @method ChildStgNodeAttribsQuery orderByAdatforr($order = Criteria::ASC) Order by the ADATFORR column

\* @method ChildStgNodeAttribsQuery orderBySzamlnap($order = Criteria::ASC) Order by the SZAMLNAP column

\* @method ChildStgNodeAttribsQuery orderByJelleg1($order = Criteria::ASC) Order by the JELLEG1 column

\* @method ChildStgNodeAttribsQuery orderByJelleg2($order = Criteria::ASC) Order by the JELLEG2 column

\* @method ChildStgNodeAttribsQuery orderByFmegj($order = Criteria::ASC) Order by the FMEGJ column

\*

\* @method ChildStgNodeAttribsQuery groupByShapeId() Group by the shape\_id column

\* @method ChildStgNodeAttribsQuery groupByKszam() Group by the kszam column

\* @method ChildStgNodeAttribsQuery groupByPkod() Group by the pkod column

\* @method ChildStgNodeAttribsQuery groupByKkod() Group by the kkod column

\* @method ChildStgNodeAttribsQuery groupByVvkod() Group by the vvkod column

\* @method ChildStgNodeAttribsQuery groupByKszelv() Group by the kszelv column

\* @method ChildStgNodeAttribsQuery groupByVszelv() Group by the vszelv column

\* @method ChildStgNodeAttribsQuery groupByRshossz() Group by the RSHOSSZ column

\* @method ChildStgNodeAttribsQuery groupByAnf() Group by the ANF column

\* @method ChildStgNodeAttribsQuery groupByAnet() Group by the ANET column

\* @method ChildStgNodeAttribsQuery groupByMof() Group by the MOF column

\* @method ChildStgNodeAttribsQuery groupByOngj() Group by the ONGJ column

\* @method ChildStgNodeAttribsQuery groupByOj() Group by the OJ column

\* @method ChildStgNodeAttribsQuery groupByOmot() Group by the OMOT column

\* @method ChildStgNodeAttribsQuery groupByEv() Group by the EV column

\* @method ChildStgNodeAttribsQuery groupByAsz() Group by the ASZ column

\* @method ChildStgNodeAttribsQuery groupByBuszcs() Group by the BUSZCS column

\* @method ChildStgNodeAttribsQuery groupByBusze() Group by the BUSZE column

\* @method ChildStgNodeAttribsQuery groupByObusz() Group by the OBUSZ column

\* @method ChildStgNodeAttribsQuery groupByNyszer() Group by the NYSZER column

\* @method ChildStgNodeAttribsQuery groupByPotktgk() Group by the POTKTGK column

\* @method ChildStgNodeAttribsQuery groupByKtgk() Group by the KTGK column

\* @method ChildStgNodeAttribsQuery groupByNtgk() Group by the NTGK column

\* @method ChildStgNodeAttribsQuery groupByKntgk() Group by the KNTGK column

\* @method ChildStgNodeAttribsQuery groupByOtgk() Group by the OTGK column

\* @method ChildStgNodeAttribsQuery groupBySzgk() Group by the SZGK column

\* @method ChildStgNodeAttribsQuery groupByOszgk() Group by the OSZGK column

\* @method ChildStgNodeAttribsQuery groupByMkp() Group by the MKP column

\* @method ChildStgNodeAttribsQuery groupByKpf() Group by the KPF column

\* @method ChildStgNodeAttribsQuery groupByLassu() Group by the LASSU column

\* @method ChildStgNodeAttribsQuery groupBySpec() Group by the SPEC column

\* @method ChildStgNodeAttribsQuery groupByFmegb() Group by the FMEGB column

\* @method ChildStgNodeAttribsQuery groupByAdatforr() Group by the ADATFORR column

\* @method ChildStgNodeAttribsQuery groupBySzamlnap() Group by the SZAMLNAP column

\* @method ChildStgNodeAttribsQuery groupByJelleg1() Group by the JELLEG1 column

\* @method ChildStgNodeAttribsQuery groupByJelleg2() Group by the JELLEG2 column

\* @method ChildStgNodeAttribsQuery groupByFmegj() Group by the FMEGJ column

\*

\* @method ChildStgNodeAttribsQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgNodeAttribsQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgNodeAttribsQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgNodeAttribs findOne(ConnectionInterface $con = null) Return the first ChildStgNodeAttribs matching the query

\* @method ChildStgNodeAttribs findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgNodeAttribs matching the query, or a new ChildStgNodeAttribs object populated from the query conditions when no match is found

\*

\* @method ChildStgNodeAttribs findOneByShapeId(int $shape\_id) Return the first ChildStgNodeAttribs filtered by the shape\_id column

\* @method ChildStgNodeAttribs findOneByKszam(string $kszam) Return the first ChildStgNodeAttribs filtered by the kszam column

\* @method ChildStgNodeAttribs findOneByPkod(int $pkod) Return the first ChildStgNodeAttribs filtered by the pkod column

\* @method ChildStgNodeAttribs findOneByKkod(string $kkod) Return the first ChildStgNodeAttribs filtered by the kkod column

\* @method ChildStgNodeAttribs findOneByVvkod(string $vvkod) Return the first ChildStgNodeAttribs filtered by the vvkod column

\* @method ChildStgNodeAttribs findOneByKszelv(string $kszelv) Return the first ChildStgNodeAttribs filtered by the kszelv column

\* @method ChildStgNodeAttribs findOneByVszelv(string $vszelv) Return the first ChildStgNodeAttribs filtered by the vszelv column

\* @method ChildStgNodeAttribs findOneByRshossz(double $RSHOSSZ) Return the first ChildStgNodeAttribs filtered by the RSHOSSZ column

\* @method ChildStgNodeAttribs findOneByAnf(int $ANF) Return the first ChildStgNodeAttribs filtered by the ANF column

\* @method ChildStgNodeAttribs findOneByAnet(int $ANET) Return the first ChildStgNodeAttribs filtered by the ANET column

\* @method ChildStgNodeAttribs findOneByMof(int $MOF) Return the first ChildStgNodeAttribs filtered by the MOF column

\* @method ChildStgNodeAttribs findOneByOngj(int $ONGJ) Return the first ChildStgNodeAttribs filtered by the ONGJ column

\* @method ChildStgNodeAttribs findOneByOj(int $OJ) Return the first ChildStgNodeAttribs filtered by the OJ column

\* @method ChildStgNodeAttribs findOneByOmot(int $OMOT) Return the first ChildStgNodeAttribs filtered by the OMOT column

\* @method ChildStgNodeAttribs findOneByEv(int $EV) Return the first ChildStgNodeAttribs filtered by the EV column

\* @method ChildStgNodeAttribs findOneByAsz(int $ASZ) Return the first ChildStgNodeAttribs filtered by the ASZ column

\* @method ChildStgNodeAttribs findOneByBuszcs(int $BUSZCS) Return the first ChildStgNodeAttribs filtered by the BUSZCS column

\* @method ChildStgNodeAttribs findOneByBusze(int $BUSZE) Return the first ChildStgNodeAttribs filtered by the BUSZE column

\* @method ChildStgNodeAttribs findOneByObusz(int $OBUSZ) Return the first ChildStgNodeAttribs filtered by the OBUSZ column

\* @method ChildStgNodeAttribs findOneByNyszer(int $NYSZER) Return the first ChildStgNodeAttribs filtered by the NYSZER column

\* @method ChildStgNodeAttribs findOneByPotktgk(int $POTKTGK) Return the first ChildStgNodeAttribs filtered by the POTKTGK column

\* @method ChildStgNodeAttribs findOneByKtgk(int $KTGK) Return the first ChildStgNodeAttribs filtered by the KTGK column

\* @method ChildStgNodeAttribs findOneByNtgk(int $NTGK) Return the first ChildStgNodeAttribs filtered by the NTGK column

\* @method ChildStgNodeAttribs findOneByKntgk(int $KNTGK) Return the first ChildStgNodeAttribs filtered by the KNTGK column

\* @method ChildStgNodeAttribs findOneByOtgk(int $OTGK) Return the first ChildStgNodeAttribs filtered by the OTGK column

\* @method ChildStgNodeAttribs findOneBySzgk(int $SZGK) Return the first ChildStgNodeAttribs filtered by the SZGK column

\* @method ChildStgNodeAttribs findOneByOszgk(int $OSZGK) Return the first ChildStgNodeAttribs filtered by the OSZGK column

\* @method ChildStgNodeAttribs findOneByMkp(int $MKP) Return the first ChildStgNodeAttribs filtered by the MKP column

\* @method ChildStgNodeAttribs findOneByKpf(int $KPF) Return the first ChildStgNodeAttribs filtered by the KPF column

\* @method ChildStgNodeAttribs findOneByLassu(int $LASSU) Return the first ChildStgNodeAttribs filtered by the LASSU column

\* @method ChildStgNodeAttribs findOneBySpec(int $SPEC) Return the first ChildStgNodeAttribs filtered by the SPEC column

\* @method ChildStgNodeAttribs findOneByFmegb(string $FMEGB) Return the first ChildStgNodeAttribs filtered by the FMEGB column

\* @method ChildStgNodeAttribs findOneByAdatforr(string $ADATFORR) Return the first ChildStgNodeAttribs filtered by the ADATFORR column

\* @method ChildStgNodeAttribs findOneBySzamlnap(int $SZAMLNAP) Return the first ChildStgNodeAttribs filtered by the SZAMLNAP column

\* @method ChildStgNodeAttribs findOneByJelleg1(string $JELLEG1) Return the first ChildStgNodeAttribs filtered by the JELLEG1 column

\* @method ChildStgNodeAttribs findOneByJelleg2(string $JELLEG2) Return the first ChildStgNodeAttribs filtered by the JELLEG2 column

\* @method ChildStgNodeAttribs findOneByFmegj(string $FMEGJ) Return the first ChildStgNodeAttribs filtered by the FMEGJ column

\*

\* @method ChildStgNodeAttribs[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgNodeAttribs objects based on current ModelCriteria

\* @method ChildStgNodeAttribs[]|ObjectCollection findByShapeId(int $shape\_id) Return ChildStgNodeAttribs objects filtered by the shape\_id column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByKszam(string $kszam) Return ChildStgNodeAttribs objects filtered by the kszam column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByPkod(int $pkod) Return ChildStgNodeAttribs objects filtered by the pkod column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByKkod(string $kkod) Return ChildStgNodeAttribs objects filtered by the kkod column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByVvkod(string $vvkod) Return ChildStgNodeAttribs objects filtered by the vvkod column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByKszelv(string $kszelv) Return ChildStgNodeAttribs objects filtered by the kszelv column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByVszelv(string $vszelv) Return ChildStgNodeAttribs objects filtered by the vszelv column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByRshossz(double $RSHOSSZ) Return ChildStgNodeAttribs objects filtered by the RSHOSSZ column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByAnf(int $ANF) Return ChildStgNodeAttribs objects filtered by the ANF column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByAnet(int $ANET) Return ChildStgNodeAttribs objects filtered by the ANET column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByMof(int $MOF) Return ChildStgNodeAttribs objects filtered by the MOF column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByOngj(int $ONGJ) Return ChildStgNodeAttribs objects filtered by the ONGJ column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByOj(int $OJ) Return ChildStgNodeAttribs objects filtered by the OJ column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByOmot(int $OMOT) Return ChildStgNodeAttribs objects filtered by the OMOT column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByEv(int $EV) Return ChildStgNodeAttribs objects filtered by the EV column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByAsz(int $ASZ) Return ChildStgNodeAttribs objects filtered by the ASZ column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByBuszcs(int $BUSZCS) Return ChildStgNodeAttribs objects filtered by the BUSZCS column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByBusze(int $BUSZE) Return ChildStgNodeAttribs objects filtered by the BUSZE column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByObusz(int $OBUSZ) Return ChildStgNodeAttribs objects filtered by the OBUSZ column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByNyszer(int $NYSZER) Return ChildStgNodeAttribs objects filtered by the NYSZER column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByPotktgk(int $POTKTGK) Return ChildStgNodeAttribs objects filtered by the POTKTGK column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByKtgk(int $KTGK) Return ChildStgNodeAttribs objects filtered by the KTGK column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByNtgk(int $NTGK) Return ChildStgNodeAttribs objects filtered by the NTGK column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByKntgk(int $KNTGK) Return ChildStgNodeAttribs objects filtered by the KNTGK column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByOtgk(int $OTGK) Return ChildStgNodeAttribs objects filtered by the OTGK column

\* @method ChildStgNodeAttribs[]|ObjectCollection findBySzgk(int $SZGK) Return ChildStgNodeAttribs objects filtered by the SZGK column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByOszgk(int $OSZGK) Return ChildStgNodeAttribs objects filtered by the OSZGK column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByMkp(int $MKP) Return ChildStgNodeAttribs objects filtered by the MKP column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByKpf(int $KPF) Return ChildStgNodeAttribs objects filtered by the KPF column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByLassu(int $LASSU) Return ChildStgNodeAttribs objects filtered by the LASSU column

\* @method ChildStgNodeAttribs[]|ObjectCollection findBySpec(int $SPEC) Return ChildStgNodeAttribs objects filtered by the SPEC column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByFmegb(string $FMEGB) Return ChildStgNodeAttribs objects filtered by the FMEGB column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByAdatforr(string $ADATFORR) Return ChildStgNodeAttribs objects filtered by the ADATFORR column

\* @method ChildStgNodeAttribs[]|ObjectCollection findBySzamlnap(int $SZAMLNAP) Return ChildStgNodeAttribs objects filtered by the SZAMLNAP column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByJelleg1(string $JELLEG1) Return ChildStgNodeAttribs objects filtered by the JELLEG1 column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByJelleg2(string $JELLEG2) Return ChildStgNodeAttribs objects filtered by the JELLEG2 column

\* @method ChildStgNodeAttribs[]|ObjectCollection findByFmegj(string $FMEGJ) Return ChildStgNodeAttribs objects filtered by the FMEGJ column

\* @method ChildStgNodeAttribs[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgNodeAttribsQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgNodeAttribsQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgNodeAttribs', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgNodeAttribsQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgNodeAttribsQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgNodeAttribsQuery) {

return $criteria;

}

$query = new ChildStgNodeAttribsQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgNodeAttribs|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgNodeAttribs object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgNodeAttribs object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgNodeAttribs object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgNodeAttribs object has no primary key');

}

/\*\*

\* Filter the query on the shape\_id column

\*

\* Example usage:

\* <code>

\* $query->filterByShapeId(1234); // WHERE shape\_id = 1234

\* $query->filterByShapeId(array(12, 34)); // WHERE shape\_id IN (12, 34)

\* $query->filterByShapeId(array('min' => 12)); // WHERE shape\_id > 12

\* </code>

\*

\* @param mixed $shapeId The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByShapeId($shapeId = null, $comparison = null)

{

if (is\_array($shapeId)) {

$useMinMax = false;

if (isset($shapeId['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_SHAPE\_ID, $shapeId['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($shapeId['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_SHAPE\_ID, $shapeId['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_SHAPE\_ID, $shapeId, $comparison);

}

/\*\*

\* Filter the query on the kszam column

\*

\* Example usage:

\* <code>

\* $query->filterByKszam('fooValue'); // WHERE kszam = 'fooValue'

\* $query->filterByKszam('%fooValue%'); // WHERE kszam LIKE '%fooValue%'

\* </code>

\*

\* @param string $kszam The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByKszam($kszam = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($kszam)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $kszam)) {

$kszam = str\_replace('\*', '%', $kszam);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_KSZAM, $kszam, $comparison);

}

/\*\*

\* Filter the query on the pkod column

\*

\* Example usage:

\* <code>

\* $query->filterByPkod(1234); // WHERE pkod = 1234

\* $query->filterByPkod(array(12, 34)); // WHERE pkod IN (12, 34)

\* $query->filterByPkod(array('min' => 12)); // WHERE pkod > 12

\* </code>

\*

\* @param mixed $pkod The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByPkod($pkod = null, $comparison = null)

{

if (is\_array($pkod)) {

$useMinMax = false;

if (isset($pkod['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_PKOD, $pkod['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($pkod['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_PKOD, $pkod['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_PKOD, $pkod, $comparison);

}

/\*\*

\* Filter the query on the kkod column

\*

\* Example usage:

\* <code>

\* $query->filterByKkod('fooValue'); // WHERE kkod = 'fooValue'

\* $query->filterByKkod('%fooValue%'); // WHERE kkod LIKE '%fooValue%'

\* </code>

\*

\* @param string $kkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByKkod($kkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($kkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $kkod)) {

$kkod = str\_replace('\*', '%', $kkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_KKOD, $kkod, $comparison);

}

/\*\*

\* Filter the query on the vvkod column

\*

\* Example usage:

\* <code>

\* $query->filterByVvkod('fooValue'); // WHERE vvkod = 'fooValue'

\* $query->filterByVvkod('%fooValue%'); // WHERE vvkod LIKE '%fooValue%'

\* </code>

\*

\* @param string $vvkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByVvkod($vvkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($vvkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $vvkod)) {

$vvkod = str\_replace('\*', '%', $vvkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_VVKOD, $vvkod, $comparison);

}

/\*\*

\* Filter the query on the kszelv column

\*

\* Example usage:

\* <code>

\* $query->filterByKszelv('fooValue'); // WHERE kszelv = 'fooValue'

\* $query->filterByKszelv('%fooValue%'); // WHERE kszelv LIKE '%fooValue%'

\* </code>

\*

\* @param string $kszelv The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByKszelv($kszelv = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($kszelv)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $kszelv)) {

$kszelv = str\_replace('\*', '%', $kszelv);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_KSZELV, $kszelv, $comparison);

}

/\*\*

\* Filter the query on the vszelv column

\*

\* Example usage:

\* <code>

\* $query->filterByVszelv('fooValue'); // WHERE vszelv = 'fooValue'

\* $query->filterByVszelv('%fooValue%'); // WHERE vszelv LIKE '%fooValue%'

\* </code>

\*

\* @param string $vszelv The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByVszelv($vszelv = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($vszelv)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $vszelv)) {

$vszelv = str\_replace('\*', '%', $vszelv);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_VSZELV, $vszelv, $comparison);

}

/\*\*

\* Filter the query on the RSHOSSZ column

\*

\* Example usage:

\* <code>

\* $query->filterByRshossz(1234); // WHERE RSHOSSZ = 1234

\* $query->filterByRshossz(array(12, 34)); // WHERE RSHOSSZ IN (12, 34)

\* $query->filterByRshossz(array('min' => 12)); // WHERE RSHOSSZ > 12

\* </code>

\*

\* @param mixed $rshossz The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByRshossz($rshossz = null, $comparison = null)

{

if (is\_array($rshossz)) {

$useMinMax = false;

if (isset($rshossz['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_RSHOSSZ, $rshossz['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($rshossz['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_RSHOSSZ, $rshossz['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_RSHOSSZ, $rshossz, $comparison);

}

/\*\*

\* Filter the query on the ANF column

\*

\* Example usage:

\* <code>

\* $query->filterByAnf(1234); // WHERE ANF = 1234

\* $query->filterByAnf(array(12, 34)); // WHERE ANF IN (12, 34)

\* $query->filterByAnf(array('min' => 12)); // WHERE ANF > 12

\* </code>

\*

\* @param mixed $anf The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByAnf($anf = null, $comparison = null)

{

if (is\_array($anf)) {

$useMinMax = false;

if (isset($anf['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_ANF, $anf['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($anf['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_ANF, $anf['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_ANF, $anf, $comparison);

}

/\*\*

\* Filter the query on the ANET column

\*

\* Example usage:

\* <code>

\* $query->filterByAnet(1234); // WHERE ANET = 1234

\* $query->filterByAnet(array(12, 34)); // WHERE ANET IN (12, 34)

\* $query->filterByAnet(array('min' => 12)); // WHERE ANET > 12

\* </code>

\*

\* @param mixed $anet The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByAnet($anet = null, $comparison = null)

{

if (is\_array($anet)) {

$useMinMax = false;

if (isset($anet['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_ANET, $anet['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($anet['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_ANET, $anet['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_ANET, $anet, $comparison);

}

/\*\*

\* Filter the query on the MOF column

\*

\* Example usage:

\* <code>

\* $query->filterByMof(1234); // WHERE MOF = 1234

\* $query->filterByMof(array(12, 34)); // WHERE MOF IN (12, 34)

\* $query->filterByMof(array('min' => 12)); // WHERE MOF > 12

\* </code>

\*

\* @param mixed $mof The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByMof($mof = null, $comparison = null)

{

if (is\_array($mof)) {

$useMinMax = false;

if (isset($mof['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_MOF, $mof['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($mof['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_MOF, $mof['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_MOF, $mof, $comparison);

}

/\*\*

\* Filter the query on the ONGJ column

\*

\* Example usage:

\* <code>

\* $query->filterByOngj(1234); // WHERE ONGJ = 1234

\* $query->filterByOngj(array(12, 34)); // WHERE ONGJ IN (12, 34)

\* $query->filterByOngj(array('min' => 12)); // WHERE ONGJ > 12

\* </code>

\*

\* @param mixed $ongj The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByOngj($ongj = null, $comparison = null)

{

if (is\_array($ongj)) {

$useMinMax = false;

if (isset($ongj['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_ONGJ, $ongj['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($ongj['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_ONGJ, $ongj['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_ONGJ, $ongj, $comparison);

}

/\*\*

\* Filter the query on the OJ column

\*

\* Example usage:

\* <code>

\* $query->filterByOj(1234); // WHERE OJ = 1234

\* $query->filterByOj(array(12, 34)); // WHERE OJ IN (12, 34)

\* $query->filterByOj(array('min' => 12)); // WHERE OJ > 12

\* </code>

\*

\* @param mixed $oj The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByOj($oj = null, $comparison = null)

{

if (is\_array($oj)) {

$useMinMax = false;

if (isset($oj['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_OJ, $oj['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($oj['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_OJ, $oj['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_OJ, $oj, $comparison);

}

/\*\*

\* Filter the query on the OMOT column

\*

\* Example usage:

\* <code>

\* $query->filterByOmot(1234); // WHERE OMOT = 1234

\* $query->filterByOmot(array(12, 34)); // WHERE OMOT IN (12, 34)

\* $query->filterByOmot(array('min' => 12)); // WHERE OMOT > 12

\* </code>

\*

\* @param mixed $omot The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByOmot($omot = null, $comparison = null)

{

if (is\_array($omot)) {

$useMinMax = false;

if (isset($omot['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_OMOT, $omot['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($omot['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_OMOT, $omot['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_OMOT, $omot, $comparison);

}

/\*\*

\* Filter the query on the EV column

\*

\* Example usage:

\* <code>

\* $query->filterByEv(1234); // WHERE EV = 1234

\* $query->filterByEv(array(12, 34)); // WHERE EV IN (12, 34)

\* $query->filterByEv(array('min' => 12)); // WHERE EV > 12

\* </code>

\*

\* @param mixed $ev The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByEv($ev = null, $comparison = null)

{

if (is\_array($ev)) {

$useMinMax = false;

if (isset($ev['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_EV, $ev['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($ev['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_EV, $ev['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_EV, $ev, $comparison);

}

/\*\*

\* Filter the query on the ASZ column

\*

\* Example usage:

\* <code>

\* $query->filterByAsz(1234); // WHERE ASZ = 1234

\* $query->filterByAsz(array(12, 34)); // WHERE ASZ IN (12, 34)

\* $query->filterByAsz(array('min' => 12)); // WHERE ASZ > 12

\* </code>

\*

\* @param mixed $asz The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByAsz($asz = null, $comparison = null)

{

if (is\_array($asz)) {

$useMinMax = false;

if (isset($asz['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_ASZ, $asz['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($asz['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_ASZ, $asz['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_ASZ, $asz, $comparison);

}

/\*\*

\* Filter the query on the BUSZCS column

\*

\* Example usage:

\* <code>

\* $query->filterByBuszcs(1234); // WHERE BUSZCS = 1234

\* $query->filterByBuszcs(array(12, 34)); // WHERE BUSZCS IN (12, 34)

\* $query->filterByBuszcs(array('min' => 12)); // WHERE BUSZCS > 12

\* </code>

\*

\* @param mixed $buszcs The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByBuszcs($buszcs = null, $comparison = null)

{

if (is\_array($buszcs)) {

$useMinMax = false;

if (isset($buszcs['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_BUSZCS, $buszcs['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($buszcs['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_BUSZCS, $buszcs['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_BUSZCS, $buszcs, $comparison);

}

/\*\*

\* Filter the query on the BUSZE column

\*

\* Example usage:

\* <code>

\* $query->filterByBusze(1234); // WHERE BUSZE = 1234

\* $query->filterByBusze(array(12, 34)); // WHERE BUSZE IN (12, 34)

\* $query->filterByBusze(array('min' => 12)); // WHERE BUSZE > 12

\* </code>

\*

\* @param mixed $busze The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByBusze($busze = null, $comparison = null)

{

if (is\_array($busze)) {

$useMinMax = false;

if (isset($busze['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_BUSZE, $busze['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($busze['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_BUSZE, $busze['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_BUSZE, $busze, $comparison);

}

/\*\*

\* Filter the query on the OBUSZ column

\*

\* Example usage:

\* <code>

\* $query->filterByObusz(1234); // WHERE OBUSZ = 1234

\* $query->filterByObusz(array(12, 34)); // WHERE OBUSZ IN (12, 34)

\* $query->filterByObusz(array('min' => 12)); // WHERE OBUSZ > 12

\* </code>

\*

\* @param mixed $obusz The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByObusz($obusz = null, $comparison = null)

{

if (is\_array($obusz)) {

$useMinMax = false;

if (isset($obusz['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_OBUSZ, $obusz['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($obusz['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_OBUSZ, $obusz['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_OBUSZ, $obusz, $comparison);

}

/\*\*

\* Filter the query on the NYSZER column

\*

\* Example usage:

\* <code>

\* $query->filterByNyszer(1234); // WHERE NYSZER = 1234

\* $query->filterByNyszer(array(12, 34)); // WHERE NYSZER IN (12, 34)

\* $query->filterByNyszer(array('min' => 12)); // WHERE NYSZER > 12

\* </code>

\*

\* @param mixed $nyszer The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByNyszer($nyszer = null, $comparison = null)

{

if (is\_array($nyszer)) {

$useMinMax = false;

if (isset($nyszer['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_NYSZER, $nyszer['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nyszer['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_NYSZER, $nyszer['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_NYSZER, $nyszer, $comparison);

}

/\*\*

\* Filter the query on the POTKTGK column

\*

\* Example usage:

\* <code>

\* $query->filterByPotktgk(1234); // WHERE POTKTGK = 1234

\* $query->filterByPotktgk(array(12, 34)); // WHERE POTKTGK IN (12, 34)

\* $query->filterByPotktgk(array('min' => 12)); // WHERE POTKTGK > 12

\* </code>

\*

\* @param mixed $potktgk The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByPotktgk($potktgk = null, $comparison = null)

{

if (is\_array($potktgk)) {

$useMinMax = false;

if (isset($potktgk['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_POTKTGK, $potktgk['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($potktgk['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_POTKTGK, $potktgk['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_POTKTGK, $potktgk, $comparison);

}

/\*\*

\* Filter the query on the KTGK column

\*

\* Example usage:

\* <code>

\* $query->filterByKtgk(1234); // WHERE KTGK = 1234

\* $query->filterByKtgk(array(12, 34)); // WHERE KTGK IN (12, 34)

\* $query->filterByKtgk(array('min' => 12)); // WHERE KTGK > 12

\* </code>

\*

\* @param mixed $ktgk The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByKtgk($ktgk = null, $comparison = null)

{

if (is\_array($ktgk)) {

$useMinMax = false;

if (isset($ktgk['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_KTGK, $ktgk['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($ktgk['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_KTGK, $ktgk['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_KTGK, $ktgk, $comparison);

}

/\*\*

\* Filter the query on the NTGK column

\*

\* Example usage:

\* <code>

\* $query->filterByNtgk(1234); // WHERE NTGK = 1234

\* $query->filterByNtgk(array(12, 34)); // WHERE NTGK IN (12, 34)

\* $query->filterByNtgk(array('min' => 12)); // WHERE NTGK > 12

\* </code>

\*

\* @param mixed $ntgk The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByNtgk($ntgk = null, $comparison = null)

{

if (is\_array($ntgk)) {

$useMinMax = false;

if (isset($ntgk['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_NTGK, $ntgk['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($ntgk['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_NTGK, $ntgk['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_NTGK, $ntgk, $comparison);

}

/\*\*

\* Filter the query on the KNTGK column

\*

\* Example usage:

\* <code>

\* $query->filterByKntgk(1234); // WHERE KNTGK = 1234

\* $query->filterByKntgk(array(12, 34)); // WHERE KNTGK IN (12, 34)

\* $query->filterByKntgk(array('min' => 12)); // WHERE KNTGK > 12

\* </code>

\*

\* @param mixed $kntgk The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByKntgk($kntgk = null, $comparison = null)

{

if (is\_array($kntgk)) {

$useMinMax = false;

if (isset($kntgk['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_KNTGK, $kntgk['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($kntgk['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_KNTGK, $kntgk['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_KNTGK, $kntgk, $comparison);

}

/\*\*

\* Filter the query on the OTGK column

\*

\* Example usage:

\* <code>

\* $query->filterByOtgk(1234); // WHERE OTGK = 1234

\* $query->filterByOtgk(array(12, 34)); // WHERE OTGK IN (12, 34)

\* $query->filterByOtgk(array('min' => 12)); // WHERE OTGK > 12

\* </code>

\*

\* @param mixed $otgk The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByOtgk($otgk = null, $comparison = null)

{

if (is\_array($otgk)) {

$useMinMax = false;

if (isset($otgk['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_OTGK, $otgk['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($otgk['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_OTGK, $otgk['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_OTGK, $otgk, $comparison);

}

/\*\*

\* Filter the query on the SZGK column

\*

\* Example usage:

\* <code>

\* $query->filterBySzgk(1234); // WHERE SZGK = 1234

\* $query->filterBySzgk(array(12, 34)); // WHERE SZGK IN (12, 34)

\* $query->filterBySzgk(array('min' => 12)); // WHERE SZGK > 12

\* </code>

\*

\* @param mixed $szgk The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterBySzgk($szgk = null, $comparison = null)

{

if (is\_array($szgk)) {

$useMinMax = false;

if (isset($szgk['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_SZGK, $szgk['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($szgk['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_SZGK, $szgk['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_SZGK, $szgk, $comparison);

}

/\*\*

\* Filter the query on the OSZGK column

\*

\* Example usage:

\* <code>

\* $query->filterByOszgk(1234); // WHERE OSZGK = 1234

\* $query->filterByOszgk(array(12, 34)); // WHERE OSZGK IN (12, 34)

\* $query->filterByOszgk(array('min' => 12)); // WHERE OSZGK > 12

\* </code>

\*

\* @param mixed $oszgk The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByOszgk($oszgk = null, $comparison = null)

{

if (is\_array($oszgk)) {

$useMinMax = false;

if (isset($oszgk['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_OSZGK, $oszgk['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($oszgk['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_OSZGK, $oszgk['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_OSZGK, $oszgk, $comparison);

}

/\*\*

\* Filter the query on the MKP column

\*

\* Example usage:

\* <code>

\* $query->filterByMkp(1234); // WHERE MKP = 1234

\* $query->filterByMkp(array(12, 34)); // WHERE MKP IN (12, 34)

\* $query->filterByMkp(array('min' => 12)); // WHERE MKP > 12

\* </code>

\*

\* @param mixed $mkp The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByMkp($mkp = null, $comparison = null)

{

if (is\_array($mkp)) {

$useMinMax = false;

if (isset($mkp['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_MKP, $mkp['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($mkp['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_MKP, $mkp['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_MKP, $mkp, $comparison);

}

/\*\*

\* Filter the query on the KPF column

\*

\* Example usage:

\* <code>

\* $query->filterByKpf(1234); // WHERE KPF = 1234

\* $query->filterByKpf(array(12, 34)); // WHERE KPF IN (12, 34)

\* $query->filterByKpf(array('min' => 12)); // WHERE KPF > 12

\* </code>

\*

\* @param mixed $kpf The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByKpf($kpf = null, $comparison = null)

{

if (is\_array($kpf)) {

$useMinMax = false;

if (isset($kpf['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_KPF, $kpf['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($kpf['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_KPF, $kpf['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_KPF, $kpf, $comparison);

}

/\*\*

\* Filter the query on the LASSU column

\*

\* Example usage:

\* <code>

\* $query->filterByLassu(1234); // WHERE LASSU = 1234

\* $query->filterByLassu(array(12, 34)); // WHERE LASSU IN (12, 34)

\* $query->filterByLassu(array('min' => 12)); // WHERE LASSU > 12

\* </code>

\*

\* @param mixed $lassu The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByLassu($lassu = null, $comparison = null)

{

if (is\_array($lassu)) {

$useMinMax = false;

if (isset($lassu['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_LASSU, $lassu['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($lassu['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_LASSU, $lassu['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_LASSU, $lassu, $comparison);

}

/\*\*

\* Filter the query on the SPEC column

\*

\* Example usage:

\* <code>

\* $query->filterBySpec(1234); // WHERE SPEC = 1234

\* $query->filterBySpec(array(12, 34)); // WHERE SPEC IN (12, 34)

\* $query->filterBySpec(array('min' => 12)); // WHERE SPEC > 12

\* </code>

\*

\* @param mixed $spec The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterBySpec($spec = null, $comparison = null)

{

if (is\_array($spec)) {

$useMinMax = false;

if (isset($spec['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_SPEC, $spec['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($spec['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_SPEC, $spec['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_SPEC, $spec, $comparison);

}

/\*\*

\* Filter the query on the FMEGB column

\*

\* Example usage:

\* <code>

\* $query->filterByFmegb('fooValue'); // WHERE FMEGB = 'fooValue'

\* $query->filterByFmegb('%fooValue%'); // WHERE FMEGB LIKE '%fooValue%'

\* </code>

\*

\* @param string $fmegb The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByFmegb($fmegb = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($fmegb)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $fmegb)) {

$fmegb = str\_replace('\*', '%', $fmegb);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_FMEGB, $fmegb, $comparison);

}

/\*\*

\* Filter the query on the ADATFORR column

\*

\* Example usage:

\* <code>

\* $query->filterByAdatforr('fooValue'); // WHERE ADATFORR = 'fooValue'

\* $query->filterByAdatforr('%fooValue%'); // WHERE ADATFORR LIKE '%fooValue%'

\* </code>

\*

\* @param string $adatforr The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByAdatforr($adatforr = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($adatforr)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $adatforr)) {

$adatforr = str\_replace('\*', '%', $adatforr);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_ADATFORR, $adatforr, $comparison);

}

/\*\*

\* Filter the query on the SZAMLNAP column

\*

\* Example usage:

\* <code>

\* $query->filterBySzamlnap(1234); // WHERE SZAMLNAP = 1234

\* $query->filterBySzamlnap(array(12, 34)); // WHERE SZAMLNAP IN (12, 34)

\* $query->filterBySzamlnap(array('min' => 12)); // WHERE SZAMLNAP > 12

\* </code>

\*

\* @param mixed $szamlnap The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterBySzamlnap($szamlnap = null, $comparison = null)

{

if (is\_array($szamlnap)) {

$useMinMax = false;

if (isset($szamlnap['min'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_SZAMLNAP, $szamlnap['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($szamlnap['max'])) {

$this->addUsingAlias(StgNodeAttribsTableMap::COL\_SZAMLNAP, $szamlnap['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_SZAMLNAP, $szamlnap, $comparison);

}

/\*\*

\* Filter the query on the JELLEG1 column

\*

\* Example usage:

\* <code>

\* $query->filterByJelleg1('fooValue'); // WHERE JELLEG1 = 'fooValue'

\* $query->filterByJelleg1('%fooValue%'); // WHERE JELLEG1 LIKE '%fooValue%'

\* </code>

\*

\* @param string $jelleg1 The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByJelleg1($jelleg1 = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($jelleg1)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $jelleg1)) {

$jelleg1 = str\_replace('\*', '%', $jelleg1);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_JELLEG1, $jelleg1, $comparison);

}

/\*\*

\* Filter the query on the JELLEG2 column

\*

\* Example usage:

\* <code>

\* $query->filterByJelleg2('fooValue'); // WHERE JELLEG2 = 'fooValue'

\* $query->filterByJelleg2('%fooValue%'); // WHERE JELLEG2 LIKE '%fooValue%'

\* </code>

\*

\* @param string $jelleg2 The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByJelleg2($jelleg2 = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($jelleg2)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $jelleg2)) {

$jelleg2 = str\_replace('\*', '%', $jelleg2);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_JELLEG2, $jelleg2, $comparison);

}

/\*\*

\* Filter the query on the FMEGJ column

\*

\* Example usage:

\* <code>

\* $query->filterByFmegj('fooValue'); // WHERE FMEGJ = 'fooValue'

\* $query->filterByFmegj('%fooValue%'); // WHERE FMEGJ LIKE '%fooValue%'

\* </code>

\*

\* @param string $fmegj The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function filterByFmegj($fmegj = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($fmegj)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $fmegj)) {

$fmegj = str\_replace('\*', '%', $fmegj);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgNodeAttribsTableMap::COL\_FMEGJ, $fmegj, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgNodeAttribs $stgNodeAttribs Object to remove from the list of results

\*

\* @return $this|ChildStgNodeAttribsQuery The current query, for fluid interface

\*/

public function prune($stgNodeAttribs = null)

{

if ($stgNodeAttribs) {

throw new LogicException('StgNodeAttribs object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_node\_attribs table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNodeAttribsTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgNodeAttribsTableMap::clearInstancePool();

StgNodeAttribsTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNodeAttribsTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgNodeAttribsTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgNodeAttribsTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgNodeAttribsTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgNodeAttribsQuery

#### StgNodes.php

<?php

namespace Base;

use \StgNodesQuery as ChildStgNodesQuery;

use \Exception;

use \PDO;

use Map\StgNodesTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgNodes implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgNodesTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the shape\_id field.

\* @var int

\*/

protected $shape\_id;

/\*\*

\* The value for the x field.

\* @var double

\*/

protected $x;

/\*\*

\* The value for the y field.

\* @var double

\*/

protected $y;

/\*\*

\* The value for the number field.

\* @var string

\*/

protected $number;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgNodes object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgNodes</code> instance. If

\* <code>obj</code> is an instance of <code>StgNodes</code>, delegates to

\* <code>equals(StgNodes)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgNodes The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [shape\_id] column value.

\*

\* @return int

\*/

public function getShapeId()

{

return $this->shape\_id;

}

/\*\*

\* Get the [x] column value.

\*

\* @return double

\*/

public function getX()

{

return $this->x;

}

/\*\*

\* Get the [y] column value.

\*

\* @return double

\*/

public function getY()

{

return $this->y;

}

/\*\*

\* Get the [number] column value.

\*

\* @return string

\*/

public function getNumber()

{

return $this->number;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgNodesTableMap::translateFieldName('ShapeId', TableMap::TYPE\_PHPNAME, $indexType)];

$this->shape\_id = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgNodesTableMap::translateFieldName('X', TableMap::TYPE\_PHPNAME, $indexType)];

$this->x = (null !== $col) ? (double) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgNodesTableMap::translateFieldName('Y', TableMap::TYPE\_PHPNAME, $indexType)];

$this->y = (null !== $col) ? (double) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgNodesTableMap::translateFieldName('Number', TableMap::TYPE\_PHPNAME, $indexType)];

$this->number = (null !== $col) ? (string) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 4; // 4 = StgNodesTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgNodes'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [shape\_id] column.

\*

\* @param int $v new value

\* @return $this|\StgNodes The current object (for fluent API support)

\*/

public function setShapeId($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->shape\_id !== $v) {

$this->shape\_id = $v;

$this->modifiedColumns[StgNodesTableMap::COL\_SHAPE\_ID] = true;

}

return $this;

} // setShapeId()

/\*\*

\* Set the value of [x] column.

\*

\* @param double $v new value

\* @return $this|\StgNodes The current object (for fluent API support)

\*/

public function setX($v)

{

if ($v !== null) {

$v = (double) $v;

}

if ($this->x !== $v) {

$this->x = $v;

$this->modifiedColumns[StgNodesTableMap::COL\_X] = true;

}

return $this;

} // setX()

/\*\*

\* Set the value of [y] column.

\*

\* @param double $v new value

\* @return $this|\StgNodes The current object (for fluent API support)

\*/

public function setY($v)

{

if ($v !== null) {

$v = (double) $v;

}

if ($this->y !== $v) {

$this->y = $v;

$this->modifiedColumns[StgNodesTableMap::COL\_Y] = true;

}

return $this;

} // setY()

/\*\*

\* Set the value of [number] column.

\*

\* @param string $v new value

\* @return $this|\StgNodes The current object (for fluent API support)

\*/

public function setNumber($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->number !== $v) {

$this->number = $v;

$this->modifiedColumns[StgNodesTableMap::COL\_NUMBER] = true;

}

return $this;

} // setNumber()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgNodesTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgNodesQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgNodes::setDeleted()

\* @see StgNodes::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNodesTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgNodesQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNodesTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgNodesTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgNodesTableMap::COL\_SHAPE\_ID)) {

$modifiedColumns[':p' . $index++] = 'SHAPE\_ID';

}

if ($this->isColumnModified(StgNodesTableMap::COL\_X)) {

$modifiedColumns[':p' . $index++] = 'X';

}

if ($this->isColumnModified(StgNodesTableMap::COL\_Y)) {

$modifiedColumns[':p' . $index++] = 'Y';

}

if ($this->isColumnModified(StgNodesTableMap::COL\_NUMBER)) {

$modifiedColumns[':p' . $index++] = 'NUMBER';

}

$sql = sprintf(

'INSERT INTO stg\_nodes (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'SHAPE\_ID':

$stmt->bindValue($identifier, $this->shape\_id, PDO::PARAM\_INT);

break;

case 'X':

$stmt->bindValue($identifier, $this->x, PDO::PARAM\_STR);

break;

case 'Y':

$stmt->bindValue($identifier, $this->y, PDO::PARAM\_STR);

break;

case 'NUMBER':

$stmt->bindValue($identifier, $this->number, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgNodesTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getShapeId();

break;

case 1:

return $this->getX();

break;

case 2:

return $this->getY();

break;

case 3:

return $this->getNumber();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgNodes'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgNodes'][$this->getPrimaryKey()] = true;

$keys = StgNodesTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getShapeId(),

$keys[1] => $this->getX(),

$keys[2] => $this->getY(),

$keys[3] => $this->getNumber(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgNodes

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgNodesTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgNodes

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setShapeId($value);

break;

case 1:

$this->setX($value);

break;

case 2:

$this->setY($value);

break;

case 3:

$this->setNumber($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgNodesTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setShapeId($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setX($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setY($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setNumber($arr[$keys[3]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgNodes The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgNodesTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgNodesTableMap::COL\_SHAPE\_ID)) {

$criteria->add(StgNodesTableMap::COL\_SHAPE\_ID, $this->shape\_id);

}

if ($this->isColumnModified(StgNodesTableMap::COL\_X)) {

$criteria->add(StgNodesTableMap::COL\_X, $this->x);

}

if ($this->isColumnModified(StgNodesTableMap::COL\_Y)) {

$criteria->add(StgNodesTableMap::COL\_Y, $this->y);

}

if ($this->isColumnModified(StgNodesTableMap::COL\_NUMBER)) {

$criteria->add(StgNodesTableMap::COL\_NUMBER, $this->number);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgNodes object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgNodes (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setShapeId($this->getShapeId());

$copyObj->setX($this->getX());

$copyObj->setY($this->getY());

$copyObj->setNumber($this->getNumber());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgNodes Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->shape\_id = null;

$this->x = null;

$this->y = null;

$this->number = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgNodesTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgNodesQuery.php

<?php

namespace Base;

use \StgNodes as ChildStgNodes;

use \StgNodesQuery as ChildStgNodesQuery;

use \Exception;

use Map\StgNodesTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_nodes' table.

\*

\*

\*

\* @method ChildStgNodesQuery orderByShapeId($order = Criteria::ASC) Order by the shape\_id column

\* @method ChildStgNodesQuery orderByX($order = Criteria::ASC) Order by the x column

\* @method ChildStgNodesQuery orderByY($order = Criteria::ASC) Order by the y column

\* @method ChildStgNodesQuery orderByNumber($order = Criteria::ASC) Order by the number column

\*

\* @method ChildStgNodesQuery groupByShapeId() Group by the shape\_id column

\* @method ChildStgNodesQuery groupByX() Group by the x column

\* @method ChildStgNodesQuery groupByY() Group by the y column

\* @method ChildStgNodesQuery groupByNumber() Group by the number column

\*

\* @method ChildStgNodesQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgNodesQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgNodesQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgNodes findOne(ConnectionInterface $con = null) Return the first ChildStgNodes matching the query

\* @method ChildStgNodes findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgNodes matching the query, or a new ChildStgNodes object populated from the query conditions when no match is found

\*

\* @method ChildStgNodes findOneByShapeId(int $shape\_id) Return the first ChildStgNodes filtered by the shape\_id column

\* @method ChildStgNodes findOneByX(double $x) Return the first ChildStgNodes filtered by the x column

\* @method ChildStgNodes findOneByY(double $y) Return the first ChildStgNodes filtered by the y column

\* @method ChildStgNodes findOneByNumber(string $number) Return the first ChildStgNodes filtered by the number column

\*

\* @method ChildStgNodes[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgNodes objects based on current ModelCriteria

\* @method ChildStgNodes[]|ObjectCollection findByShapeId(int $shape\_id) Return ChildStgNodes objects filtered by the shape\_id column

\* @method ChildStgNodes[]|ObjectCollection findByX(double $x) Return ChildStgNodes objects filtered by the x column

\* @method ChildStgNodes[]|ObjectCollection findByY(double $y) Return ChildStgNodes objects filtered by the y column

\* @method ChildStgNodes[]|ObjectCollection findByNumber(string $number) Return ChildStgNodes objects filtered by the number column

\* @method ChildStgNodes[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgNodesQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgNodesQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgNodes', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgNodesQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgNodesQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgNodesQuery) {

return $criteria;

}

$query = new ChildStgNodesQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgNodes|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgNodes object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgNodes object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgNodesQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgNodes object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgNodesQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgNodes object has no primary key');

}

/\*\*

\* Filter the query on the shape\_id column

\*

\* Example usage:

\* <code>

\* $query->filterByShapeId(1234); // WHERE shape\_id = 1234

\* $query->filterByShapeId(array(12, 34)); // WHERE shape\_id IN (12, 34)

\* $query->filterByShapeId(array('min' => 12)); // WHERE shape\_id > 12

\* </code>

\*

\* @param mixed $shapeId The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodesQuery The current query, for fluid interface

\*/

public function filterByShapeId($shapeId = null, $comparison = null)

{

if (is\_array($shapeId)) {

$useMinMax = false;

if (isset($shapeId['min'])) {

$this->addUsingAlias(StgNodesTableMap::COL\_SHAPE\_ID, $shapeId['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($shapeId['max'])) {

$this->addUsingAlias(StgNodesTableMap::COL\_SHAPE\_ID, $shapeId['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodesTableMap::COL\_SHAPE\_ID, $shapeId, $comparison);

}

/\*\*

\* Filter the query on the x column

\*

\* Example usage:

\* <code>

\* $query->filterByX(1234); // WHERE x = 1234

\* $query->filterByX(array(12, 34)); // WHERE x IN (12, 34)

\* $query->filterByX(array('min' => 12)); // WHERE x > 12

\* </code>

\*

\* @param mixed $x The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodesQuery The current query, for fluid interface

\*/

public function filterByX($x = null, $comparison = null)

{

if (is\_array($x)) {

$useMinMax = false;

if (isset($x['min'])) {

$this->addUsingAlias(StgNodesTableMap::COL\_X, $x['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($x['max'])) {

$this->addUsingAlias(StgNodesTableMap::COL\_X, $x['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodesTableMap::COL\_X, $x, $comparison);

}

/\*\*

\* Filter the query on the y column

\*

\* Example usage:

\* <code>

\* $query->filterByY(1234); // WHERE y = 1234

\* $query->filterByY(array(12, 34)); // WHERE y IN (12, 34)

\* $query->filterByY(array('min' => 12)); // WHERE y > 12

\* </code>

\*

\* @param mixed $y The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodesQuery The current query, for fluid interface

\*/

public function filterByY($y = null, $comparison = null)

{

if (is\_array($y)) {

$useMinMax = false;

if (isset($y['min'])) {

$this->addUsingAlias(StgNodesTableMap::COL\_Y, $y['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($y['max'])) {

$this->addUsingAlias(StgNodesTableMap::COL\_Y, $y['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodesTableMap::COL\_Y, $y, $comparison);

}

/\*\*

\* Filter the query on the number column

\*

\* Example usage:

\* <code>

\* $query->filterByNumber(1234); // WHERE number = 1234

\* $query->filterByNumber(array(12, 34)); // WHERE number IN (12, 34)

\* $query->filterByNumber(array('min' => 12)); // WHERE number > 12

\* </code>

\*

\* @param mixed $number The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgNodesQuery The current query, for fluid interface

\*/

public function filterByNumber($number = null, $comparison = null)

{

if (is\_array($number)) {

$useMinMax = false;

if (isset($number['min'])) {

$this->addUsingAlias(StgNodesTableMap::COL\_NUMBER, $number['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($number['max'])) {

$this->addUsingAlias(StgNodesTableMap::COL\_NUMBER, $number['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgNodesTableMap::COL\_NUMBER, $number, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgNodes $stgNodes Object to remove from the list of results

\*

\* @return $this|ChildStgNodesQuery The current query, for fluid interface

\*/

public function prune($stgNodes = null)

{

if ($stgNodes) {

throw new LogicException('StgNodes object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_nodes table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNodesTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgNodesTableMap::clearInstancePool();

StgNodesTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNodesTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgNodesTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgNodesTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgNodesTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgNodesQuery

#### StgOktatasiFerohelyekSzama.php

<?php

namespace Base;

use \StgOktatasiFerohelyekSzamaQuery as ChildStgOktatasiFerohelyekSzamaQuery;

use \Exception;

use \PDO;

use Map\StgOktatasiFerohelyekSzamaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgOktatasiFerohelyekSzama implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgOktatasiFerohelyekSzamaTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the nappali\_alt\_iskolai\_osztalyok\_szama field.

\* @var int

\*/

protected $nappali\_alt\_iskolai\_osztalyok\_szama;

/\*\*

\* The value for the nappali\_kozepiskolai\_osztalyok\_szama field.

\* @var int

\*/

protected $nappali\_kozepiskolai\_osztalyok\_szama;

/\*\*

\* The value for the nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama field.

\* @var int

\*/

protected $nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama;

/\*\*

\* The value for the nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben field.

\* @var int

\*/

protected $nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben;

/\*\*

\* The value for the nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben field.

\* @var int

\*/

protected $nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgOktatasiFerohelyekSzama object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgOktatasiFerohelyekSzama</code> instance. If

\* <code>obj</code> is an instance of <code>StgOktatasiFerohelyekSzama</code>, delegates to

\* <code>equals(StgOktatasiFerohelyekSzama)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgOktatasiFerohelyekSzama The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [nappali\_alt\_iskolai\_osztalyok\_szama] column value.

\*

\* @return int

\*/

public function getNappaliAltIskolaiOsztalyokSzama()

{

return $this->nappali\_alt\_iskolai\_osztalyok\_szama;

}

/\*\*

\* Get the [nappali\_kozepiskolai\_osztalyok\_szama] column value.

\*

\* @return int

\*/

public function getNappaliKozepiskolaiOsztalyokSzama()

{

return $this->nappali\_kozepiskolai\_osztalyok\_szama;

}

/\*\*

\* Get the [nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama] column value.

\*

\* @return int

\*/

public function getNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama()

{

return $this->nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama;

}

/\*\*

\* Get the [nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben] column value.

\*

\* @return int

\*/

public function getNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben()

{

return $this->nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben;

}

/\*\*

\* Get the [nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben] column value.

\*

\* @return int

\*/

public function getNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben()

{

return $this->nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgOktatasiFerohelyekSzamaTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgOktatasiFerohelyekSzamaTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgOktatasiFerohelyekSzamaTableMap::translateFieldName('NappaliAltIskolaiOsztalyokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_alt\_iskolai\_osztalyok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgOktatasiFerohelyekSzamaTableMap::translateFieldName('NappaliKozepiskolaiOsztalyokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_kozepiskolai\_osztalyok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgOktatasiFerohelyekSzamaTableMap::translateFieldName('NappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 5 + $startcol : StgOktatasiFerohelyekSzamaTableMap::translateFieldName('NappaliHallgatokSzamaFelsofokuAlapMesterkepzesben', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 6 + $startcol : StgOktatasiFerohelyekSzamaTableMap::translateFieldName('NemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 7; // 7 = StgOktatasiFerohelyekSzamaTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgOktatasiFerohelyekSzama'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgOktatasiFerohelyekSzama The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgOktatasiFerohelyekSzamaTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgOktatasiFerohelyekSzama The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgOktatasiFerohelyekSzamaTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [nappali\_alt\_iskolai\_osztalyok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgOktatasiFerohelyekSzama The current object (for fluent API support)

\*/

public function setNappaliAltIskolaiOsztalyokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_alt\_iskolai\_osztalyok\_szama !== $v) {

$this->nappali\_alt\_iskolai\_osztalyok\_szama = $v;

$this->modifiedColumns[StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA] = true;

}

return $this;

} // setNappaliAltIskolaiOsztalyokSzama()

/\*\*

\* Set the value of [nappali\_kozepiskolai\_osztalyok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgOktatasiFerohelyekSzama The current object (for fluent API support)

\*/

public function setNappaliKozepiskolaiOsztalyokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_kozepiskolai\_osztalyok\_szama !== $v) {

$this->nappali\_kozepiskolai\_osztalyok\_szama = $v;

$this->modifiedColumns[StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA] = true;

}

return $this;

} // setNappaliKozepiskolaiOsztalyokSzama()

/\*\*

\* Set the value of [nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgOktatasiFerohelyekSzama The current object (for fluent API support)

\*/

public function setNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama !== $v) {

$this->nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama = $v;

$this->modifiedColumns[StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA] = true;

}

return $this;

} // setNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama()

/\*\*

\* Set the value of [nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben] column.

\*

\* @param int $v new value

\* @return $this|\StgOktatasiFerohelyekSzama The current object (for fluent API support)

\*/

public function setNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben !== $v) {

$this->nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben = $v;

$this->modifiedColumns[StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN] = true;

}

return $this;

} // setNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben()

/\*\*

\* Set the value of [nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben] column.

\*

\* @param int $v new value

\* @return $this|\StgOktatasiFerohelyekSzama The current object (for fluent API support)

\*/

public function setNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben !== $v) {

$this->nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben = $v;

$this->modifiedColumns[StgOktatasiFerohelyekSzamaTableMap::COL\_NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN] = true;

}

return $this;

} // setNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgOktatasiFerohelyekSzamaTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgOktatasiFerohelyekSzamaQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgOktatasiFerohelyekSzama::setDeleted()

\* @see StgOktatasiFerohelyekSzama::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgOktatasiFerohelyekSzamaTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgOktatasiFerohelyekSzamaQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgOktatasiFerohelyekSzamaTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgOktatasiFerohelyekSzamaTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgOktatasiFerohelyekSzamaTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgOktatasiFerohelyekSzamaTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA';

}

if ($this->isColumnModified(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA';

}

if ($this->isColumnModified(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA';

}

if ($this->isColumnModified(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN';

}

if ($this->isColumnModified(StgOktatasiFerohelyekSzamaTableMap::COL\_NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN)) {

$modifiedColumns[':p' . $index++] = 'NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN';

}

$sql = sprintf(

'INSERT INTO stg\_oktatasi\_ferohelyek\_szama (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA':

$stmt->bindValue($identifier, $this->nappali\_alt\_iskolai\_osztalyok\_szama, PDO::PARAM\_INT);

break;

case 'NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA':

$stmt->bindValue($identifier, $this->nappali\_kozepiskolai\_osztalyok\_szama, PDO::PARAM\_INT);

break;

case 'NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA':

$stmt->bindValue($identifier, $this->nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama, PDO::PARAM\_INT);

break;

case 'NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN':

$stmt->bindValue($identifier, $this->nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben, PDO::PARAM\_INT);

break;

case 'NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN':

$stmt->bindValue($identifier, $this->nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgOktatasiFerohelyekSzamaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getNappaliAltIskolaiOsztalyokSzama();

break;

case 3:

return $this->getNappaliKozepiskolaiOsztalyokSzama();

break;

case 4:

return $this->getNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama();

break;

case 5:

return $this->getNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben();

break;

case 6:

return $this->getNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgOktatasiFerohelyekSzama'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgOktatasiFerohelyekSzama'][$this->getPrimaryKey()] = true;

$keys = StgOktatasiFerohelyekSzamaTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getNappaliAltIskolaiOsztalyokSzama(),

$keys[3] => $this->getNappaliKozepiskolaiOsztalyokSzama(),

$keys[4] => $this->getNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama(),

$keys[5] => $this->getNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben(),

$keys[6] => $this->getNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgOktatasiFerohelyekSzama

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgOktatasiFerohelyekSzamaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgOktatasiFerohelyekSzama

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setNappaliAltIskolaiOsztalyokSzama($value);

break;

case 3:

$this->setNappaliKozepiskolaiOsztalyokSzama($value);

break;

case 4:

$this->setNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama($value);

break;

case 5:

$this->setNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben($value);

break;

case 6:

$this->setNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgOktatasiFerohelyekSzamaTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setNappaliAltIskolaiOsztalyokSzama($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setNappaliKozepiskolaiOsztalyokSzama($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama($arr[$keys[4]]);

}

if (array\_key\_exists($keys[5], $arr)) {

$this->setNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben($arr[$keys[5]]);

}

if (array\_key\_exists($keys[6], $arr)) {

$this->setNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben($arr[$keys[6]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgOktatasiFerohelyekSzama The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgOktatasiFerohelyekSzamaTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgOktatasiFerohelyekSzamaTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgOktatasiFerohelyekSzamaTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgOktatasiFerohelyekSzamaTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgOktatasiFerohelyekSzamaTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA)) {

$criteria->add(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA, $this->nappali\_alt\_iskolai\_osztalyok\_szama);

}

if ($this->isColumnModified(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA)) {

$criteria->add(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA, $this->nappali\_kozepiskolai\_osztalyok\_szama);

}

if ($this->isColumnModified(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA)) {

$criteria->add(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA, $this->nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama);

}

if ($this->isColumnModified(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN)) {

$criteria->add(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN, $this->nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben);

}

if ($this->isColumnModified(StgOktatasiFerohelyekSzamaTableMap::COL\_NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN)) {

$criteria->add(StgOktatasiFerohelyekSzamaTableMap::COL\_NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN, $this->nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgOktatasiFerohelyekSzama object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgOktatasiFerohelyekSzama (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setNappaliAltIskolaiOsztalyokSzama($this->getNappaliAltIskolaiOsztalyokSzama());

$copyObj->setNappaliKozepiskolaiOsztalyokSzama($this->getNappaliKozepiskolaiOsztalyokSzama());

$copyObj->setNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama($this->getNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama());

$copyObj->setNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben($this->getNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben());

$copyObj->setNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben($this->getNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgOktatasiFerohelyekSzama Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->nappali\_alt\_iskolai\_osztalyok\_szama = null;

$this->nappali\_kozepiskolai\_osztalyok\_szama = null;

$this->nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama = null;

$this->nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben = null;

$this->nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgOktatasiFerohelyekSzamaTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgOktatasiFerohelyekSzamaQuery.php

<?php

namespace Base;

use \StgOktatasiFerohelyekSzama as ChildStgOktatasiFerohelyekSzama;

use \StgOktatasiFerohelyekSzamaQuery as ChildStgOktatasiFerohelyekSzamaQuery;

use \Exception;

use Map\StgOktatasiFerohelyekSzamaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_oktatasi\_ferohelyek\_szama' table.

\*

\*

\*

\* @method ChildStgOktatasiFerohelyekSzamaQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgOktatasiFerohelyekSzamaQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKod column

\* @method ChildStgOktatasiFerohelyekSzamaQuery orderByNappaliAltIskolaiOsztalyokSzama($order = Criteria::ASC) Order by the nappali\_alt\_iskolai\_osztalyok\_szama column

\* @method ChildStgOktatasiFerohelyekSzamaQuery orderByNappaliKozepiskolaiOsztalyokSzama($order = Criteria::ASC) Order by the nappali\_kozepiskolai\_osztalyok\_szama column

\* @method ChildStgOktatasiFerohelyekSzamaQuery orderByNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama($order = Criteria::ASC) Order by the nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama column

\* @method ChildStgOktatasiFerohelyekSzamaQuery orderByNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben($order = Criteria::ASC) Order by the nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben column

\* @method ChildStgOktatasiFerohelyekSzamaQuery orderByNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben($order = Criteria::ASC) Order by the nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben column

\*

\* @method ChildStgOktatasiFerohelyekSzamaQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgOktatasiFerohelyekSzamaQuery groupByTelepulesKshkod() Group by the telepules\_KSHKod column

\* @method ChildStgOktatasiFerohelyekSzamaQuery groupByNappaliAltIskolaiOsztalyokSzama() Group by the nappali\_alt\_iskolai\_osztalyok\_szama column

\* @method ChildStgOktatasiFerohelyekSzamaQuery groupByNappaliKozepiskolaiOsztalyokSzama() Group by the nappali\_kozepiskolai\_osztalyok\_szama column

\* @method ChildStgOktatasiFerohelyekSzamaQuery groupByNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama() Group by the nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama column

\* @method ChildStgOktatasiFerohelyekSzamaQuery groupByNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben() Group by the nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben column

\* @method ChildStgOktatasiFerohelyekSzamaQuery groupByNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben() Group by the nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben column

\*

\* @method ChildStgOktatasiFerohelyekSzamaQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgOktatasiFerohelyekSzamaQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgOktatasiFerohelyekSzamaQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgOktatasiFerohelyekSzama findOne(ConnectionInterface $con = null) Return the first ChildStgOktatasiFerohelyekSzama matching the query

\* @method ChildStgOktatasiFerohelyekSzama findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgOktatasiFerohelyekSzama matching the query, or a new ChildStgOktatasiFerohelyekSzama object populated from the query conditions when no match is found

\*

\* @method ChildStgOktatasiFerohelyekSzama findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgOktatasiFerohelyekSzama filtered by the telepules\_nev column

\* @method ChildStgOktatasiFerohelyekSzama findOneByTelepulesKshkod(string $telepules\_KSHKod) Return the first ChildStgOktatasiFerohelyekSzama filtered by the telepules\_KSHKod column

\* @method ChildStgOktatasiFerohelyekSzama findOneByNappaliAltIskolaiOsztalyokSzama(int $nappali\_alt\_iskolai\_osztalyok\_szama) Return the first ChildStgOktatasiFerohelyekSzama filtered by the nappali\_alt\_iskolai\_osztalyok\_szama column

\* @method ChildStgOktatasiFerohelyekSzama findOneByNappaliKozepiskolaiOsztalyokSzama(int $nappali\_kozepiskolai\_osztalyok\_szama) Return the first ChildStgOktatasiFerohelyekSzama filtered by the nappali\_kozepiskolai\_osztalyok\_szama column

\* @method ChildStgOktatasiFerohelyekSzama findOneByNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama(int $nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama) Return the first ChildStgOktatasiFerohelyekSzama filtered by the nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama column

\* @method ChildStgOktatasiFerohelyekSzama findOneByNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben(int $nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben) Return the first ChildStgOktatasiFerohelyekSzama filtered by the nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben column

\* @method ChildStgOktatasiFerohelyekSzama findOneByNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben(int $nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben) Return the first ChildStgOktatasiFerohelyekSzama filtered by the nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben column

\*

\* @method ChildStgOktatasiFerohelyekSzama[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgOktatasiFerohelyekSzama objects based on current ModelCriteria

\* @method ChildStgOktatasiFerohelyekSzama[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgOktatasiFerohelyekSzama objects filtered by the telepules\_nev column

\* @method ChildStgOktatasiFerohelyekSzama[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKod) Return ChildStgOktatasiFerohelyekSzama objects filtered by the telepules\_KSHKod column

\* @method ChildStgOktatasiFerohelyekSzama[]|ObjectCollection findByNappaliAltIskolaiOsztalyokSzama(int $nappali\_alt\_iskolai\_osztalyok\_szama) Return ChildStgOktatasiFerohelyekSzama objects filtered by the nappali\_alt\_iskolai\_osztalyok\_szama column

\* @method ChildStgOktatasiFerohelyekSzama[]|ObjectCollection findByNappaliKozepiskolaiOsztalyokSzama(int $nappali\_kozepiskolai\_osztalyok\_szama) Return ChildStgOktatasiFerohelyekSzama objects filtered by the nappali\_kozepiskolai\_osztalyok\_szama column

\* @method ChildStgOktatasiFerohelyekSzama[]|ObjectCollection findByNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama(int $nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama) Return ChildStgOktatasiFerohelyekSzama objects filtered by the nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama column

\* @method ChildStgOktatasiFerohelyekSzama[]|ObjectCollection findByNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben(int $nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben) Return ChildStgOktatasiFerohelyekSzama objects filtered by the nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben column

\* @method ChildStgOktatasiFerohelyekSzama[]|ObjectCollection findByNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben(int $nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben) Return ChildStgOktatasiFerohelyekSzama objects filtered by the nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben column

\* @method ChildStgOktatasiFerohelyekSzama[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgOktatasiFerohelyekSzamaQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgOktatasiFerohelyekSzamaQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgOktatasiFerohelyekSzama', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgOktatasiFerohelyekSzamaQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgOktatasiFerohelyekSzamaQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgOktatasiFerohelyekSzamaQuery) {

return $criteria;

}

$query = new ChildStgOktatasiFerohelyekSzamaQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgOktatasiFerohelyekSzama|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgOktatasiFerohelyekSzama object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgOktatasiFerohelyekSzama object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgOktatasiFerohelyekSzamaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgOktatasiFerohelyekSzama object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgOktatasiFerohelyekSzamaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgOktatasiFerohelyekSzama object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgOktatasiFerohelyekSzamaQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgOktatasiFerohelyekSzamaTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKod column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKod = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKod LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgOktatasiFerohelyekSzamaQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgOktatasiFerohelyekSzamaTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the nappali\_alt\_iskolai\_osztalyok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliAltIskolaiOsztalyokSzama(1234); // WHERE nappali\_alt\_iskolai\_osztalyok\_szama = 1234

\* $query->filterByNappaliAltIskolaiOsztalyokSzama(array(12, 34)); // WHERE nappali\_alt\_iskolai\_osztalyok\_szama IN (12, 34)

\* $query->filterByNappaliAltIskolaiOsztalyokSzama(array('min' => 12)); // WHERE nappali\_alt\_iskolai\_osztalyok\_szama > 12

\* </code>

\*

\* @param mixed $nappaliAltIskolaiOsztalyokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgOktatasiFerohelyekSzamaQuery The current query, for fluid interface

\*/

public function filterByNappaliAltIskolaiOsztalyokSzama($nappaliAltIskolaiOsztalyokSzama = null, $comparison = null)

{

if (is\_array($nappaliAltIskolaiOsztalyokSzama)) {

$useMinMax = false;

if (isset($nappaliAltIskolaiOsztalyokSzama['min'])) {

$this->addUsingAlias(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA, $nappaliAltIskolaiOsztalyokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliAltIskolaiOsztalyokSzama['max'])) {

$this->addUsingAlias(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA, $nappaliAltIskolaiOsztalyokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA, $nappaliAltIskolaiOsztalyokSzama, $comparison);

}

/\*\*

\* Filter the query on the nappali\_kozepiskolai\_osztalyok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliKozepiskolaiOsztalyokSzama(1234); // WHERE nappali\_kozepiskolai\_osztalyok\_szama = 1234

\* $query->filterByNappaliKozepiskolaiOsztalyokSzama(array(12, 34)); // WHERE nappali\_kozepiskolai\_osztalyok\_szama IN (12, 34)

\* $query->filterByNappaliKozepiskolaiOsztalyokSzama(array('min' => 12)); // WHERE nappali\_kozepiskolai\_osztalyok\_szama > 12

\* </code>

\*

\* @param mixed $nappaliKozepiskolaiOsztalyokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgOktatasiFerohelyekSzamaQuery The current query, for fluid interface

\*/

public function filterByNappaliKozepiskolaiOsztalyokSzama($nappaliKozepiskolaiOsztalyokSzama = null, $comparison = null)

{

if (is\_array($nappaliKozepiskolaiOsztalyokSzama)) {

$useMinMax = false;

if (isset($nappaliKozepiskolaiOsztalyokSzama['min'])) {

$this->addUsingAlias(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA, $nappaliKozepiskolaiOsztalyokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliKozepiskolaiOsztalyokSzama['max'])) {

$this->addUsingAlias(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA, $nappaliKozepiskolaiOsztalyokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA, $nappaliKozepiskolaiOsztalyokSzama, $comparison);

}

/\*\*

\* Filter the query on the nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama(1234); // WHERE nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama = 1234

\* $query->filterByNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama(array(12, 34)); // WHERE nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama IN (12, 34)

\* $query->filterByNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama(array('min' => 12)); // WHERE nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama > 12

\* </code>

\*

\* @param mixed $nappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgOktatasiFerohelyekSzamaQuery The current query, for fluid interface

\*/

public function filterByNappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama($nappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama = null, $comparison = null)

{

if (is\_array($nappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama)) {

$useMinMax = false;

if (isset($nappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama['min'])) {

$this->addUsingAlias(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA, $nappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama['max'])) {

$this->addUsingAlias(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA, $nappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA, $nappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama, $comparison);

}

/\*\*

\* Filter the query on the nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben(1234); // WHERE nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben = 1234

\* $query->filterByNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben(array(12, 34)); // WHERE nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben IN (12, 34)

\* $query->filterByNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben(array('min' => 12)); // WHERE nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben > 12

\* </code>

\*

\* @param mixed $nappaliHallgatokSzamaFelsofokuAlapMesterkepzesben The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgOktatasiFerohelyekSzamaQuery The current query, for fluid interface

\*/

public function filterByNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben($nappaliHallgatokSzamaFelsofokuAlapMesterkepzesben = null, $comparison = null)

{

if (is\_array($nappaliHallgatokSzamaFelsofokuAlapMesterkepzesben)) {

$useMinMax = false;

if (isset($nappaliHallgatokSzamaFelsofokuAlapMesterkepzesben['min'])) {

$this->addUsingAlias(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN, $nappaliHallgatokSzamaFelsofokuAlapMesterkepzesben['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliHallgatokSzamaFelsofokuAlapMesterkepzesben['max'])) {

$this->addUsingAlias(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN, $nappaliHallgatokSzamaFelsofokuAlapMesterkepzesben['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN, $nappaliHallgatokSzamaFelsofokuAlapMesterkepzesben, $comparison);

}

/\*\*

\* Filter the query on the nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben column

\*

\* Example usage:

\* <code>

\* $query->filterByNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben(1234); // WHERE nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben = 1234

\* $query->filterByNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben(array(12, 34)); // WHERE nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben IN (12, 34)

\* $query->filterByNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben(array('min' => 12)); // WHERE nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben > 12

\* </code>

\*

\* @param mixed $nemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgOktatasiFerohelyekSzamaQuery The current query, for fluid interface

\*/

public function filterByNemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben($nemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben = null, $comparison = null)

{

if (is\_array($nemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben)) {

$useMinMax = false;

if (isset($nemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben['min'])) {

$this->addUsingAlias(StgOktatasiFerohelyekSzamaTableMap::COL\_NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN, $nemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben['max'])) {

$this->addUsingAlias(StgOktatasiFerohelyekSzamaTableMap::COL\_NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN, $nemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgOktatasiFerohelyekSzamaTableMap::COL\_NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN, $nemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgOktatasiFerohelyekSzama $stgOktatasiFerohelyekSzama Object to remove from the list of results

\*

\* @return $this|ChildStgOktatasiFerohelyekSzamaQuery The current query, for fluid interface

\*/

public function prune($stgOktatasiFerohelyekSzama = null)

{

if ($stgOktatasiFerohelyekSzama) {

throw new LogicException('StgOktatasiFerohelyekSzama object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_oktatasi\_ferohelyek\_szama table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgOktatasiFerohelyekSzamaTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgOktatasiFerohelyekSzamaTableMap::clearInstancePool();

StgOktatasiFerohelyekSzamaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgOktatasiFerohelyekSzamaTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgOktatasiFerohelyekSzamaTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgOktatasiFerohelyekSzamaTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgOktatasiFerohelyekSzamaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgOktatasiFerohelyekSzamaQuery

#### StgSzabadIskolavalasztasValtozo.php

<?php

namespace Base;

use \StgSzabadIskolavalasztasValtozoQuery as ChildStgSzabadIskolavalasztasValtozoQuery;

use \Exception;

use \PDO;

use Map\StgSzabadIskolavalasztasValtozoTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgSzabadIskolavalasztasValtozo implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgSzabadIskolavalasztasValtozoTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the allando\_nepessegbol\_6\_13\_evesek\_szama field.

\* @var int

\*/

protected $allando\_nepessegbol\_6\_13\_evesek\_szama;

/\*\*

\* The value for the allando\_nepessegbol\_14\_evesek\_szama field.

\* @var int

\*/

protected $allando\_nepessegbol\_14\_evesek\_szama;

/\*\*

\* The value for the allando\_nepessegbol\_15\_17\_evesek\_szama field.

\* @var int

\*/

protected $allando\_nepessegbol\_15\_17\_evesek\_szama;

/\*\*

\* The value for the nappali\_altalanos\_iskolai\_tanulok\_szama field.

\* @var int

\*/

protected $nappali\_altalanos\_iskolai\_tanulok\_szama;

/\*\*

\* The value for the nappali\_kozepiskolai\_tanulok\_szama field.

\* @var int

\*/

protected $nappali\_kozepiskolai\_tanulok\_szama;

/\*\*

\* The value for the nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama field.

\* @var int

\*/

protected $nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama;

/\*\*

\* The value for the mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama field.

\* @var int

\*/

protected $mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama;

/\*\*

\* The value for the mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama field.

\* @var int

\*/

protected $mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgSzabadIskolavalasztasValtozo object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgSzabadIskolavalasztasValtozo</code> instance. If

\* <code>obj</code> is an instance of <code>StgSzabadIskolavalasztasValtozo</code>, delegates to

\* <code>equals(StgSzabadIskolavalasztasValtozo)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgSzabadIskolavalasztasValtozo The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [allando\_nepessegbol\_6\_13\_evesek\_szama] column value.

\*

\* @return int

\*/

public function getAllandoNepessegbol613EvesekSzama()

{

return $this->allando\_nepessegbol\_6\_13\_evesek\_szama;

}

/\*\*

\* Get the [allando\_nepessegbol\_14\_evesek\_szama] column value.

\*

\* @return int

\*/

public function getAllandoNepessegbol14EvesekSzama()

{

return $this->allando\_nepessegbol\_14\_evesek\_szama;

}

/\*\*

\* Get the [allando\_nepessegbol\_15\_17\_evesek\_szama] column value.

\*

\* @return int

\*/

public function getAllandoNepessegbol1517EvesekSzama()

{

return $this->allando\_nepessegbol\_15\_17\_evesek\_szama;

}

/\*\*

\* Get the [nappali\_altalanos\_iskolai\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getNappaliAltalanosIskolaiTanulokSzama()

{

return $this->nappali\_altalanos\_iskolai\_tanulok\_szama;

}

/\*\*

\* Get the [nappali\_kozepiskolai\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getNappaliKozepiskolaiTanulokSzama()

{

return $this->nappali\_kozepiskolai\_tanulok\_szama;

}

/\*\*

\* Get the [nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama()

{

return $this->nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama;

}

/\*\*

\* Get the [mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama()

{

return $this->mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama;

}

/\*\*

\* Get the [mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama()

{

return $this->mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgSzabadIskolavalasztasValtozoTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgSzabadIskolavalasztasValtozoTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgSzabadIskolavalasztasValtozoTableMap::translateFieldName('AllandoNepessegbol613EvesekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->allando\_nepessegbol\_6\_13\_evesek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgSzabadIskolavalasztasValtozoTableMap::translateFieldName('AllandoNepessegbol14EvesekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->allando\_nepessegbol\_14\_evesek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgSzabadIskolavalasztasValtozoTableMap::translateFieldName('AllandoNepessegbol1517EvesekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->allando\_nepessegbol\_15\_17\_evesek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 5 + $startcol : StgSzabadIskolavalasztasValtozoTableMap::translateFieldName('NappaliAltalanosIskolaiTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_altalanos\_iskolai\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 6 + $startcol : StgSzabadIskolavalasztasValtozoTableMap::translateFieldName('NappaliKozepiskolaiTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_kozepiskolai\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 7 + $startcol : StgSzabadIskolavalasztasValtozoTableMap::translateFieldName('NappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 8 + $startcol : StgSzabadIskolavalasztasValtozoTableMap::translateFieldName('MasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 9 + $startcol : StgSzabadIskolavalasztasValtozoTableMap::translateFieldName('MasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 10; // 10 = StgSzabadIskolavalasztasValtozoTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgSzabadIskolavalasztasValtozo'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgSzabadIskolavalasztasValtozo The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgSzabadIskolavalasztasValtozoTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgSzabadIskolavalasztasValtozo The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgSzabadIskolavalasztasValtozoTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [allando\_nepessegbol\_6\_13\_evesek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgSzabadIskolavalasztasValtozo The current object (for fluent API support)

\*/

public function setAllandoNepessegbol613EvesekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->allando\_nepessegbol\_6\_13\_evesek\_szama !== $v) {

$this->allando\_nepessegbol\_6\_13\_evesek\_szama = $v;

$this->modifiedColumns[StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA] = true;

}

return $this;

} // setAllandoNepessegbol613EvesekSzama()

/\*\*

\* Set the value of [allando\_nepessegbol\_14\_evesek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgSzabadIskolavalasztasValtozo The current object (for fluent API support)

\*/

public function setAllandoNepessegbol14EvesekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->allando\_nepessegbol\_14\_evesek\_szama !== $v) {

$this->allando\_nepessegbol\_14\_evesek\_szama = $v;

$this->modifiedColumns[StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA] = true;

}

return $this;

} // setAllandoNepessegbol14EvesekSzama()

/\*\*

\* Set the value of [allando\_nepessegbol\_15\_17\_evesek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgSzabadIskolavalasztasValtozo The current object (for fluent API support)

\*/

public function setAllandoNepessegbol1517EvesekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->allando\_nepessegbol\_15\_17\_evesek\_szama !== $v) {

$this->allando\_nepessegbol\_15\_17\_evesek\_szama = $v;

$this->modifiedColumns[StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA] = true;

}

return $this;

} // setAllandoNepessegbol1517EvesekSzama()

/\*\*

\* Set the value of [nappali\_altalanos\_iskolai\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgSzabadIskolavalasztasValtozo The current object (for fluent API support)

\*/

public function setNappaliAltalanosIskolaiTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_altalanos\_iskolai\_tanulok\_szama !== $v) {

$this->nappali\_altalanos\_iskolai\_tanulok\_szama = $v;

$this->modifiedColumns[StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA] = true;

}

return $this;

} // setNappaliAltalanosIskolaiTanulokSzama()

/\*\*

\* Set the value of [nappali\_kozepiskolai\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgSzabadIskolavalasztasValtozo The current object (for fluent API support)

\*/

public function setNappaliKozepiskolaiTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_kozepiskolai\_tanulok\_szama !== $v) {

$this->nappali\_kozepiskolai\_tanulok\_szama = $v;

$this->modifiedColumns[StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA] = true;

}

return $this;

} // setNappaliKozepiskolaiTanulokSzama()

/\*\*

\* Set the value of [nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgSzabadIskolavalasztasValtozo The current object (for fluent API support)

\*/

public function setNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama !== $v) {

$this->nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama = $v;

$this->modifiedColumns[StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA] = true;

}

return $this;

} // setNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama()

/\*\*

\* Set the value of [mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgSzabadIskolavalasztasValtozo The current object (for fluent API support)

\*/

public function setMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama !== $v) {

$this->mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama = $v;

$this->modifiedColumns[StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA] = true;

}

return $this;

} // setMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama()

/\*\*

\* Set the value of [mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgSzabadIskolavalasztasValtozo The current object (for fluent API support)

\*/

public function setMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama !== $v) {

$this->mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama = $v;

$this->modifiedColumns[StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA] = true;

}

return $this;

} // setMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgSzabadIskolavalasztasValtozoTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgSzabadIskolavalasztasValtozoQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgSzabadIskolavalasztasValtozo::setDeleted()

\* @see StgSzabadIskolavalasztasValtozo::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzabadIskolavalasztasValtozoTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgSzabadIskolavalasztasValtozoQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzabadIskolavalasztasValtozoTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgSzabadIskolavalasztasValtozoTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA';

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA';

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA';

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA';

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA';

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA';

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA';

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_szabad\_iskolavalasztas\_valtozo (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA':

$stmt->bindValue($identifier, $this->allando\_nepessegbol\_6\_13\_evesek\_szama, PDO::PARAM\_INT);

break;

case 'ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA':

$stmt->bindValue($identifier, $this->allando\_nepessegbol\_14\_evesek\_szama, PDO::PARAM\_INT);

break;

case 'ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA':

$stmt->bindValue($identifier, $this->allando\_nepessegbol\_15\_17\_evesek\_szama, PDO::PARAM\_INT);

break;

case 'NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->nappali\_altalanos\_iskolai\_tanulok\_szama, PDO::PARAM\_INT);

break;

case 'NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->nappali\_kozepiskolai\_tanulok\_szama, PDO::PARAM\_INT);

break;

case 'NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama, PDO::PARAM\_INT);

break;

case 'MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama, PDO::PARAM\_INT);

break;

case 'MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgSzabadIskolavalasztasValtozoTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getAllandoNepessegbol613EvesekSzama();

break;

case 3:

return $this->getAllandoNepessegbol14EvesekSzama();

break;

case 4:

return $this->getAllandoNepessegbol1517EvesekSzama();

break;

case 5:

return $this->getNappaliAltalanosIskolaiTanulokSzama();

break;

case 6:

return $this->getNappaliKozepiskolaiTanulokSzama();

break;

case 7:

return $this->getNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama();

break;

case 8:

return $this->getMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama();

break;

case 9:

return $this->getMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgSzabadIskolavalasztasValtozo'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgSzabadIskolavalasztasValtozo'][$this->getPrimaryKey()] = true;

$keys = StgSzabadIskolavalasztasValtozoTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getAllandoNepessegbol613EvesekSzama(),

$keys[3] => $this->getAllandoNepessegbol14EvesekSzama(),

$keys[4] => $this->getAllandoNepessegbol1517EvesekSzama(),

$keys[5] => $this->getNappaliAltalanosIskolaiTanulokSzama(),

$keys[6] => $this->getNappaliKozepiskolaiTanulokSzama(),

$keys[7] => $this->getNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama(),

$keys[8] => $this->getMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama(),

$keys[9] => $this->getMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgSzabadIskolavalasztasValtozo

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgSzabadIskolavalasztasValtozoTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgSzabadIskolavalasztasValtozo

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setAllandoNepessegbol613EvesekSzama($value);

break;

case 3:

$this->setAllandoNepessegbol14EvesekSzama($value);

break;

case 4:

$this->setAllandoNepessegbol1517EvesekSzama($value);

break;

case 5:

$this->setNappaliAltalanosIskolaiTanulokSzama($value);

break;

case 6:

$this->setNappaliKozepiskolaiTanulokSzama($value);

break;

case 7:

$this->setNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama($value);

break;

case 8:

$this->setMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama($value);

break;

case 9:

$this->setMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgSzabadIskolavalasztasValtozoTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setAllandoNepessegbol613EvesekSzama($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setAllandoNepessegbol14EvesekSzama($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setAllandoNepessegbol1517EvesekSzama($arr[$keys[4]]);

}

if (array\_key\_exists($keys[5], $arr)) {

$this->setNappaliAltalanosIskolaiTanulokSzama($arr[$keys[5]]);

}

if (array\_key\_exists($keys[6], $arr)) {

$this->setNappaliKozepiskolaiTanulokSzama($arr[$keys[6]]);

}

if (array\_key\_exists($keys[7], $arr)) {

$this->setNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama($arr[$keys[7]]);

}

if (array\_key\_exists($keys[8], $arr)) {

$this->setMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama($arr[$keys[8]]);

}

if (array\_key\_exists($keys[9], $arr)) {

$this->setMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama($arr[$keys[9]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgSzabadIskolavalasztasValtozo The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgSzabadIskolavalasztasValtozoTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgSzabadIskolavalasztasValtozoTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgSzabadIskolavalasztasValtozoTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA)) {

$criteria->add(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA, $this->allando\_nepessegbol\_6\_13\_evesek\_szama);

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA)) {

$criteria->add(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA, $this->allando\_nepessegbol\_14\_evesek\_szama);

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA)) {

$criteria->add(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA, $this->allando\_nepessegbol\_15\_17\_evesek\_szama);

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA)) {

$criteria->add(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA, $this->nappali\_altalanos\_iskolai\_tanulok\_szama);

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA)) {

$criteria->add(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $this->nappali\_kozepiskolai\_tanulok\_szama);

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA)) {

$criteria->add(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, $this->nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama);

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA)) {

$criteria->add(StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA, $this->mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama);

}

if ($this->isColumnModified(StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA)) {

$criteria->add(StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $this->mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgSzabadIskolavalasztasValtozo object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgSzabadIskolavalasztasValtozo (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setAllandoNepessegbol613EvesekSzama($this->getAllandoNepessegbol613EvesekSzama());

$copyObj->setAllandoNepessegbol14EvesekSzama($this->getAllandoNepessegbol14EvesekSzama());

$copyObj->setAllandoNepessegbol1517EvesekSzama($this->getAllandoNepessegbol1517EvesekSzama());

$copyObj->setNappaliAltalanosIskolaiTanulokSzama($this->getNappaliAltalanosIskolaiTanulokSzama());

$copyObj->setNappaliKozepiskolaiTanulokSzama($this->getNappaliKozepiskolaiTanulokSzama());

$copyObj->setNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama($this->getNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama());

$copyObj->setMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama($this->getMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama());

$copyObj->setMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama($this->getMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgSzabadIskolavalasztasValtozo Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->allando\_nepessegbol\_6\_13\_evesek\_szama = null;

$this->allando\_nepessegbol\_14\_evesek\_szama = null;

$this->allando\_nepessegbol\_15\_17\_evesek\_szama = null;

$this->nappali\_altalanos\_iskolai\_tanulok\_szama = null;

$this->nappali\_kozepiskolai\_tanulok\_szama = null;

$this->nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama = null;

$this->mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama = null;

$this->mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgSzabadIskolavalasztasValtozoTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgSzabadIskolavalasztasValtozoQuery.php

<?php

namespace Base;

use \StgSzabadIskolavalasztasValtozo as ChildStgSzabadIskolavalasztasValtozo;

use \StgSzabadIskolavalasztasValtozoQuery as ChildStgSzabadIskolavalasztasValtozoQuery;

use \Exception;

use Map\StgSzabadIskolavalasztasValtozoTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_szabad\_iskolavalasztas\_valtozo' table.

\*

\*

\*

\* @method ChildStgSzabadIskolavalasztasValtozoQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery orderByAllandoNepessegbol613EvesekSzama($order = Criteria::ASC) Order by the allando\_nepessegbol\_6\_13\_evesek\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery orderByAllandoNepessegbol14EvesekSzama($order = Criteria::ASC) Order by the allando\_nepessegbol\_14\_evesek\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery orderByAllandoNepessegbol1517EvesekSzama($order = Criteria::ASC) Order by the allando\_nepessegbol\_15\_17\_evesek\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery orderByNappaliAltalanosIskolaiTanulokSzama($order = Criteria::ASC) Order by the nappali\_altalanos\_iskolai\_tanulok\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery orderByNappaliKozepiskolaiTanulokSzama($order = Criteria::ASC) Order by the nappali\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery orderByNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama($order = Criteria::ASC) Order by the nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery orderByMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama($order = Criteria::ASC) Order by the mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery orderByMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama($order = Criteria::ASC) Order by the mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama column

\*

\* @method ChildStgSzabadIskolavalasztasValtozoQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery groupByAllandoNepessegbol613EvesekSzama() Group by the allando\_nepessegbol\_6\_13\_evesek\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery groupByAllandoNepessegbol14EvesekSzama() Group by the allando\_nepessegbol\_14\_evesek\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery groupByAllandoNepessegbol1517EvesekSzama() Group by the allando\_nepessegbol\_15\_17\_evesek\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery groupByNappaliAltalanosIskolaiTanulokSzama() Group by the nappali\_altalanos\_iskolai\_tanulok\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery groupByNappaliKozepiskolaiTanulokSzama() Group by the nappali\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery groupByNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama() Group by the nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery groupByMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama() Group by the mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozoQuery groupByMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama() Group by the mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama column

\*

\* @method ChildStgSzabadIskolavalasztasValtozoQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgSzabadIskolavalasztasValtozoQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgSzabadIskolavalasztasValtozoQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgSzabadIskolavalasztasValtozo findOne(ConnectionInterface $con = null) Return the first ChildStgSzabadIskolavalasztasValtozo matching the query

\* @method ChildStgSzabadIskolavalasztasValtozo findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgSzabadIskolavalasztasValtozo matching the query, or a new ChildStgSzabadIskolavalasztasValtozo object populated from the query conditions when no match is found

\*

\* @method ChildStgSzabadIskolavalasztasValtozo findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgSzabadIskolavalasztasValtozo filtered by the telepules\_nev column

\* @method ChildStgSzabadIskolavalasztasValtozo findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgSzabadIskolavalasztasValtozo filtered by the telepules\_KSHKOD column

\* @method ChildStgSzabadIskolavalasztasValtozo findOneByAllandoNepessegbol613EvesekSzama(int $allando\_nepessegbol\_6\_13\_evesek\_szama) Return the first ChildStgSzabadIskolavalasztasValtozo filtered by the allando\_nepessegbol\_6\_13\_evesek\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozo findOneByAllandoNepessegbol14EvesekSzama(int $allando\_nepessegbol\_14\_evesek\_szama) Return the first ChildStgSzabadIskolavalasztasValtozo filtered by the allando\_nepessegbol\_14\_evesek\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozo findOneByAllandoNepessegbol1517EvesekSzama(int $allando\_nepessegbol\_15\_17\_evesek\_szama) Return the first ChildStgSzabadIskolavalasztasValtozo filtered by the allando\_nepessegbol\_15\_17\_evesek\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozo findOneByNappaliAltalanosIskolaiTanulokSzama(int $nappali\_altalanos\_iskolai\_tanulok\_szama) Return the first ChildStgSzabadIskolavalasztasValtozo filtered by the nappali\_altalanos\_iskolai\_tanulok\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozo findOneByNappaliKozepiskolaiTanulokSzama(int $nappali\_kozepiskolai\_tanulok\_szama) Return the first ChildStgSzabadIskolavalasztasValtozo filtered by the nappali\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozo findOneByNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama(int $nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama) Return the first ChildStgSzabadIskolavalasztasValtozo filtered by the nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozo findOneByMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama(int $mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama) Return the first ChildStgSzabadIskolavalasztasValtozo filtered by the mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozo findOneByMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama(int $mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama) Return the first ChildStgSzabadIskolavalasztasValtozo filtered by the mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama column

\*

\* @method ChildStgSzabadIskolavalasztasValtozo[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgSzabadIskolavalasztasValtozo objects based on current ModelCriteria

\* @method ChildStgSzabadIskolavalasztasValtozo[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgSzabadIskolavalasztasValtozo objects filtered by the telepules\_nev column

\* @method ChildStgSzabadIskolavalasztasValtozo[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgSzabadIskolavalasztasValtozo objects filtered by the telepules\_KSHKOD column

\* @method ChildStgSzabadIskolavalasztasValtozo[]|ObjectCollection findByAllandoNepessegbol613EvesekSzama(int $allando\_nepessegbol\_6\_13\_evesek\_szama) Return ChildStgSzabadIskolavalasztasValtozo objects filtered by the allando\_nepessegbol\_6\_13\_evesek\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozo[]|ObjectCollection findByAllandoNepessegbol14EvesekSzama(int $allando\_nepessegbol\_14\_evesek\_szama) Return ChildStgSzabadIskolavalasztasValtozo objects filtered by the allando\_nepessegbol\_14\_evesek\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozo[]|ObjectCollection findByAllandoNepessegbol1517EvesekSzama(int $allando\_nepessegbol\_15\_17\_evesek\_szama) Return ChildStgSzabadIskolavalasztasValtozo objects filtered by the allando\_nepessegbol\_15\_17\_evesek\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozo[]|ObjectCollection findByNappaliAltalanosIskolaiTanulokSzama(int $nappali\_altalanos\_iskolai\_tanulok\_szama) Return ChildStgSzabadIskolavalasztasValtozo objects filtered by the nappali\_altalanos\_iskolai\_tanulok\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozo[]|ObjectCollection findByNappaliKozepiskolaiTanulokSzama(int $nappali\_kozepiskolai\_tanulok\_szama) Return ChildStgSzabadIskolavalasztasValtozo objects filtered by the nappali\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozo[]|ObjectCollection findByNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama(int $nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama) Return ChildStgSzabadIskolavalasztasValtozo objects filtered by the nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozo[]|ObjectCollection findByMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama(int $mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama) Return ChildStgSzabadIskolavalasztasValtozo objects filtered by the mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozo[]|ObjectCollection findByMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama(int $mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama) Return ChildStgSzabadIskolavalasztasValtozo objects filtered by the mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgSzabadIskolavalasztasValtozo[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgSzabadIskolavalasztasValtozoQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgSzabadIskolavalasztasValtozoQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgSzabadIskolavalasztasValtozo', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgSzabadIskolavalasztasValtozoQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgSzabadIskolavalasztasValtozoQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgSzabadIskolavalasztasValtozoQuery) {

return $criteria;

}

$query = new ChildStgSzabadIskolavalasztasValtozoQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgSzabadIskolavalasztasValtozo|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgSzabadIskolavalasztasValtozo object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgSzabadIskolavalasztasValtozo object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgSzabadIskolavalasztasValtozoQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgSzabadIskolavalasztasValtozo object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgSzabadIskolavalasztasValtozoQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgSzabadIskolavalasztasValtozo object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzabadIskolavalasztasValtozoQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzabadIskolavalasztasValtozoQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the allando\_nepessegbol\_6\_13\_evesek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByAllandoNepessegbol613EvesekSzama(1234); // WHERE allando\_nepessegbol\_6\_13\_evesek\_szama = 1234

\* $query->filterByAllandoNepessegbol613EvesekSzama(array(12, 34)); // WHERE allando\_nepessegbol\_6\_13\_evesek\_szama IN (12, 34)

\* $query->filterByAllandoNepessegbol613EvesekSzama(array('min' => 12)); // WHERE allando\_nepessegbol\_6\_13\_evesek\_szama > 12

\* </code>

\*

\* @param mixed $allandoNepessegbol613EvesekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzabadIskolavalasztasValtozoQuery The current query, for fluid interface

\*/

public function filterByAllandoNepessegbol613EvesekSzama($allandoNepessegbol613EvesekSzama = null, $comparison = null)

{

if (is\_array($allandoNepessegbol613EvesekSzama)) {

$useMinMax = false;

if (isset($allandoNepessegbol613EvesekSzama['min'])) {

$this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA, $allandoNepessegbol613EvesekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($allandoNepessegbol613EvesekSzama['max'])) {

$this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA, $allandoNepessegbol613EvesekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA, $allandoNepessegbol613EvesekSzama, $comparison);

}

/\*\*

\* Filter the query on the allando\_nepessegbol\_14\_evesek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByAllandoNepessegbol14EvesekSzama(1234); // WHERE allando\_nepessegbol\_14\_evesek\_szama = 1234

\* $query->filterByAllandoNepessegbol14EvesekSzama(array(12, 34)); // WHERE allando\_nepessegbol\_14\_evesek\_szama IN (12, 34)

\* $query->filterByAllandoNepessegbol14EvesekSzama(array('min' => 12)); // WHERE allando\_nepessegbol\_14\_evesek\_szama > 12

\* </code>

\*

\* @param mixed $allandoNepessegbol14EvesekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzabadIskolavalasztasValtozoQuery The current query, for fluid interface

\*/

public function filterByAllandoNepessegbol14EvesekSzama($allandoNepessegbol14EvesekSzama = null, $comparison = null)

{

if (is\_array($allandoNepessegbol14EvesekSzama)) {

$useMinMax = false;

if (isset($allandoNepessegbol14EvesekSzama['min'])) {

$this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA, $allandoNepessegbol14EvesekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($allandoNepessegbol14EvesekSzama['max'])) {

$this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA, $allandoNepessegbol14EvesekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA, $allandoNepessegbol14EvesekSzama, $comparison);

}

/\*\*

\* Filter the query on the allando\_nepessegbol\_15\_17\_evesek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByAllandoNepessegbol1517EvesekSzama(1234); // WHERE allando\_nepessegbol\_15\_17\_evesek\_szama = 1234

\* $query->filterByAllandoNepessegbol1517EvesekSzama(array(12, 34)); // WHERE allando\_nepessegbol\_15\_17\_evesek\_szama IN (12, 34)

\* $query->filterByAllandoNepessegbol1517EvesekSzama(array('min' => 12)); // WHERE allando\_nepessegbol\_15\_17\_evesek\_szama > 12

\* </code>

\*

\* @param mixed $allandoNepessegbol1517EvesekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzabadIskolavalasztasValtozoQuery The current query, for fluid interface

\*/

public function filterByAllandoNepessegbol1517EvesekSzama($allandoNepessegbol1517EvesekSzama = null, $comparison = null)

{

if (is\_array($allandoNepessegbol1517EvesekSzama)) {

$useMinMax = false;

if (isset($allandoNepessegbol1517EvesekSzama['min'])) {

$this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA, $allandoNepessegbol1517EvesekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($allandoNepessegbol1517EvesekSzama['max'])) {

$this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA, $allandoNepessegbol1517EvesekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA, $allandoNepessegbol1517EvesekSzama, $comparison);

}

/\*\*

\* Filter the query on the nappali\_altalanos\_iskolai\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliAltalanosIskolaiTanulokSzama(1234); // WHERE nappali\_altalanos\_iskolai\_tanulok\_szama = 1234

\* $query->filterByNappaliAltalanosIskolaiTanulokSzama(array(12, 34)); // WHERE nappali\_altalanos\_iskolai\_tanulok\_szama IN (12, 34)

\* $query->filterByNappaliAltalanosIskolaiTanulokSzama(array('min' => 12)); // WHERE nappali\_altalanos\_iskolai\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $nappaliAltalanosIskolaiTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzabadIskolavalasztasValtozoQuery The current query, for fluid interface

\*/

public function filterByNappaliAltalanosIskolaiTanulokSzama($nappaliAltalanosIskolaiTanulokSzama = null, $comparison = null)

{

if (is\_array($nappaliAltalanosIskolaiTanulokSzama)) {

$useMinMax = false;

if (isset($nappaliAltalanosIskolaiTanulokSzama['min'])) {

$this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA, $nappaliAltalanosIskolaiTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliAltalanosIskolaiTanulokSzama['max'])) {

$this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA, $nappaliAltalanosIskolaiTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA, $nappaliAltalanosIskolaiTanulokSzama, $comparison);

}

/\*\*

\* Filter the query on the nappali\_kozepiskolai\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliKozepiskolaiTanulokSzama(1234); // WHERE nappali\_kozepiskolai\_tanulok\_szama = 1234

\* $query->filterByNappaliKozepiskolaiTanulokSzama(array(12, 34)); // WHERE nappali\_kozepiskolai\_tanulok\_szama IN (12, 34)

\* $query->filterByNappaliKozepiskolaiTanulokSzama(array('min' => 12)); // WHERE nappali\_kozepiskolai\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $nappaliKozepiskolaiTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzabadIskolavalasztasValtozoQuery The current query, for fluid interface

\*/

public function filterByNappaliKozepiskolaiTanulokSzama($nappaliKozepiskolaiTanulokSzama = null, $comparison = null)

{

if (is\_array($nappaliKozepiskolaiTanulokSzama)) {

$useMinMax = false;

if (isset($nappaliKozepiskolaiTanulokSzama['min'])) {

$this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $nappaliKozepiskolaiTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliKozepiskolaiTanulokSzama['max'])) {

$this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $nappaliKozepiskolaiTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $nappaliKozepiskolaiTanulokSzama, $comparison);

}

/\*\*

\* Filter the query on the nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama(1234); // WHERE nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama = 1234

\* $query->filterByNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama(array(12, 34)); // WHERE nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama IN (12, 34)

\* $query->filterByNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama(array('min' => 12)); // WHERE nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $nappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzabadIskolavalasztasValtozoQuery The current query, for fluid interface

\*/

public function filterByNappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama($nappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama = null, $comparison = null)

{

if (is\_array($nappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama)) {

$useMinMax = false;

if (isset($nappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama['min'])) {

$this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, $nappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama['max'])) {

$this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, $nappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, $nappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama, $comparison);

}

/\*\*

\* Filter the query on the mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama(1234); // WHERE mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama = 1234

\* $query->filterByMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama(array(12, 34)); // WHERE mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama IN (12, 34)

\* $query->filterByMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama(array('min' => 12)); // WHERE mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzabadIskolavalasztasValtozoQuery The current query, for fluid interface

\*/

public function filterByMasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama($masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama = null, $comparison = null)

{

if (is\_array($masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama)) {

$useMinMax = false;

if (isset($masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama['min'])) {

$this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA, $masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama['max'])) {

$this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA, $masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA, $masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama, $comparison);

}

/\*\*

\* Filter the query on the mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama(1234); // WHERE mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama = 1234

\* $query->filterByMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama(array(12, 34)); // WHERE mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama IN (12, 34)

\* $query->filterByMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama(array('min' => 12)); // WHERE mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzabadIskolavalasztasValtozoQuery The current query, for fluid interface

\*/

public function filterByMasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama($masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama = null, $comparison = null)

{

if (is\_array($masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama)) {

$useMinMax = false;

if (isset($masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama['min'])) {

$this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama['max'])) {

$this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgSzabadIskolavalasztasValtozo $stgSzabadIskolavalasztasValtozo Object to remove from the list of results

\*

\* @return $this|ChildStgSzabadIskolavalasztasValtozoQuery The current query, for fluid interface

\*/

public function prune($stgSzabadIskolavalasztasValtozo = null)

{

if ($stgSzabadIskolavalasztasValtozo) {

throw new LogicException('StgSzabadIskolavalasztasValtozo object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_szabad\_iskolavalasztas\_valtozo table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzabadIskolavalasztasValtozoTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgSzabadIskolavalasztasValtozoTableMap::clearInstancePool();

StgSzabadIskolavalasztasValtozoTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzabadIskolavalasztasValtozoTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgSzabadIskolavalasztasValtozoTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgSzabadIskolavalasztasValtozoTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgSzabadIskolavalasztasValtozoTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgSzabadIskolavalasztasValtozoQuery

#### StgSzallashely.php

<?php

namespace Base;

use \StgSzallashelyQuery as ChildStgSzallashelyQuery;

use \Exception;

use \PDO;

use Map\StgSzallashelyTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgSzallashely implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgSzallashelyTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the fizetovendeglatas\_szallashelyeinek\_szama field.

\* @var int

\*/

protected $fizetovendeglatas\_szallashelyeinek\_szama;

/\*\*

\* The value for the osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama field.

\* @var int

\*/

protected $osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgSzallashely object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgSzallashely</code> instance. If

\* <code>obj</code> is an instance of <code>StgSzallashely</code>, delegates to

\* <code>equals(StgSzallashely)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgSzallashely The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [fizetovendeglatas\_szallashelyeinek\_szama] column value.

\*

\* @return int

\*/

public function getFizetovendeglatasSzallashelyeinekSzama()

{

return $this->fizetovendeglatas\_szallashelyeinek\_szama;

}

/\*\*

\* Get the [osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama] column value.

\*

\* @return int

\*/

public function getOsszesKereskedelmiSzallashelyFerohelyeinekSzama()

{

return $this->osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgSzallashelyTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgSzallashelyTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgSzallashelyTableMap::translateFieldName('FizetovendeglatasSzallashelyeinekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->fizetovendeglatas\_szallashelyeinek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgSzallashelyTableMap::translateFieldName('OsszesKereskedelmiSzallashelyFerohelyeinekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 4; // 4 = StgSzallashelyTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgSzallashely'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgSzallashely The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgSzallashelyTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgSzallashely The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgSzallashelyTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [fizetovendeglatas\_szallashelyeinek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgSzallashely The current object (for fluent API support)

\*/

public function setFizetovendeglatasSzallashelyeinekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->fizetovendeglatas\_szallashelyeinek\_szama !== $v) {

$this->fizetovendeglatas\_szallashelyeinek\_szama = $v;

$this->modifiedColumns[StgSzallashelyTableMap::COL\_FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA] = true;

}

return $this;

} // setFizetovendeglatasSzallashelyeinekSzama()

/\*\*

\* Set the value of [osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgSzallashely The current object (for fluent API support)

\*/

public function setOsszesKereskedelmiSzallashelyFerohelyeinekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama !== $v) {

$this->osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama = $v;

$this->modifiedColumns[StgSzallashelyTableMap::COL\_OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA] = true;

}

return $this;

} // setOsszesKereskedelmiSzallashelyFerohelyeinekSzama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgSzallashelyTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgSzallashelyQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgSzallashely::setDeleted()

\* @see StgSzallashely::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzallashelyTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgSzallashelyQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzallashelyTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgSzallashelyTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgSzallashelyTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgSzallashelyTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgSzallashelyTableMap::COL\_FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA';

}

if ($this->isColumnModified(StgSzallashelyTableMap::COL\_OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_szallashely (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA':

$stmt->bindValue($identifier, $this->fizetovendeglatas\_szallashelyeinek\_szama, PDO::PARAM\_INT);

break;

case 'OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA':

$stmt->bindValue($identifier, $this->osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgSzallashelyTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getFizetovendeglatasSzallashelyeinekSzama();

break;

case 3:

return $this->getOsszesKereskedelmiSzallashelyFerohelyeinekSzama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgSzallashely'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgSzallashely'][$this->getPrimaryKey()] = true;

$keys = StgSzallashelyTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getFizetovendeglatasSzallashelyeinekSzama(),

$keys[3] => $this->getOsszesKereskedelmiSzallashelyFerohelyeinekSzama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgSzallashely

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgSzallashelyTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgSzallashely

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setFizetovendeglatasSzallashelyeinekSzama($value);

break;

case 3:

$this->setOsszesKereskedelmiSzallashelyFerohelyeinekSzama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgSzallashelyTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setFizetovendeglatasSzallashelyeinekSzama($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setOsszesKereskedelmiSzallashelyFerohelyeinekSzama($arr[$keys[3]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgSzallashely The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgSzallashelyTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgSzallashelyTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgSzallashelyTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgSzallashelyTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgSzallashelyTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgSzallashelyTableMap::COL\_FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA)) {

$criteria->add(StgSzallashelyTableMap::COL\_FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA, $this->fizetovendeglatas\_szallashelyeinek\_szama);

}

if ($this->isColumnModified(StgSzallashelyTableMap::COL\_OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA)) {

$criteria->add(StgSzallashelyTableMap::COL\_OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA, $this->osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgSzallashely object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgSzallashely (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setFizetovendeglatasSzallashelyeinekSzama($this->getFizetovendeglatasSzallashelyeinekSzama());

$copyObj->setOsszesKereskedelmiSzallashelyFerohelyeinekSzama($this->getOsszesKereskedelmiSzallashelyFerohelyeinekSzama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgSzallashely Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->fizetovendeglatas\_szallashelyeinek\_szama = null;

$this->osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgSzallashelyTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgSzallashelyQuery.php

<?php

namespace Base;

use \StgSzallashely as ChildStgSzallashely;

use \StgSzallashelyQuery as ChildStgSzallashelyQuery;

use \Exception;

use Map\StgSzallashelyTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_szallashely' table.

\*

\*

\*

\* @method ChildStgSzallashelyQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgSzallashelyQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgSzallashelyQuery orderByFizetovendeglatasSzallashelyeinekSzama($order = Criteria::ASC) Order by the fizetovendeglatas\_szallashelyeinek\_szama column

\* @method ChildStgSzallashelyQuery orderByOsszesKereskedelmiSzallashelyFerohelyeinekSzama($order = Criteria::ASC) Order by the osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama column

\*

\* @method ChildStgSzallashelyQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgSzallashelyQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgSzallashelyQuery groupByFizetovendeglatasSzallashelyeinekSzama() Group by the fizetovendeglatas\_szallashelyeinek\_szama column

\* @method ChildStgSzallashelyQuery groupByOsszesKereskedelmiSzallashelyFerohelyeinekSzama() Group by the osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama column

\*

\* @method ChildStgSzallashelyQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgSzallashelyQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgSzallashelyQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgSzallashely findOne(ConnectionInterface $con = null) Return the first ChildStgSzallashely matching the query

\* @method ChildStgSzallashely findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgSzallashely matching the query, or a new ChildStgSzallashely object populated from the query conditions when no match is found

\*

\* @method ChildStgSzallashely findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgSzallashely filtered by the telepules\_nev column

\* @method ChildStgSzallashely findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgSzallashely filtered by the telepules\_KSHKOD column

\* @method ChildStgSzallashely findOneByFizetovendeglatasSzallashelyeinekSzama(int $fizetovendeglatas\_szallashelyeinek\_szama) Return the first ChildStgSzallashely filtered by the fizetovendeglatas\_szallashelyeinek\_szama column

\* @method ChildStgSzallashely findOneByOsszesKereskedelmiSzallashelyFerohelyeinekSzama(int $osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama) Return the first ChildStgSzallashely filtered by the osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama column

\*

\* @method ChildStgSzallashely[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgSzallashely objects based on current ModelCriteria

\* @method ChildStgSzallashely[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgSzallashely objects filtered by the telepules\_nev column

\* @method ChildStgSzallashely[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgSzallashely objects filtered by the telepules\_KSHKOD column

\* @method ChildStgSzallashely[]|ObjectCollection findByFizetovendeglatasSzallashelyeinekSzama(int $fizetovendeglatas\_szallashelyeinek\_szama) Return ChildStgSzallashely objects filtered by the fizetovendeglatas\_szallashelyeinek\_szama column

\* @method ChildStgSzallashely[]|ObjectCollection findByOsszesKereskedelmiSzallashelyFerohelyeinekSzama(int $osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama) Return ChildStgSzallashely objects filtered by the osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama column

\* @method ChildStgSzallashely[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgSzallashelyQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgSzallashelyQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgSzallashely', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgSzallashelyQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgSzallashelyQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgSzallashelyQuery) {

return $criteria;

}

$query = new ChildStgSzallashelyQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgSzallashely|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgSzallashely object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgSzallashely object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgSzallashelyQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgSzallashely object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgSzallashelyQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgSzallashely object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzallashelyQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgSzallashelyTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzallashelyQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgSzallashelyTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the fizetovendeglatas\_szallashelyeinek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByFizetovendeglatasSzallashelyeinekSzama(1234); // WHERE fizetovendeglatas\_szallashelyeinek\_szama = 1234

\* $query->filterByFizetovendeglatasSzallashelyeinekSzama(array(12, 34)); // WHERE fizetovendeglatas\_szallashelyeinek\_szama IN (12, 34)

\* $query->filterByFizetovendeglatasSzallashelyeinekSzama(array('min' => 12)); // WHERE fizetovendeglatas\_szallashelyeinek\_szama > 12

\* </code>

\*

\* @param mixed $fizetovendeglatasSzallashelyeinekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzallashelyQuery The current query, for fluid interface

\*/

public function filterByFizetovendeglatasSzallashelyeinekSzama($fizetovendeglatasSzallashelyeinekSzama = null, $comparison = null)

{

if (is\_array($fizetovendeglatasSzallashelyeinekSzama)) {

$useMinMax = false;

if (isset($fizetovendeglatasSzallashelyeinekSzama['min'])) {

$this->addUsingAlias(StgSzallashelyTableMap::COL\_FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA, $fizetovendeglatasSzallashelyeinekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($fizetovendeglatasSzallashelyeinekSzama['max'])) {

$this->addUsingAlias(StgSzallashelyTableMap::COL\_FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA, $fizetovendeglatasSzallashelyeinekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgSzallashelyTableMap::COL\_FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA, $fizetovendeglatasSzallashelyeinekSzama, $comparison);

}

/\*\*

\* Filter the query on the osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByOsszesKereskedelmiSzallashelyFerohelyeinekSzama(1234); // WHERE osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama = 1234

\* $query->filterByOsszesKereskedelmiSzallashelyFerohelyeinekSzama(array(12, 34)); // WHERE osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama IN (12, 34)

\* $query->filterByOsszesKereskedelmiSzallashelyFerohelyeinekSzama(array('min' => 12)); // WHERE osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama > 12

\* </code>

\*

\* @param mixed $osszesKereskedelmiSzallashelyFerohelyeinekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzallashelyQuery The current query, for fluid interface

\*/

public function filterByOsszesKereskedelmiSzallashelyFerohelyeinekSzama($osszesKereskedelmiSzallashelyFerohelyeinekSzama = null, $comparison = null)

{

if (is\_array($osszesKereskedelmiSzallashelyFerohelyeinekSzama)) {

$useMinMax = false;

if (isset($osszesKereskedelmiSzallashelyFerohelyeinekSzama['min'])) {

$this->addUsingAlias(StgSzallashelyTableMap::COL\_OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA, $osszesKereskedelmiSzallashelyFerohelyeinekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($osszesKereskedelmiSzallashelyFerohelyeinekSzama['max'])) {

$this->addUsingAlias(StgSzallashelyTableMap::COL\_OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA, $osszesKereskedelmiSzallashelyFerohelyeinekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgSzallashelyTableMap::COL\_OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA, $osszesKereskedelmiSzallashelyFerohelyeinekSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgSzallashely $stgSzallashely Object to remove from the list of results

\*

\* @return $this|ChildStgSzallashelyQuery The current query, for fluid interface

\*/

public function prune($stgSzallashely = null)

{

if ($stgSzallashely) {

throw new LogicException('StgSzallashely object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_szallashely table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzallashelyTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgSzallashelyTableMap::clearInstancePool();

StgSzallashelyTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzallashelyTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgSzallashelyTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgSzallashelyTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgSzallashelyTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgSzallashelyQuery

#### StgSzemelygepkocsik.php

<?php

namespace Base;

use \StgSzallashely as ChildStgSzallashely;

use \StgSzallashelyQuery as ChildStgSzallashelyQuery;

use \Exception;

use Map\StgSzallashelyTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_szallashely' table.

\*

\*

\*

\* @method ChildStgSzallashelyQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgSzallashelyQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgSzallashelyQuery orderByFizetovendeglatasSzallashelyeinekSzama($order = Criteria::ASC) Order by the fizetovendeglatas\_szallashelyeinek\_szama column

\* @method ChildStgSzallashelyQuery orderByOsszesKereskedelmiSzallashelyFerohelyeinekSzama($order = Criteria::ASC) Order by the osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama column

\*

\* @method ChildStgSzallashelyQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgSzallashelyQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgSzallashelyQuery groupByFizetovendeglatasSzallashelyeinekSzama() Group by the fizetovendeglatas\_szallashelyeinek\_szama column

\* @method ChildStgSzallashelyQuery groupByOsszesKereskedelmiSzallashelyFerohelyeinekSzama() Group by the osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama column

\*

\* @method ChildStgSzallashelyQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgSzallashelyQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgSzallashelyQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgSzallashely findOne(ConnectionInterface $con = null) Return the first ChildStgSzallashely matching the query

\* @method ChildStgSzallashely findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgSzallashely matching the query, or a new ChildStgSzallashely object populated from the query conditions when no match is found

\*

\* @method ChildStgSzallashely findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgSzallashely filtered by the telepules\_nev column

\* @method ChildStgSzallashely findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgSzallashely filtered by the telepules\_KSHKOD column

\* @method ChildStgSzallashely findOneByFizetovendeglatasSzallashelyeinekSzama(int $fizetovendeglatas\_szallashelyeinek\_szama) Return the first ChildStgSzallashely filtered by the fizetovendeglatas\_szallashelyeinek\_szama column

\* @method ChildStgSzallashely findOneByOsszesKereskedelmiSzallashelyFerohelyeinekSzama(int $osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama) Return the first ChildStgSzallashely filtered by the osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama column

\*

\* @method ChildStgSzallashely[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgSzallashely objects based on current ModelCriteria

\* @method ChildStgSzallashely[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgSzallashely objects filtered by the telepules\_nev column

\* @method ChildStgSzallashely[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgSzallashely objects filtered by the telepules\_KSHKOD column

\* @method ChildStgSzallashely[]|ObjectCollection findByFizetovendeglatasSzallashelyeinekSzama(int $fizetovendeglatas\_szallashelyeinek\_szama) Return ChildStgSzallashely objects filtered by the fizetovendeglatas\_szallashelyeinek\_szama column

\* @method ChildStgSzallashely[]|ObjectCollection findByOsszesKereskedelmiSzallashelyFerohelyeinekSzama(int $osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama) Return ChildStgSzallashely objects filtered by the osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama column

\* @method ChildStgSzallashely[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgSzallashelyQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgSzallashelyQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgSzallashely', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgSzallashelyQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgSzallashelyQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgSzallashelyQuery) {

return $criteria;

}

$query = new ChildStgSzallashelyQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgSzallashely|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgSzallashely object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgSzallashely object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgSzallashelyQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgSzallashely object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgSzallashelyQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgSzallashely object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzallashelyQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgSzallashelyTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzallashelyQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgSzallashelyTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the fizetovendeglatas\_szallashelyeinek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByFizetovendeglatasSzallashelyeinekSzama(1234); // WHERE fizetovendeglatas\_szallashelyeinek\_szama = 1234

\* $query->filterByFizetovendeglatasSzallashelyeinekSzama(array(12, 34)); // WHERE fizetovendeglatas\_szallashelyeinek\_szama IN (12, 34)

\* $query->filterByFizetovendeglatasSzallashelyeinekSzama(array('min' => 12)); // WHERE fizetovendeglatas\_szallashelyeinek\_szama > 12

\* </code>

\*

\* @param mixed $fizetovendeglatasSzallashelyeinekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzallashelyQuery The current query, for fluid interface

\*/

public function filterByFizetovendeglatasSzallashelyeinekSzama($fizetovendeglatasSzallashelyeinekSzama = null, $comparison = null)

{

if (is\_array($fizetovendeglatasSzallashelyeinekSzama)) {

$useMinMax = false;

if (isset($fizetovendeglatasSzallashelyeinekSzama['min'])) {

$this->addUsingAlias(StgSzallashelyTableMap::COL\_FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA, $fizetovendeglatasSzallashelyeinekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($fizetovendeglatasSzallashelyeinekSzama['max'])) {

$this->addUsingAlias(StgSzallashelyTableMap::COL\_FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA, $fizetovendeglatasSzallashelyeinekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgSzallashelyTableMap::COL\_FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA, $fizetovendeglatasSzallashelyeinekSzama, $comparison);

}

/\*\*

\* Filter the query on the osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByOsszesKereskedelmiSzallashelyFerohelyeinekSzama(1234); // WHERE osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama = 1234

\* $query->filterByOsszesKereskedelmiSzallashelyFerohelyeinekSzama(array(12, 34)); // WHERE osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama IN (12, 34)

\* $query->filterByOsszesKereskedelmiSzallashelyFerohelyeinekSzama(array('min' => 12)); // WHERE osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama > 12

\* </code>

\*

\* @param mixed $osszesKereskedelmiSzallashelyFerohelyeinekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzallashelyQuery The current query, for fluid interface

\*/

public function filterByOsszesKereskedelmiSzallashelyFerohelyeinekSzama($osszesKereskedelmiSzallashelyFerohelyeinekSzama = null, $comparison = null)

{

if (is\_array($osszesKereskedelmiSzallashelyFerohelyeinekSzama)) {

$useMinMax = false;

if (isset($osszesKereskedelmiSzallashelyFerohelyeinekSzama['min'])) {

$this->addUsingAlias(StgSzallashelyTableMap::COL\_OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA, $osszesKereskedelmiSzallashelyFerohelyeinekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($osszesKereskedelmiSzallashelyFerohelyeinekSzama['max'])) {

$this->addUsingAlias(StgSzallashelyTableMap::COL\_OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA, $osszesKereskedelmiSzallashelyFerohelyeinekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgSzallashelyTableMap::COL\_OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA, $osszesKereskedelmiSzallashelyFerohelyeinekSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgSzallashely $stgSzallashely Object to remove from the list of results

\*

\* @return $this|ChildStgSzallashelyQuery The current query, for fluid interface

\*/

public function prune($stgSzallashely = null)

{

if ($stgSzallashely) {

throw new LogicException('StgSzallashely object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_szallashely table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzallashelyTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgSzallashelyTableMap::clearInstancePool();

StgSzallashelyTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzallashelyTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgSzallashelyTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgSzallashelyTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgSzallashelyTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgSzallashelyQuery

#### StgSzemelygepkocsikQuery.php

<?php

namespace Base;

use \StgSzemelygepkocsik as ChildStgSzemelygepkocsik;

use \StgSzemelygepkocsikQuery as ChildStgSzemelygepkocsikQuery;

use \Exception;

use Map\StgSzemelygepkocsikTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_szemelygepkocsik' table.

\*

\*

\*

\* @method ChildStgSzemelygepkocsikQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgSzemelygepkocsikQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgSzemelygepkocsikQuery orderBySzemelygepkocsikSzama($order = Criteria::ASC) Order by the szemelygepkocsik\_szama column

\*

\* @method ChildStgSzemelygepkocsikQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgSzemelygepkocsikQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgSzemelygepkocsikQuery groupBySzemelygepkocsikSzama() Group by the szemelygepkocsik\_szama column

\*

\* @method ChildStgSzemelygepkocsikQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgSzemelygepkocsikQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgSzemelygepkocsikQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgSzemelygepkocsik findOne(ConnectionInterface $con = null) Return the first ChildStgSzemelygepkocsik matching the query

\* @method ChildStgSzemelygepkocsik findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgSzemelygepkocsik matching the query, or a new ChildStgSzemelygepkocsik object populated from the query conditions when no match is found

\*

\* @method ChildStgSzemelygepkocsik findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgSzemelygepkocsik filtered by the telepules\_nev column

\* @method ChildStgSzemelygepkocsik findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgSzemelygepkocsik filtered by the telepules\_KSHKOD column

\* @method ChildStgSzemelygepkocsik findOneBySzemelygepkocsikSzama(int $szemelygepkocsik\_szama) Return the first ChildStgSzemelygepkocsik filtered by the szemelygepkocsik\_szama column

\*

\* @method ChildStgSzemelygepkocsik[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgSzemelygepkocsik objects based on current ModelCriteria

\* @method ChildStgSzemelygepkocsik[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgSzemelygepkocsik objects filtered by the telepules\_nev column

\* @method ChildStgSzemelygepkocsik[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgSzemelygepkocsik objects filtered by the telepules\_KSHKOD column

\* @method ChildStgSzemelygepkocsik[]|ObjectCollection findBySzemelygepkocsikSzama(int $szemelygepkocsik\_szama) Return ChildStgSzemelygepkocsik objects filtered by the szemelygepkocsik\_szama column

\* @method ChildStgSzemelygepkocsik[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgSzemelygepkocsikQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgSzemelygepkocsikQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgSzemelygepkocsik', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgSzemelygepkocsikQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgSzemelygepkocsikQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgSzemelygepkocsikQuery) {

return $criteria;

}

$query = new ChildStgSzemelygepkocsikQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgSzemelygepkocsik|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgSzemelygepkocsik object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgSzemelygepkocsik object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgSzemelygepkocsikQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgSzemelygepkocsik object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgSzemelygepkocsikQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgSzemelygepkocsik object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzemelygepkocsikQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgSzemelygepkocsikTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzemelygepkocsikQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgSzemelygepkocsikTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the szemelygepkocsik\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterBySzemelygepkocsikSzama(1234); // WHERE szemelygepkocsik\_szama = 1234

\* $query->filterBySzemelygepkocsikSzama(array(12, 34)); // WHERE szemelygepkocsik\_szama IN (12, 34)

\* $query->filterBySzemelygepkocsikSzama(array('min' => 12)); // WHERE szemelygepkocsik\_szama > 12

\* </code>

\*

\* @param mixed $szemelygepkocsikSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgSzemelygepkocsikQuery The current query, for fluid interface

\*/

public function filterBySzemelygepkocsikSzama($szemelygepkocsikSzama = null, $comparison = null)

{

if (is\_array($szemelygepkocsikSzama)) {

$useMinMax = false;

if (isset($szemelygepkocsikSzama['min'])) {

$this->addUsingAlias(StgSzemelygepkocsikTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA, $szemelygepkocsikSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($szemelygepkocsikSzama['max'])) {

$this->addUsingAlias(StgSzemelygepkocsikTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA, $szemelygepkocsikSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgSzemelygepkocsikTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA, $szemelygepkocsikSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgSzemelygepkocsik $stgSzemelygepkocsik Object to remove from the list of results

\*

\* @return $this|ChildStgSzemelygepkocsikQuery The current query, for fluid interface

\*/

public function prune($stgSzemelygepkocsik = null)

{

if ($stgSzemelygepkocsik) {

throw new LogicException('StgSzemelygepkocsik object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_szemelygepkocsik table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzemelygepkocsikTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgSzemelygepkocsikTableMap::clearInstancePool();

StgSzemelygepkocsikTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzemelygepkocsikTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgSzemelygepkocsikTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgSzemelygepkocsikTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgSzemelygepkocsikTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgSzemelygepkocsikQuery

#### StgTanulasiCeluUtazasokAltalanosJellemzese.php

<?php

namespace Base;

use \StgTanulasiCeluUtazasokAltalanosJellemzeseQuery as ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery;

use \Exception;

use \PDO;

use Map\StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgTanulasiCeluUtazasokAltalanosJellemzese implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the nappali\_altalanos\_iskolai\_oktatas field.

\* @var int

\*/

protected $nappali\_altalanos\_iskolai\_oktatas;

/\*\*

\* The value for the nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas field.

\* @var int

\*/

protected $nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas;

/\*\*

\* The value for the felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama field.

\* @var int

\*/

protected $felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama;

/\*\*

\* The value for the nappali\_es\_nem\_nappali\_felsooktatas field.

\* @var int

\*/

protected $nappali\_es\_nem\_nappali\_felsooktatas;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgTanulasiCeluUtazasokAltalanosJellemzese object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgTanulasiCeluUtazasokAltalanosJellemzese</code> instance. If

\* <code>obj</code> is an instance of <code>StgTanulasiCeluUtazasokAltalanosJellemzese</code>, delegates to

\* <code>equals(StgTanulasiCeluUtazasokAltalanosJellemzese)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgTanulasiCeluUtazasokAltalanosJellemzese The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [nappali\_altalanos\_iskolai\_oktatas] column value.

\*

\* @return int

\*/

public function getNappaliAltalanosIskolaiOktatas()

{

return $this->nappali\_altalanos\_iskolai\_oktatas;

}

/\*\*

\* Get the [nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas] column value.

\*

\* @return int

\*/

public function getNappaliKozepiskolaiEsFelnottoktatasiOktatas()

{

return $this->nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas;

}

/\*\*

\* Get the [felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama()

{

return $this->felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama;

}

/\*\*

\* Get the [nappali\_es\_nem\_nappali\_felsooktatas] column value.

\*

\* @return int

\*/

public function getNappaliEsNemNappaliFelsooktatas()

{

return $this->nappali\_es\_nem\_nappali\_felsooktatas;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::translateFieldName('NappaliAltalanosIskolaiOktatas', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_altalanos\_iskolai\_oktatas = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::translateFieldName('NappaliKozepiskolaiEsFelnottoktatasiOktatas', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::translateFieldName('FelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 5 + $startcol : StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::translateFieldName('NappaliEsNemNappaliFelsooktatas', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_es\_nem\_nappali\_felsooktatas = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 6; // 6 = StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgTanulasiCeluUtazasokAltalanosJellemzese'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgTanulasiCeluUtazasokAltalanosJellemzese The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgTanulasiCeluUtazasokAltalanosJellemzese The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [nappali\_altalanos\_iskolai\_oktatas] column.

\*

\* @param int $v new value

\* @return $this|\StgTanulasiCeluUtazasokAltalanosJellemzese The current object (for fluent API support)

\*/

public function setNappaliAltalanosIskolaiOktatas($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_altalanos\_iskolai\_oktatas !== $v) {

$this->nappali\_altalanos\_iskolai\_oktatas = $v;

$this->modifiedColumns[StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS] = true;

}

return $this;

} // setNappaliAltalanosIskolaiOktatas()

/\*\*

\* Set the value of [nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas] column.

\*

\* @param int $v new value

\* @return $this|\StgTanulasiCeluUtazasokAltalanosJellemzese The current object (for fluent API support)

\*/

public function setNappaliKozepiskolaiEsFelnottoktatasiOktatas($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas !== $v) {

$this->nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas = $v;

$this->modifiedColumns[StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS] = true;

}

return $this;

} // setNappaliKozepiskolaiEsFelnottoktatasiOktatas()

/\*\*

\* Set the value of [felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgTanulasiCeluUtazasokAltalanosJellemzese The current object (for fluent API support)

\*/

public function setFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama !== $v) {

$this->felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama = $v;

$this->modifiedColumns[StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA] = true;

}

return $this;

} // setFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama()

/\*\*

\* Set the value of [nappali\_es\_nem\_nappali\_felsooktatas] column.

\*

\* @param int $v new value

\* @return $this|\StgTanulasiCeluUtazasokAltalanosJellemzese The current object (for fluent API support)

\*/

public function setNappaliEsNemNappaliFelsooktatas($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_es\_nem\_nappali\_felsooktatas !== $v) {

$this->nappali\_es\_nem\_nappali\_felsooktatas = $v;

$this->modifiedColumns[StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS] = true;

}

return $this;

} // setNappaliEsNemNappaliFelsooktatas()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgTanulasiCeluUtazasokAltalanosJellemzese::setDeleted()

\* @see StgTanulasiCeluUtazasokAltalanosJellemzese::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS';

}

if ($this->isColumnModified(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS';

}

if ($this->isColumnModified(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA';

}

if ($this->isColumnModified(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS';

}

$sql = sprintf(

'INSERT INTO stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS':

$stmt->bindValue($identifier, $this->nappali\_altalanos\_iskolai\_oktatas, PDO::PARAM\_INT);

break;

case 'NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS':

$stmt->bindValue($identifier, $this->nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas, PDO::PARAM\_INT);

break;

case 'FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama, PDO::PARAM\_INT);

break;

case 'NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS':

$stmt->bindValue($identifier, $this->nappali\_es\_nem\_nappali\_felsooktatas, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getNappaliAltalanosIskolaiOktatas();

break;

case 3:

return $this->getNappaliKozepiskolaiEsFelnottoktatasiOktatas();

break;

case 4:

return $this->getFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama();

break;

case 5:

return $this->getNappaliEsNemNappaliFelsooktatas();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgTanulasiCeluUtazasokAltalanosJellemzese'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgTanulasiCeluUtazasokAltalanosJellemzese'][$this->getPrimaryKey()] = true;

$keys = StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getNappaliAltalanosIskolaiOktatas(),

$keys[3] => $this->getNappaliKozepiskolaiEsFelnottoktatasiOktatas(),

$keys[4] => $this->getFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama(),

$keys[5] => $this->getNappaliEsNemNappaliFelsooktatas(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgTanulasiCeluUtazasokAltalanosJellemzese

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgTanulasiCeluUtazasokAltalanosJellemzese

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setNappaliAltalanosIskolaiOktatas($value);

break;

case 3:

$this->setNappaliKozepiskolaiEsFelnottoktatasiOktatas($value);

break;

case 4:

$this->setFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama($value);

break;

case 5:

$this->setNappaliEsNemNappaliFelsooktatas($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setNappaliAltalanosIskolaiOktatas($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setNappaliKozepiskolaiEsFelnottoktatasiOktatas($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama($arr[$keys[4]]);

}

if (array\_key\_exists($keys[5], $arr)) {

$this->setNappaliEsNemNappaliFelsooktatas($arr[$keys[5]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgTanulasiCeluUtazasokAltalanosJellemzese The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS)) {

$criteria->add(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS, $this->nappali\_altalanos\_iskolai\_oktatas);

}

if ($this->isColumnModified(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS)) {

$criteria->add(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS, $this->nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas);

}

if ($this->isColumnModified(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA)) {

$criteria->add(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, $this->felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama);

}

if ($this->isColumnModified(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS)) {

$criteria->add(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS, $this->nappali\_es\_nem\_nappali\_felsooktatas);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgTanulasiCeluUtazasokAltalanosJellemzese object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgTanulasiCeluUtazasokAltalanosJellemzese (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setNappaliAltalanosIskolaiOktatas($this->getNappaliAltalanosIskolaiOktatas());

$copyObj->setNappaliKozepiskolaiEsFelnottoktatasiOktatas($this->getNappaliKozepiskolaiEsFelnottoktatasiOktatas());

$copyObj->setFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama($this->getFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama());

$copyObj->setNappaliEsNemNappaliFelsooktatas($this->getNappaliEsNemNappaliFelsooktatas());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgTanulasiCeluUtazasokAltalanosJellemzese Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->nappali\_altalanos\_iskolai\_oktatas = null;

$this->nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas = null;

$this->felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama = null;

$this->nappali\_es\_nem\_nappali\_felsooktatas = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgTanulasiCeluUtazasokAltalanosJellemzeseQuery.php

<?php

namespace Base;

use \StgTanulasiCeluUtazasokAltalanosJellemzese as ChildStgTanulasiCeluUtazasokAltalanosJellemzese;

use \StgTanulasiCeluUtazasokAltalanosJellemzeseQuery as ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery;

use \Exception;

use Map\StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese' table.

\*

\*

\*

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery orderByNappaliAltalanosIskolaiOktatas($order = Criteria::ASC) Order by the nappali\_altalanos\_iskolai\_oktatas column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery orderByNappaliKozepiskolaiEsFelnottoktatasiOktatas($order = Criteria::ASC) Order by the nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery orderByFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama($order = Criteria::ASC) Order by the felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery orderByNappaliEsNemNappaliFelsooktatas($order = Criteria::ASC) Order by the nappali\_es\_nem\_nappali\_felsooktatas column

\*

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery groupByNappaliAltalanosIskolaiOktatas() Group by the nappali\_altalanos\_iskolai\_oktatas column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery groupByNappaliKozepiskolaiEsFelnottoktatasiOktatas() Group by the nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery groupByFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama() Group by the felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery groupByNappaliEsNemNappaliFelsooktatas() Group by the nappali\_es\_nem\_nappali\_felsooktatas column

\*

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzese findOne(ConnectionInterface $con = null) Return the first ChildStgTanulasiCeluUtazasokAltalanosJellemzese matching the query

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzese findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgTanulasiCeluUtazasokAltalanosJellemzese matching the query, or a new ChildStgTanulasiCeluUtazasokAltalanosJellemzese object populated from the query conditions when no match is found

\*

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzese findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgTanulasiCeluUtazasokAltalanosJellemzese filtered by the telepules\_nev column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzese findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgTanulasiCeluUtazasokAltalanosJellemzese filtered by the telepules\_KSHKOD column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzese findOneByNappaliAltalanosIskolaiOktatas(int $nappali\_altalanos\_iskolai\_oktatas) Return the first ChildStgTanulasiCeluUtazasokAltalanosJellemzese filtered by the nappali\_altalanos\_iskolai\_oktatas column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzese findOneByNappaliKozepiskolaiEsFelnottoktatasiOktatas(int $nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas) Return the first ChildStgTanulasiCeluUtazasokAltalanosJellemzese filtered by the nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzese findOneByFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama(int $felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama) Return the first ChildStgTanulasiCeluUtazasokAltalanosJellemzese filtered by the felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzese findOneByNappaliEsNemNappaliFelsooktatas(int $nappali\_es\_nem\_nappali\_felsooktatas) Return the first ChildStgTanulasiCeluUtazasokAltalanosJellemzese filtered by the nappali\_es\_nem\_nappali\_felsooktatas column

\*

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzese[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgTanulasiCeluUtazasokAltalanosJellemzese objects based on current ModelCriteria

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzese[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgTanulasiCeluUtazasokAltalanosJellemzese objects filtered by the telepules\_nev column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzese[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgTanulasiCeluUtazasokAltalanosJellemzese objects filtered by the telepules\_KSHKOD column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzese[]|ObjectCollection findByNappaliAltalanosIskolaiOktatas(int $nappali\_altalanos\_iskolai\_oktatas) Return ChildStgTanulasiCeluUtazasokAltalanosJellemzese objects filtered by the nappali\_altalanos\_iskolai\_oktatas column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzese[]|ObjectCollection findByNappaliKozepiskolaiEsFelnottoktatasiOktatas(int $nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas) Return ChildStgTanulasiCeluUtazasokAltalanosJellemzese objects filtered by the nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzese[]|ObjectCollection findByFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama(int $felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama) Return ChildStgTanulasiCeluUtazasokAltalanosJellemzese objects filtered by the felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzese[]|ObjectCollection findByNappaliEsNemNappaliFelsooktatas(int $nappali\_es\_nem\_nappali\_felsooktatas) Return ChildStgTanulasiCeluUtazasokAltalanosJellemzese objects filtered by the nappali\_es\_nem\_nappali\_felsooktatas column

\* @method ChildStgTanulasiCeluUtazasokAltalanosJellemzese[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgTanulasiCeluUtazasokAltalanosJellemzeseQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgTanulasiCeluUtazasokAltalanosJellemzeseQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgTanulasiCeluUtazasokAltalanosJellemzese', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery) {

return $criteria;

}

$query = new ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgTanulasiCeluUtazasokAltalanosJellemzese|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgTanulasiCeluUtazasokAltalanosJellemzese object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgTanulasiCeluUtazasokAltalanosJellemzese object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgTanulasiCeluUtazasokAltalanosJellemzese object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgTanulasiCeluUtazasokAltalanosJellemzese object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the nappali\_altalanos\_iskolai\_oktatas column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliAltalanosIskolaiOktatas(1234); // WHERE nappali\_altalanos\_iskolai\_oktatas = 1234

\* $query->filterByNappaliAltalanosIskolaiOktatas(array(12, 34)); // WHERE nappali\_altalanos\_iskolai\_oktatas IN (12, 34)

\* $query->filterByNappaliAltalanosIskolaiOktatas(array('min' => 12)); // WHERE nappali\_altalanos\_iskolai\_oktatas > 12

\* </code>

\*

\* @param mixed $nappaliAltalanosIskolaiOktatas The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByNappaliAltalanosIskolaiOktatas($nappaliAltalanosIskolaiOktatas = null, $comparison = null)

{

if (is\_array($nappaliAltalanosIskolaiOktatas)) {

$useMinMax = false;

if (isset($nappaliAltalanosIskolaiOktatas['min'])) {

$this->addUsingAlias(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS, $nappaliAltalanosIskolaiOktatas['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliAltalanosIskolaiOktatas['max'])) {

$this->addUsingAlias(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS, $nappaliAltalanosIskolaiOktatas['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS, $nappaliAltalanosIskolaiOktatas, $comparison);

}

/\*\*

\* Filter the query on the nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliKozepiskolaiEsFelnottoktatasiOktatas(1234); // WHERE nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas = 1234

\* $query->filterByNappaliKozepiskolaiEsFelnottoktatasiOktatas(array(12, 34)); // WHERE nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas IN (12, 34)

\* $query->filterByNappaliKozepiskolaiEsFelnottoktatasiOktatas(array('min' => 12)); // WHERE nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas > 12

\* </code>

\*

\* @param mixed $nappaliKozepiskolaiEsFelnottoktatasiOktatas The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByNappaliKozepiskolaiEsFelnottoktatasiOktatas($nappaliKozepiskolaiEsFelnottoktatasiOktatas = null, $comparison = null)

{

if (is\_array($nappaliKozepiskolaiEsFelnottoktatasiOktatas)) {

$useMinMax = false;

if (isset($nappaliKozepiskolaiEsFelnottoktatasiOktatas['min'])) {

$this->addUsingAlias(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS, $nappaliKozepiskolaiEsFelnottoktatasiOktatas['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliKozepiskolaiEsFelnottoktatasiOktatas['max'])) {

$this->addUsingAlias(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS, $nappaliKozepiskolaiEsFelnottoktatasiOktatas['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS, $nappaliKozepiskolaiEsFelnottoktatasiOktatas, $comparison);

}

/\*\*

\* Filter the query on the felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama(1234); // WHERE felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama = 1234

\* $query->filterByFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama(array(12, 34)); // WHERE felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama IN (12, 34)

\* $query->filterByFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama(array('min' => 12)); // WHERE felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama($felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama = null, $comparison = null)

{

if (is\_array($felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama)) {

$useMinMax = false;

if (isset($felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama['min'])) {

$this->addUsingAlias(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, $felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama['max'])) {

$this->addUsingAlias(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, $felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, $felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama, $comparison);

}

/\*\*

\* Filter the query on the nappali\_es\_nem\_nappali\_felsooktatas column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliEsNemNappaliFelsooktatas(1234); // WHERE nappali\_es\_nem\_nappali\_felsooktatas = 1234

\* $query->filterByNappaliEsNemNappaliFelsooktatas(array(12, 34)); // WHERE nappali\_es\_nem\_nappali\_felsooktatas IN (12, 34)

\* $query->filterByNappaliEsNemNappaliFelsooktatas(array('min' => 12)); // WHERE nappali\_es\_nem\_nappali\_felsooktatas > 12

\* </code>

\*

\* @param mixed $nappaliEsNemNappaliFelsooktatas The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByNappaliEsNemNappaliFelsooktatas($nappaliEsNemNappaliFelsooktatas = null, $comparison = null)

{

if (is\_array($nappaliEsNemNappaliFelsooktatas)) {

$useMinMax = false;

if (isset($nappaliEsNemNappaliFelsooktatas['min'])) {

$this->addUsingAlias(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS, $nappaliEsNemNappaliFelsooktatas['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliEsNemNappaliFelsooktatas['max'])) {

$this->addUsingAlias(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS, $nappaliEsNemNappaliFelsooktatas['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS, $nappaliEsNemNappaliFelsooktatas, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgTanulasiCeluUtazasokAltalanosJellemzese $stgTanulasiCeluUtazasokAltalanosJellemzese Object to remove from the list of results

\*

\* @return $this|ChildStgTanulasiCeluUtazasokAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function prune($stgTanulasiCeluUtazasokAltalanosJellemzese = null)

{

if ($stgTanulasiCeluUtazasokAltalanosJellemzese) {

throw new LogicException('StgTanulasiCeluUtazasokAltalanosJellemzese object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::clearInstancePool();

StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgTanulasiCeluUtazasokAltalanosJellemzeseQuery

#### StgTanulokEsHallgatokSzama.php

<?php

namespace Base;

use \StgTanulokEsHallgatokSzamaQuery as ChildStgTanulokEsHallgatokSzamaQuery;

use \Exception;

use \PDO;

use Map\StgTanulokEsHallgatokSzamaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgTanulokEsHallgatokSzama implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgTanulokEsHallgatokSzamaTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the nappali\_alt\_iskolai\_tanulok\_szama field.

\* @var int

\*/

protected $nappali\_alt\_iskolai\_tanulok\_szama;

/\*\*

\* The value for the felnottoktatasban\_alt\_iskolai\_tanulok\_szama field.

\* @var int

\*/

protected $felnottoktatasban\_alt\_iskolai\_tanulok\_szama;

/\*\*

\* The value for the nappali\_kozepiskolai\_tanulok\_szama field.

\* @var int

\*/

protected $nappali\_kozepiskolai\_tanulok\_szama;

/\*\*

\* The value for the felnottoktatasban\_kozepiskolai\_tanulok\_szama field.

\* @var int

\*/

protected $felnottoktatasban\_kozepiskolai\_tanulok\_szama;

/\*\*

\* The value for the nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama field.

\* @var int

\*/

protected $nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama;

/\*\*

\* The value for the felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama field.

\* @var int

\*/

protected $felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama;

/\*\*

\* The value for the nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama field.

\* @var int

\*/

protected $nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama;

/\*\*

\* The value for the nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama field.

\* @var int

\*/

protected $nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgTanulokEsHallgatokSzama object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgTanulokEsHallgatokSzama</code> instance. If

\* <code>obj</code> is an instance of <code>StgTanulokEsHallgatokSzama</code>, delegates to

\* <code>equals(StgTanulokEsHallgatokSzama)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgTanulokEsHallgatokSzama The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [nappali\_alt\_iskolai\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getNappaliAltIskolaiTanulokSzama()

{

return $this->nappali\_alt\_iskolai\_tanulok\_szama;

}

/\*\*

\* Get the [felnottoktatasban\_alt\_iskolai\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getFelnottoktatasbanAltIskolaiTanulokSzama()

{

return $this->felnottoktatasban\_alt\_iskolai\_tanulok\_szama;

}

/\*\*

\* Get the [nappali\_kozepiskolai\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getNappaliKozepiskolaiTanulokSzama()

{

return $this->nappali\_kozepiskolai\_tanulok\_szama;

}

/\*\*

\* Get the [felnottoktatasban\_kozepiskolai\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getFelnottoktatasbanKozepiskolaiTanulokSzama()

{

return $this->felnottoktatasban\_kozepiskolai\_tanulok\_szama;

}

/\*\*

\* Get the [nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama()

{

return $this->nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama;

}

/\*\*

\* Get the [felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama] column value.

\*

\* @return int

\*/

public function getFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama()

{

return $this->felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama;

}

/\*\*

\* Get the [nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama] column value.

\*

\* @return int

\*/

public function getNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama()

{

return $this->nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama;

}

/\*\*

\* Get the [nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama] column value.

\*

\* @return int

\*/

public function getNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama()

{

return $this->nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgTanulokEsHallgatokSzamaTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgTanulokEsHallgatokSzamaTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgTanulokEsHallgatokSzamaTableMap::translateFieldName('NappaliAltIskolaiTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_alt\_iskolai\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgTanulokEsHallgatokSzamaTableMap::translateFieldName('FelnottoktatasbanAltIskolaiTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->felnottoktatasban\_alt\_iskolai\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgTanulokEsHallgatokSzamaTableMap::translateFieldName('NappaliKozepiskolaiTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_kozepiskolai\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 5 + $startcol : StgTanulokEsHallgatokSzamaTableMap::translateFieldName('FelnottoktatasbanKozepiskolaiTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->felnottoktatasban\_kozepiskolai\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 6 + $startcol : StgTanulokEsHallgatokSzamaTableMap::translateFieldName('NappaliSzakiskolaiSpecSzakiskolaiTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 7 + $startcol : StgTanulokEsHallgatokSzamaTableMap::translateFieldName('FelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 8 + $startcol : StgTanulokEsHallgatokSzamaTableMap::translateFieldName('NappaliFelsofokuAlapMesterkepzesbenHallgatokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 9 + $startcol : StgTanulokEsHallgatokSzamaTableMap::translateFieldName('NemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 10; // 10 = StgTanulokEsHallgatokSzamaTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgTanulokEsHallgatokSzama'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgTanulokEsHallgatokSzama The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgTanulokEsHallgatokSzamaTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgTanulokEsHallgatokSzama The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgTanulokEsHallgatokSzamaTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [nappali\_alt\_iskolai\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgTanulokEsHallgatokSzama The current object (for fluent API support)

\*/

public function setNappaliAltIskolaiTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_alt\_iskolai\_tanulok\_szama !== $v) {

$this->nappali\_alt\_iskolai\_tanulok\_szama = $v;

$this->modifiedColumns[StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA] = true;

}

return $this;

} // setNappaliAltIskolaiTanulokSzama()

/\*\*

\* Set the value of [felnottoktatasban\_alt\_iskolai\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgTanulokEsHallgatokSzama The current object (for fluent API support)

\*/

public function setFelnottoktatasbanAltIskolaiTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->felnottoktatasban\_alt\_iskolai\_tanulok\_szama !== $v) {

$this->felnottoktatasban\_alt\_iskolai\_tanulok\_szama = $v;

$this->modifiedColumns[StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA] = true;

}

return $this;

} // setFelnottoktatasbanAltIskolaiTanulokSzama()

/\*\*

\* Set the value of [nappali\_kozepiskolai\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgTanulokEsHallgatokSzama The current object (for fluent API support)

\*/

public function setNappaliKozepiskolaiTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_kozepiskolai\_tanulok\_szama !== $v) {

$this->nappali\_kozepiskolai\_tanulok\_szama = $v;

$this->modifiedColumns[StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA] = true;

}

return $this;

} // setNappaliKozepiskolaiTanulokSzama()

/\*\*

\* Set the value of [felnottoktatasban\_kozepiskolai\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgTanulokEsHallgatokSzama The current object (for fluent API support)

\*/

public function setFelnottoktatasbanKozepiskolaiTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->felnottoktatasban\_kozepiskolai\_tanulok\_szama !== $v) {

$this->felnottoktatasban\_kozepiskolai\_tanulok\_szama = $v;

$this->modifiedColumns[StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA] = true;

}

return $this;

} // setFelnottoktatasbanKozepiskolaiTanulokSzama()

/\*\*

\* Set the value of [nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgTanulokEsHallgatokSzama The current object (for fluent API support)

\*/

public function setNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama !== $v) {

$this->nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama = $v;

$this->modifiedColumns[StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA] = true;

}

return $this;

} // setNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama()

/\*\*

\* Set the value of [felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgTanulokEsHallgatokSzama The current object (for fluent API support)

\*/

public function setFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama !== $v) {

$this->felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama = $v;

$this->modifiedColumns[StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA] = true;

}

return $this;

} // setFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama()

/\*\*

\* Set the value of [nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgTanulokEsHallgatokSzama The current object (for fluent API support)

\*/

public function setNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama !== $v) {

$this->nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama = $v;

$this->modifiedColumns[StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA] = true;

}

return $this;

} // setNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama()

/\*\*

\* Set the value of [nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgTanulokEsHallgatokSzama The current object (for fluent API support)

\*/

public function setNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama !== $v) {

$this->nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama = $v;

$this->modifiedColumns[StgTanulokEsHallgatokSzamaTableMap::COL\_NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA] = true;

}

return $this;

} // setNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgTanulokEsHallgatokSzamaTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgTanulokEsHallgatokSzamaQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgTanulokEsHallgatokSzama::setDeleted()

\* @see StgTanulokEsHallgatokSzama::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTanulokEsHallgatokSzamaTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgTanulokEsHallgatokSzamaQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTanulokEsHallgatokSzamaTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgTanulokEsHallgatokSzamaTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA';

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA';

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA';

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA';

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA';

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA';

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA';

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_tanulok\_es\_hallgatok\_szama (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->nappali\_alt\_iskolai\_tanulok\_szama, PDO::PARAM\_INT);

break;

case 'FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->felnottoktatasban\_alt\_iskolai\_tanulok\_szama, PDO::PARAM\_INT);

break;

case 'NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->nappali\_kozepiskolai\_tanulok\_szama, PDO::PARAM\_INT);

break;

case 'FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->felnottoktatasban\_kozepiskolai\_tanulok\_szama, PDO::PARAM\_INT);

break;

case 'NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama, PDO::PARAM\_INT);

break;

case 'FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA':

$stmt->bindValue($identifier, $this->felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama, PDO::PARAM\_INT);

break;

case 'NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA':

$stmt->bindValue($identifier, $this->nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama, PDO::PARAM\_INT);

break;

case 'NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA':

$stmt->bindValue($identifier, $this->nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgTanulokEsHallgatokSzamaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getNappaliAltIskolaiTanulokSzama();

break;

case 3:

return $this->getFelnottoktatasbanAltIskolaiTanulokSzama();

break;

case 4:

return $this->getNappaliKozepiskolaiTanulokSzama();

break;

case 5:

return $this->getFelnottoktatasbanKozepiskolaiTanulokSzama();

break;

case 6:

return $this->getNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama();

break;

case 7:

return $this->getFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama();

break;

case 8:

return $this->getNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama();

break;

case 9:

return $this->getNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgTanulokEsHallgatokSzama'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgTanulokEsHallgatokSzama'][$this->getPrimaryKey()] = true;

$keys = StgTanulokEsHallgatokSzamaTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getNappaliAltIskolaiTanulokSzama(),

$keys[3] => $this->getFelnottoktatasbanAltIskolaiTanulokSzama(),

$keys[4] => $this->getNappaliKozepiskolaiTanulokSzama(),

$keys[5] => $this->getFelnottoktatasbanKozepiskolaiTanulokSzama(),

$keys[6] => $this->getNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama(),

$keys[7] => $this->getFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama(),

$keys[8] => $this->getNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama(),

$keys[9] => $this->getNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgTanulokEsHallgatokSzama

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgTanulokEsHallgatokSzamaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgTanulokEsHallgatokSzama

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setNappaliAltIskolaiTanulokSzama($value);

break;

case 3:

$this->setFelnottoktatasbanAltIskolaiTanulokSzama($value);

break;

case 4:

$this->setNappaliKozepiskolaiTanulokSzama($value);

break;

case 5:

$this->setFelnottoktatasbanKozepiskolaiTanulokSzama($value);

break;

case 6:

$this->setNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama($value);

break;

case 7:

$this->setFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama($value);

break;

case 8:

$this->setNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama($value);

break;

case 9:

$this->setNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgTanulokEsHallgatokSzamaTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setNappaliAltIskolaiTanulokSzama($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setFelnottoktatasbanAltIskolaiTanulokSzama($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setNappaliKozepiskolaiTanulokSzama($arr[$keys[4]]);

}

if (array\_key\_exists($keys[5], $arr)) {

$this->setFelnottoktatasbanKozepiskolaiTanulokSzama($arr[$keys[5]]);

}

if (array\_key\_exists($keys[6], $arr)) {

$this->setNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama($arr[$keys[6]]);

}

if (array\_key\_exists($keys[7], $arr)) {

$this->setFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama($arr[$keys[7]]);

}

if (array\_key\_exists($keys[8], $arr)) {

$this->setNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama($arr[$keys[8]]);

}

if (array\_key\_exists($keys[9], $arr)) {

$this->setNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama($arr[$keys[9]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgTanulokEsHallgatokSzama The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgTanulokEsHallgatokSzamaTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgTanulokEsHallgatokSzamaTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgTanulokEsHallgatokSzamaTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA)) {

$criteria->add(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA, $this->nappali\_alt\_iskolai\_tanulok\_szama);

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA)) {

$criteria->add(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA, $this->felnottoktatasban\_alt\_iskolai\_tanulok\_szama);

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA)) {

$criteria->add(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $this->nappali\_kozepiskolai\_tanulok\_szama);

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA)) {

$criteria->add(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA, $this->felnottoktatasban\_kozepiskolai\_tanulok\_szama);

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA)) {

$criteria->add(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, $this->nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama);

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA)) {

$criteria->add(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, $this->felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama);

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA)) {

$criteria->add(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA, $this->nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama);

}

if ($this->isColumnModified(StgTanulokEsHallgatokSzamaTableMap::COL\_NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA)) {

$criteria->add(StgTanulokEsHallgatokSzamaTableMap::COL\_NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA, $this->nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgTanulokEsHallgatokSzama object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgTanulokEsHallgatokSzama (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setNappaliAltIskolaiTanulokSzama($this->getNappaliAltIskolaiTanulokSzama());

$copyObj->setFelnottoktatasbanAltIskolaiTanulokSzama($this->getFelnottoktatasbanAltIskolaiTanulokSzama());

$copyObj->setNappaliKozepiskolaiTanulokSzama($this->getNappaliKozepiskolaiTanulokSzama());

$copyObj->setFelnottoktatasbanKozepiskolaiTanulokSzama($this->getFelnottoktatasbanKozepiskolaiTanulokSzama());

$copyObj->setNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama($this->getNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama());

$copyObj->setFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama($this->getFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama());

$copyObj->setNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama($this->getNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama());

$copyObj->setNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama($this->getNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgTanulokEsHallgatokSzama Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->nappali\_alt\_iskolai\_tanulok\_szama = null;

$this->felnottoktatasban\_alt\_iskolai\_tanulok\_szama = null;

$this->nappali\_kozepiskolai\_tanulok\_szama = null;

$this->felnottoktatasban\_kozepiskolai\_tanulok\_szama = null;

$this->nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama = null;

$this->felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama = null;

$this->nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama = null;

$this->nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgTanulokEsHallgatokSzamaTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgTanulokEsHallgatokSzamaQuery.php

<?php

namespace Base;

use \StgTanulokEsHallgatokSzama as ChildStgTanulokEsHallgatokSzama;

use \StgTanulokEsHallgatokSzamaQuery as ChildStgTanulokEsHallgatokSzamaQuery;

use \Exception;

use Map\StgTanulokEsHallgatokSzamaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_tanulok\_es\_hallgatok\_szama' table.

\*

\*

\*

\* @method ChildStgTanulokEsHallgatokSzamaQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgTanulokEsHallgatokSzamaQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgTanulokEsHallgatokSzamaQuery orderByNappaliAltIskolaiTanulokSzama($order = Criteria::ASC) Order by the nappali\_alt\_iskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzamaQuery orderByFelnottoktatasbanAltIskolaiTanulokSzama($order = Criteria::ASC) Order by the felnottoktatasban\_alt\_iskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzamaQuery orderByNappaliKozepiskolaiTanulokSzama($order = Criteria::ASC) Order by the nappali\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzamaQuery orderByFelnottoktatasbanKozepiskolaiTanulokSzama($order = Criteria::ASC) Order by the felnottoktatasban\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzamaQuery orderByNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama($order = Criteria::ASC) Order by the nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzamaQuery orderByFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama($order = Criteria::ASC) Order by the felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzamaQuery orderByNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama($order = Criteria::ASC) Order by the nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama column

\* @method ChildStgTanulokEsHallgatokSzamaQuery orderByNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama($order = Criteria::ASC) Order by the nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama column

\*

\* @method ChildStgTanulokEsHallgatokSzamaQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgTanulokEsHallgatokSzamaQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgTanulokEsHallgatokSzamaQuery groupByNappaliAltIskolaiTanulokSzama() Group by the nappali\_alt\_iskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzamaQuery groupByFelnottoktatasbanAltIskolaiTanulokSzama() Group by the felnottoktatasban\_alt\_iskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzamaQuery groupByNappaliKozepiskolaiTanulokSzama() Group by the nappali\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzamaQuery groupByFelnottoktatasbanKozepiskolaiTanulokSzama() Group by the felnottoktatasban\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzamaQuery groupByNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama() Group by the nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzamaQuery groupByFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama() Group by the felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzamaQuery groupByNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama() Group by the nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama column

\* @method ChildStgTanulokEsHallgatokSzamaQuery groupByNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama() Group by the nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama column

\*

\* @method ChildStgTanulokEsHallgatokSzamaQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgTanulokEsHallgatokSzamaQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgTanulokEsHallgatokSzamaQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgTanulokEsHallgatokSzama findOne(ConnectionInterface $con = null) Return the first ChildStgTanulokEsHallgatokSzama matching the query

\* @method ChildStgTanulokEsHallgatokSzama findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgTanulokEsHallgatokSzama matching the query, or a new ChildStgTanulokEsHallgatokSzama object populated from the query conditions when no match is found

\*

\* @method ChildStgTanulokEsHallgatokSzama findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgTanulokEsHallgatokSzama filtered by the telepules\_nev column

\* @method ChildStgTanulokEsHallgatokSzama findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgTanulokEsHallgatokSzama filtered by the telepules\_KSHKOD column

\* @method ChildStgTanulokEsHallgatokSzama findOneByNappaliAltIskolaiTanulokSzama(int $nappali\_alt\_iskolai\_tanulok\_szama) Return the first ChildStgTanulokEsHallgatokSzama filtered by the nappali\_alt\_iskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzama findOneByFelnottoktatasbanAltIskolaiTanulokSzama(int $felnottoktatasban\_alt\_iskolai\_tanulok\_szama) Return the first ChildStgTanulokEsHallgatokSzama filtered by the felnottoktatasban\_alt\_iskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzama findOneByNappaliKozepiskolaiTanulokSzama(int $nappali\_kozepiskolai\_tanulok\_szama) Return the first ChildStgTanulokEsHallgatokSzama filtered by the nappali\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzama findOneByFelnottoktatasbanKozepiskolaiTanulokSzama(int $felnottoktatasban\_kozepiskolai\_tanulok\_szama) Return the first ChildStgTanulokEsHallgatokSzama filtered by the felnottoktatasban\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzama findOneByNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama(int $nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama) Return the first ChildStgTanulokEsHallgatokSzama filtered by the nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzama findOneByFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama(int $felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama) Return the first ChildStgTanulokEsHallgatokSzama filtered by the felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzama findOneByNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama(int $nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama) Return the first ChildStgTanulokEsHallgatokSzama filtered by the nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama column

\* @method ChildStgTanulokEsHallgatokSzama findOneByNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama(int $nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama) Return the first ChildStgTanulokEsHallgatokSzama filtered by the nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama column

\*

\* @method ChildStgTanulokEsHallgatokSzama[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgTanulokEsHallgatokSzama objects based on current ModelCriteria

\* @method ChildStgTanulokEsHallgatokSzama[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgTanulokEsHallgatokSzama objects filtered by the telepules\_nev column

\* @method ChildStgTanulokEsHallgatokSzama[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgTanulokEsHallgatokSzama objects filtered by the telepules\_KSHKOD column

\* @method ChildStgTanulokEsHallgatokSzama[]|ObjectCollection findByNappaliAltIskolaiTanulokSzama(int $nappali\_alt\_iskolai\_tanulok\_szama) Return ChildStgTanulokEsHallgatokSzama objects filtered by the nappali\_alt\_iskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzama[]|ObjectCollection findByFelnottoktatasbanAltIskolaiTanulokSzama(int $felnottoktatasban\_alt\_iskolai\_tanulok\_szama) Return ChildStgTanulokEsHallgatokSzama objects filtered by the felnottoktatasban\_alt\_iskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzama[]|ObjectCollection findByNappaliKozepiskolaiTanulokSzama(int $nappali\_kozepiskolai\_tanulok\_szama) Return ChildStgTanulokEsHallgatokSzama objects filtered by the nappali\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzama[]|ObjectCollection findByFelnottoktatasbanKozepiskolaiTanulokSzama(int $felnottoktatasban\_kozepiskolai\_tanulok\_szama) Return ChildStgTanulokEsHallgatokSzama objects filtered by the felnottoktatasban\_kozepiskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzama[]|ObjectCollection findByNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama(int $nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama) Return ChildStgTanulokEsHallgatokSzama objects filtered by the nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzama[]|ObjectCollection findByFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama(int $felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama) Return ChildStgTanulokEsHallgatokSzama objects filtered by the felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama column

\* @method ChildStgTanulokEsHallgatokSzama[]|ObjectCollection findByNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama(int $nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama) Return ChildStgTanulokEsHallgatokSzama objects filtered by the nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama column

\* @method ChildStgTanulokEsHallgatokSzama[]|ObjectCollection findByNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama(int $nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama) Return ChildStgTanulokEsHallgatokSzama objects filtered by the nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama column

\* @method ChildStgTanulokEsHallgatokSzama[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgTanulokEsHallgatokSzamaQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgTanulokEsHallgatokSzamaQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgTanulokEsHallgatokSzama', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgTanulokEsHallgatokSzamaQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgTanulokEsHallgatokSzamaQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgTanulokEsHallgatokSzamaQuery) {

return $criteria;

}

$query = new ChildStgTanulokEsHallgatokSzamaQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgTanulokEsHallgatokSzama|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgTanulokEsHallgatokSzama object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgTanulokEsHallgatokSzama object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgTanulokEsHallgatokSzamaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgTanulokEsHallgatokSzama object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgTanulokEsHallgatokSzamaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgTanulokEsHallgatokSzama object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTanulokEsHallgatokSzamaQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTanulokEsHallgatokSzamaQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the nappali\_alt\_iskolai\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliAltIskolaiTanulokSzama(1234); // WHERE nappali\_alt\_iskolai\_tanulok\_szama = 1234

\* $query->filterByNappaliAltIskolaiTanulokSzama(array(12, 34)); // WHERE nappali\_alt\_iskolai\_tanulok\_szama IN (12, 34)

\* $query->filterByNappaliAltIskolaiTanulokSzama(array('min' => 12)); // WHERE nappali\_alt\_iskolai\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $nappaliAltIskolaiTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTanulokEsHallgatokSzamaQuery The current query, for fluid interface

\*/

public function filterByNappaliAltIskolaiTanulokSzama($nappaliAltIskolaiTanulokSzama = null, $comparison = null)

{

if (is\_array($nappaliAltIskolaiTanulokSzama)) {

$useMinMax = false;

if (isset($nappaliAltIskolaiTanulokSzama['min'])) {

$this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA, $nappaliAltIskolaiTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliAltIskolaiTanulokSzama['max'])) {

$this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA, $nappaliAltIskolaiTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA, $nappaliAltIskolaiTanulokSzama, $comparison);

}

/\*\*

\* Filter the query on the felnottoktatasban\_alt\_iskolai\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByFelnottoktatasbanAltIskolaiTanulokSzama(1234); // WHERE felnottoktatasban\_alt\_iskolai\_tanulok\_szama = 1234

\* $query->filterByFelnottoktatasbanAltIskolaiTanulokSzama(array(12, 34)); // WHERE felnottoktatasban\_alt\_iskolai\_tanulok\_szama IN (12, 34)

\* $query->filterByFelnottoktatasbanAltIskolaiTanulokSzama(array('min' => 12)); // WHERE felnottoktatasban\_alt\_iskolai\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $felnottoktatasbanAltIskolaiTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTanulokEsHallgatokSzamaQuery The current query, for fluid interface

\*/

public function filterByFelnottoktatasbanAltIskolaiTanulokSzama($felnottoktatasbanAltIskolaiTanulokSzama = null, $comparison = null)

{

if (is\_array($felnottoktatasbanAltIskolaiTanulokSzama)) {

$useMinMax = false;

if (isset($felnottoktatasbanAltIskolaiTanulokSzama['min'])) {

$this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA, $felnottoktatasbanAltIskolaiTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($felnottoktatasbanAltIskolaiTanulokSzama['max'])) {

$this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA, $felnottoktatasbanAltIskolaiTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA, $felnottoktatasbanAltIskolaiTanulokSzama, $comparison);

}

/\*\*

\* Filter the query on the nappali\_kozepiskolai\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliKozepiskolaiTanulokSzama(1234); // WHERE nappali\_kozepiskolai\_tanulok\_szama = 1234

\* $query->filterByNappaliKozepiskolaiTanulokSzama(array(12, 34)); // WHERE nappali\_kozepiskolai\_tanulok\_szama IN (12, 34)

\* $query->filterByNappaliKozepiskolaiTanulokSzama(array('min' => 12)); // WHERE nappali\_kozepiskolai\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $nappaliKozepiskolaiTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTanulokEsHallgatokSzamaQuery The current query, for fluid interface

\*/

public function filterByNappaliKozepiskolaiTanulokSzama($nappaliKozepiskolaiTanulokSzama = null, $comparison = null)

{

if (is\_array($nappaliKozepiskolaiTanulokSzama)) {

$useMinMax = false;

if (isset($nappaliKozepiskolaiTanulokSzama['min'])) {

$this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $nappaliKozepiskolaiTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliKozepiskolaiTanulokSzama['max'])) {

$this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $nappaliKozepiskolaiTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, $nappaliKozepiskolaiTanulokSzama, $comparison);

}

/\*\*

\* Filter the query on the felnottoktatasban\_kozepiskolai\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByFelnottoktatasbanKozepiskolaiTanulokSzama(1234); // WHERE felnottoktatasban\_kozepiskolai\_tanulok\_szama = 1234

\* $query->filterByFelnottoktatasbanKozepiskolaiTanulokSzama(array(12, 34)); // WHERE felnottoktatasban\_kozepiskolai\_tanulok\_szama IN (12, 34)

\* $query->filterByFelnottoktatasbanKozepiskolaiTanulokSzama(array('min' => 12)); // WHERE felnottoktatasban\_kozepiskolai\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $felnottoktatasbanKozepiskolaiTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTanulokEsHallgatokSzamaQuery The current query, for fluid interface

\*/

public function filterByFelnottoktatasbanKozepiskolaiTanulokSzama($felnottoktatasbanKozepiskolaiTanulokSzama = null, $comparison = null)

{

if (is\_array($felnottoktatasbanKozepiskolaiTanulokSzama)) {

$useMinMax = false;

if (isset($felnottoktatasbanKozepiskolaiTanulokSzama['min'])) {

$this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA, $felnottoktatasbanKozepiskolaiTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($felnottoktatasbanKozepiskolaiTanulokSzama['max'])) {

$this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA, $felnottoktatasbanKozepiskolaiTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA, $felnottoktatasbanKozepiskolaiTanulokSzama, $comparison);

}

/\*\*

\* Filter the query on the nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama(1234); // WHERE nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama = 1234

\* $query->filterByNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama(array(12, 34)); // WHERE nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama IN (12, 34)

\* $query->filterByNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama(array('min' => 12)); // WHERE nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $nappaliSzakiskolaiSpecSzakiskolaiTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTanulokEsHallgatokSzamaQuery The current query, for fluid interface

\*/

public function filterByNappaliSzakiskolaiSpecSzakiskolaiTanulokSzama($nappaliSzakiskolaiSpecSzakiskolaiTanulokSzama = null, $comparison = null)

{

if (is\_array($nappaliSzakiskolaiSpecSzakiskolaiTanulokSzama)) {

$useMinMax = false;

if (isset($nappaliSzakiskolaiSpecSzakiskolaiTanulokSzama['min'])) {

$this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, $nappaliSzakiskolaiSpecSzakiskolaiTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliSzakiskolaiSpecSzakiskolaiTanulokSzama['max'])) {

$this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, $nappaliSzakiskolaiSpecSzakiskolaiTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, $nappaliSzakiskolaiSpecSzakiskolaiTanulokSzama, $comparison);

}

/\*\*

\* Filter the query on the felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama(1234); // WHERE felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama = 1234

\* $query->filterByFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama(array(12, 34)); // WHERE felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama IN (12, 34)

\* $query->filterByFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama(array('min' => 12)); // WHERE felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama > 12

\* </code>

\*

\* @param mixed $felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTanulokEsHallgatokSzamaQuery The current query, for fluid interface

\*/

public function filterByFelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama($felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama = null, $comparison = null)

{

if (is\_array($felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama)) {

$useMinMax = false;

if (isset($felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama['min'])) {

$this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, $felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama['max'])) {

$this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, $felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, $felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama, $comparison);

}

/\*\*

\* Filter the query on the nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama(1234); // WHERE nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama = 1234

\* $query->filterByNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama(array(12, 34)); // WHERE nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama IN (12, 34)

\* $query->filterByNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama(array('min' => 12)); // WHERE nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama > 12

\* </code>

\*

\* @param mixed $nappaliFelsofokuAlapMesterkepzesbenHallgatokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTanulokEsHallgatokSzamaQuery The current query, for fluid interface

\*/

public function filterByNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama($nappaliFelsofokuAlapMesterkepzesbenHallgatokSzama = null, $comparison = null)

{

if (is\_array($nappaliFelsofokuAlapMesterkepzesbenHallgatokSzama)) {

$useMinMax = false;

if (isset($nappaliFelsofokuAlapMesterkepzesbenHallgatokSzama['min'])) {

$this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA, $nappaliFelsofokuAlapMesterkepzesbenHallgatokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nappaliFelsofokuAlapMesterkepzesbenHallgatokSzama['max'])) {

$this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA, $nappaliFelsofokuAlapMesterkepzesbenHallgatokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA, $nappaliFelsofokuAlapMesterkepzesbenHallgatokSzama, $comparison);

}

/\*\*

\* Filter the query on the nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama(1234); // WHERE nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama = 1234

\* $query->filterByNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama(array(12, 34)); // WHERE nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama IN (12, 34)

\* $query->filterByNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama(array('min' => 12)); // WHERE nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama > 12

\* </code>

\*

\* @param mixed $nemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTanulokEsHallgatokSzamaQuery The current query, for fluid interface

\*/

public function filterByNemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama($nemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama = null, $comparison = null)

{

if (is\_array($nemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama)) {

$useMinMax = false;

if (isset($nemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama['min'])) {

$this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA, $nemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($nemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama['max'])) {

$this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA, $nemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgTanulokEsHallgatokSzamaTableMap::COL\_NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA, $nemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgTanulokEsHallgatokSzama $stgTanulokEsHallgatokSzama Object to remove from the list of results

\*

\* @return $this|ChildStgTanulokEsHallgatokSzamaQuery The current query, for fluid interface

\*/

public function prune($stgTanulokEsHallgatokSzama = null)

{

if ($stgTanulokEsHallgatokSzama) {

throw new LogicException('StgTanulokEsHallgatokSzama object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_tanulok\_es\_hallgatok\_szama table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTanulokEsHallgatokSzamaTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgTanulokEsHallgatokSzamaTableMap::clearInstancePool();

StgTanulokEsHallgatokSzamaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTanulokEsHallgatokSzamaTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgTanulokEsHallgatokSzamaTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgTanulokEsHallgatokSzamaTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgTanulokEsHallgatokSzamaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgTanulokEsHallgatokSzamaQuery

#### StgTelepules.php

<?php

namespace Base;

use \StgTelepulesQuery as ChildStgTelepulesQuery;

use \Exception;

use \PDO;

use Map\StgTelepulesTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgTelepules implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgTelepulesTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the ksh\_kod field.

\* @var string

\*/

protected $ksh\_kod;

/\*\*

\* The value for the jogallas field.

\* @var string

\*/

protected $jogallas;

/\*\*

\* The value for the megye field.

\* @var string

\*/

protected $megye;

/\*\*

\* The value for the kisterseg\_kod field.

\* @var string

\*/

protected $kisterseg\_kod;

/\*\*

\* The value for the kisterseg\_nev field.

\* @var string

\*/

protected $kisterseg\_nev;

/\*\*

\* The value for the kisterseg\_szekhelye field.

\* @var string

\*/

protected $kisterseg\_szekhelye;

/\*\*

\* The value for the eov\_x field.

\* @var string

\*/

protected $eov\_x;

/\*\*

\* The value for the eov\_y field.

\* @var string

\*/

protected $eov\_y;

/\*\*

\* The value for the wgs84\_lat field.

\* @var string

\*/

protected $wgs84\_lat;

/\*\*

\* The value for the wgs84\_lob field.

\* @var string

\*/

protected $wgs84\_lob;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgTelepules object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgTelepules</code> instance. If

\* <code>obj</code> is an instance of <code>StgTelepules</code>, delegates to

\* <code>equals(StgTelepules)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgTelepules The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [ksh\_kod] column value.

\*

\* @return string

\*/

public function getKshKod()

{

return $this->ksh\_kod;

}

/\*\*

\* Get the [jogallas] column value.

\*

\* @return string

\*/

public function getJogallas()

{

return $this->jogallas;

}

/\*\*

\* Get the [megye] column value.

\*

\* @return string

\*/

public function getMegye()

{

return $this->megye;

}

/\*\*

\* Get the [kisterseg\_kod] column value.

\*

\* @return string

\*/

public function getKistersegKod()

{

return $this->kisterseg\_kod;

}

/\*\*

\* Get the [kisterseg\_nev] column value.

\*

\* @return string

\*/

public function getKistersegNev()

{

return $this->kisterseg\_nev;

}

/\*\*

\* Get the [kisterseg\_szekhelye] column value.

\*

\* @return string

\*/

public function getKistersegSzekhelye()

{

return $this->kisterseg\_szekhelye;

}

/\*\*

\* Get the [eov\_x] column value.

\*

\* @return string

\*/

public function getEovX()

{

return $this->eov\_x;

}

/\*\*

\* Get the [eov\_y] column value.

\*

\* @return string

\*/

public function getEovY()

{

return $this->eov\_y;

}

/\*\*

\* Get the [wgs84\_lat] column value.

\*

\* @return string

\*/

public function getWgs84Lat()

{

return $this->wgs84\_lat;

}

/\*\*

\* Get the [wgs84\_lob] column value.

\*

\* @return string

\*/

public function getWgs84Lob()

{

return $this->wgs84\_lob;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgTelepulesTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgTelepulesTableMap::translateFieldName('KshKod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->ksh\_kod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgTelepulesTableMap::translateFieldName('Jogallas', TableMap::TYPE\_PHPNAME, $indexType)];

$this->jogallas = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgTelepulesTableMap::translateFieldName('Megye', TableMap::TYPE\_PHPNAME, $indexType)];

$this->megye = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgTelepulesTableMap::translateFieldName('KistersegKod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kisterseg\_kod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 5 + $startcol : StgTelepulesTableMap::translateFieldName('KistersegNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kisterseg\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 6 + $startcol : StgTelepulesTableMap::translateFieldName('KistersegSzekhelye', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kisterseg\_szekhelye = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 7 + $startcol : StgTelepulesTableMap::translateFieldName('EovX', TableMap::TYPE\_PHPNAME, $indexType)];

$this->eov\_x = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 8 + $startcol : StgTelepulesTableMap::translateFieldName('EovY', TableMap::TYPE\_PHPNAME, $indexType)];

$this->eov\_y = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 9 + $startcol : StgTelepulesTableMap::translateFieldName('Wgs84Lat', TableMap::TYPE\_PHPNAME, $indexType)];

$this->wgs84\_lat = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 10 + $startcol : StgTelepulesTableMap::translateFieldName('Wgs84Lob', TableMap::TYPE\_PHPNAME, $indexType)];

$this->wgs84\_lob = (null !== $col) ? (string) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 11; // 11 = StgTelepulesTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgTelepules'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgTelepules The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgTelepulesTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [ksh\_kod] column.

\*

\* @param string $v new value

\* @return $this|\StgTelepules The current object (for fluent API support)

\*/

public function setKshKod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->ksh\_kod !== $v) {

$this->ksh\_kod = $v;

$this->modifiedColumns[StgTelepulesTableMap::COL\_KSH\_KOD] = true;

}

return $this;

} // setKshKod()

/\*\*

\* Set the value of [jogallas] column.

\*

\* @param string $v new value

\* @return $this|\StgTelepules The current object (for fluent API support)

\*/

public function setJogallas($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->jogallas !== $v) {

$this->jogallas = $v;

$this->modifiedColumns[StgTelepulesTableMap::COL\_JOGALLAS] = true;

}

return $this;

} // setJogallas()

/\*\*

\* Set the value of [megye] column.

\*

\* @param string $v new value

\* @return $this|\StgTelepules The current object (for fluent API support)

\*/

public function setMegye($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->megye !== $v) {

$this->megye = $v;

$this->modifiedColumns[StgTelepulesTableMap::COL\_MEGYE] = true;

}

return $this;

} // setMegye()

/\*\*

\* Set the value of [kisterseg\_kod] column.

\*

\* @param string $v new value

\* @return $this|\StgTelepules The current object (for fluent API support)

\*/

public function setKistersegKod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->kisterseg\_kod !== $v) {

$this->kisterseg\_kod = $v;

$this->modifiedColumns[StgTelepulesTableMap::COL\_KISTERSEG\_KOD] = true;

}

return $this;

} // setKistersegKod()

/\*\*

\* Set the value of [kisterseg\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgTelepules The current object (for fluent API support)

\*/

public function setKistersegNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->kisterseg\_nev !== $v) {

$this->kisterseg\_nev = $v;

$this->modifiedColumns[StgTelepulesTableMap::COL\_KISTERSEG\_NEV] = true;

}

return $this;

} // setKistersegNev()

/\*\*

\* Set the value of [kisterseg\_szekhelye] column.

\*

\* @param string $v new value

\* @return $this|\StgTelepules The current object (for fluent API support)

\*/

public function setKistersegSzekhelye($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->kisterseg\_szekhelye !== $v) {

$this->kisterseg\_szekhelye = $v;

$this->modifiedColumns[StgTelepulesTableMap::COL\_KISTERSEG\_SZEKHELYE] = true;

}

return $this;

} // setKistersegSzekhelye()

/\*\*

\* Set the value of [eov\_x] column.

\*

\* @param string $v new value

\* @return $this|\StgTelepules The current object (for fluent API support)

\*/

public function setEovX($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->eov\_x !== $v) {

$this->eov\_x = $v;

$this->modifiedColumns[StgTelepulesTableMap::COL\_EOV\_X] = true;

}

return $this;

} // setEovX()

/\*\*

\* Set the value of [eov\_y] column.

\*

\* @param string $v new value

\* @return $this|\StgTelepules The current object (for fluent API support)

\*/

public function setEovY($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->eov\_y !== $v) {

$this->eov\_y = $v;

$this->modifiedColumns[StgTelepulesTableMap::COL\_EOV\_Y] = true;

}

return $this;

} // setEovY()

/\*\*

\* Set the value of [wgs84\_lat] column.

\*

\* @param string $v new value

\* @return $this|\StgTelepules The current object (for fluent API support)

\*/

public function setWgs84Lat($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->wgs84\_lat !== $v) {

$this->wgs84\_lat = $v;

$this->modifiedColumns[StgTelepulesTableMap::COL\_WGS84\_LAT] = true;

}

return $this;

} // setWgs84Lat()

/\*\*

\* Set the value of [wgs84\_lob] column.

\*

\* @param string $v new value

\* @return $this|\StgTelepules The current object (for fluent API support)

\*/

public function setWgs84Lob($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->wgs84\_lob !== $v) {

$this->wgs84\_lob = $v;

$this->modifiedColumns[StgTelepulesTableMap::COL\_WGS84\_LOB] = true;

}

return $this;

} // setWgs84Lob()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgTelepulesTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgTelepulesQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgTelepules::setDeleted()

\* @see StgTelepules::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTelepulesTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgTelepulesQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTelepulesTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgTelepulesTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgTelepulesTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_KSH\_KOD)) {

$modifiedColumns[':p' . $index++] = 'KSH\_KOD';

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_JOGALLAS)) {

$modifiedColumns[':p' . $index++] = 'JOGALLAS';

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_MEGYE)) {

$modifiedColumns[':p' . $index++] = 'MEGYE';

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_KISTERSEG\_KOD)) {

$modifiedColumns[':p' . $index++] = 'KISTERSEG\_KOD';

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_KISTERSEG\_NEV)) {

$modifiedColumns[':p' . $index++] = 'KISTERSEG\_NEV';

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_KISTERSEG\_SZEKHELYE)) {

$modifiedColumns[':p' . $index++] = 'KISTERSEG\_SZEKHELYE';

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_EOV\_X)) {

$modifiedColumns[':p' . $index++] = 'EOV\_X';

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_EOV\_Y)) {

$modifiedColumns[':p' . $index++] = 'EOV\_Y';

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_WGS84\_LAT)) {

$modifiedColumns[':p' . $index++] = 'WGS84\_LAT';

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_WGS84\_LOB)) {

$modifiedColumns[':p' . $index++] = 'WGS84\_LOB';

}

$sql = sprintf(

'INSERT INTO stg\_telepules (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'KSH\_KOD':

$stmt->bindValue($identifier, $this->ksh\_kod, PDO::PARAM\_STR);

break;

case 'JOGALLAS':

$stmt->bindValue($identifier, $this->jogallas, PDO::PARAM\_STR);

break;

case 'MEGYE':

$stmt->bindValue($identifier, $this->megye, PDO::PARAM\_STR);

break;

case 'KISTERSEG\_KOD':

$stmt->bindValue($identifier, $this->kisterseg\_kod, PDO::PARAM\_STR);

break;

case 'KISTERSEG\_NEV':

$stmt->bindValue($identifier, $this->kisterseg\_nev, PDO::PARAM\_STR);

break;

case 'KISTERSEG\_SZEKHELYE':

$stmt->bindValue($identifier, $this->kisterseg\_szekhelye, PDO::PARAM\_STR);

break;

case 'EOV\_X':

$stmt->bindValue($identifier, $this->eov\_x, PDO::PARAM\_STR);

break;

case 'EOV\_Y':

$stmt->bindValue($identifier, $this->eov\_y, PDO::PARAM\_STR);

break;

case 'WGS84\_LAT':

$stmt->bindValue($identifier, $this->wgs84\_lat, PDO::PARAM\_STR);

break;

case 'WGS84\_LOB':

$stmt->bindValue($identifier, $this->wgs84\_lob, PDO::PARAM\_STR);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgTelepulesTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getKshKod();

break;

case 2:

return $this->getJogallas();

break;

case 3:

return $this->getMegye();

break;

case 4:

return $this->getKistersegKod();

break;

case 5:

return $this->getKistersegNev();

break;

case 6:

return $this->getKistersegSzekhelye();

break;

case 7:

return $this->getEovX();

break;

case 8:

return $this->getEovY();

break;

case 9:

return $this->getWgs84Lat();

break;

case 10:

return $this->getWgs84Lob();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgTelepules'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgTelepules'][$this->getPrimaryKey()] = true;

$keys = StgTelepulesTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getKshKod(),

$keys[2] => $this->getJogallas(),

$keys[3] => $this->getMegye(),

$keys[4] => $this->getKistersegKod(),

$keys[5] => $this->getKistersegNev(),

$keys[6] => $this->getKistersegSzekhelye(),

$keys[7] => $this->getEovX(),

$keys[8] => $this->getEovY(),

$keys[9] => $this->getWgs84Lat(),

$keys[10] => $this->getWgs84Lob(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgTelepules

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgTelepulesTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgTelepules

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setKshKod($value);

break;

case 2:

$this->setJogallas($value);

break;

case 3:

$this->setMegye($value);

break;

case 4:

$this->setKistersegKod($value);

break;

case 5:

$this->setKistersegNev($value);

break;

case 6:

$this->setKistersegSzekhelye($value);

break;

case 7:

$this->setEovX($value);

break;

case 8:

$this->setEovY($value);

break;

case 9:

$this->setWgs84Lat($value);

break;

case 10:

$this->setWgs84Lob($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgTelepulesTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setKshKod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setJogallas($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setMegye($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setKistersegKod($arr[$keys[4]]);

}

if (array\_key\_exists($keys[5], $arr)) {

$this->setKistersegNev($arr[$keys[5]]);

}

if (array\_key\_exists($keys[6], $arr)) {

$this->setKistersegSzekhelye($arr[$keys[6]]);

}

if (array\_key\_exists($keys[7], $arr)) {

$this->setEovX($arr[$keys[7]]);

}

if (array\_key\_exists($keys[8], $arr)) {

$this->setEovY($arr[$keys[8]]);

}

if (array\_key\_exists($keys[9], $arr)) {

$this->setWgs84Lat($arr[$keys[9]]);

}

if (array\_key\_exists($keys[10], $arr)) {

$this->setWgs84Lob($arr[$keys[10]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgTelepules The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgTelepulesTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgTelepulesTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgTelepulesTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_KSH\_KOD)) {

$criteria->add(StgTelepulesTableMap::COL\_KSH\_KOD, $this->ksh\_kod);

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_JOGALLAS)) {

$criteria->add(StgTelepulesTableMap::COL\_JOGALLAS, $this->jogallas);

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_MEGYE)) {

$criteria->add(StgTelepulesTableMap::COL\_MEGYE, $this->megye);

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_KISTERSEG\_KOD)) {

$criteria->add(StgTelepulesTableMap::COL\_KISTERSEG\_KOD, $this->kisterseg\_kod);

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_KISTERSEG\_NEV)) {

$criteria->add(StgTelepulesTableMap::COL\_KISTERSEG\_NEV, $this->kisterseg\_nev);

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_KISTERSEG\_SZEKHELYE)) {

$criteria->add(StgTelepulesTableMap::COL\_KISTERSEG\_SZEKHELYE, $this->kisterseg\_szekhelye);

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_EOV\_X)) {

$criteria->add(StgTelepulesTableMap::COL\_EOV\_X, $this->eov\_x);

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_EOV\_Y)) {

$criteria->add(StgTelepulesTableMap::COL\_EOV\_Y, $this->eov\_y);

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_WGS84\_LAT)) {

$criteria->add(StgTelepulesTableMap::COL\_WGS84\_LAT, $this->wgs84\_lat);

}

if ($this->isColumnModified(StgTelepulesTableMap::COL\_WGS84\_LOB)) {

$criteria->add(StgTelepulesTableMap::COL\_WGS84\_LOB, $this->wgs84\_lob);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

$criteria = new Criteria(StgTelepulesTableMap::DATABASE\_NAME);

$criteria->add(StgTelepulesTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = null !== $this->getTelepulesNev();

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns the primary key for this object (row).

\* @return string

\*/

public function getPrimaryKey()

{

return $this->getTelepulesNev();

}

/\*\*

\* Generic method to set the primary key (telepules\_nev column).

\*

\* @param string $key Primary key.

\* @return void

\*/

public function setPrimaryKey($key)

{

$this->setTelepulesNev($key);

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return null === $this->getTelepulesNev();

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgTelepules (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setKshKod($this->getKshKod());

$copyObj->setJogallas($this->getJogallas());

$copyObj->setMegye($this->getMegye());

$copyObj->setKistersegKod($this->getKistersegKod());

$copyObj->setKistersegNev($this->getKistersegNev());

$copyObj->setKistersegSzekhelye($this->getKistersegSzekhelye());

$copyObj->setEovX($this->getEovX());

$copyObj->setEovY($this->getEovY());

$copyObj->setWgs84Lat($this->getWgs84Lat());

$copyObj->setWgs84Lob($this->getWgs84Lob());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgTelepules Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->ksh\_kod = null;

$this->jogallas = null;

$this->megye = null;

$this->kisterseg\_kod = null;

$this->kisterseg\_nev = null;

$this->kisterseg\_szekhelye = null;

$this->eov\_x = null;

$this->eov\_y = null;

$this->wgs84\_lat = null;

$this->wgs84\_lob = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgTelepulesTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgTelepulesKoordinata.php

<?php

namespace Base;

use \StgTelepulesKoordinataQuery as ChildStgTelepulesKoordinataQuery;

use \Exception;

use \PDO;

use Map\StgTelepulesKoordinataTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgTelepulesKoordinata implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgTelepulesKoordinataTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the eov\_x field.

\* @var string

\*/

protected $eov\_x;

/\*\*

\* The value for the eov\_y field.

\* @var string

\*/

protected $eov\_y;

/\*\*

\* The value for the wgs84\_lat field.

\* @var string

\*/

protected $wgs84\_lat;

/\*\*

\* The value for the wgs84\_lob field.

\* @var string

\*/

protected $wgs84\_lob;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgTelepulesKoordinata object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgTelepulesKoordinata</code> instance. If

\* <code>obj</code> is an instance of <code>StgTelepulesKoordinata</code>, delegates to

\* <code>equals(StgTelepulesKoordinata)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgTelepulesKoordinata The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [eov\_x] column value.

\*

\* @return string

\*/

public function getEovX()

{

return $this->eov\_x;

}

/\*\*

\* Get the [eov\_y] column value.

\*

\* @return string

\*/

public function getEovY()

{

return $this->eov\_y;

}

/\*\*

\* Get the [wgs84\_lat] column value.

\*

\* @return string

\*/

public function getWgs84Lat()

{

return $this->wgs84\_lat;

}

/\*\*

\* Get the [wgs84\_lob] column value.

\*

\* @return string

\*/

public function getWgs84Lob()

{

return $this->wgs84\_lob;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgTelepulesKoordinataTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgTelepulesKoordinataTableMap::translateFieldName('EovX', TableMap::TYPE\_PHPNAME, $indexType)];

$this->eov\_x = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgTelepulesKoordinataTableMap::translateFieldName('EovY', TableMap::TYPE\_PHPNAME, $indexType)];

$this->eov\_y = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgTelepulesKoordinataTableMap::translateFieldName('Wgs84Lat', TableMap::TYPE\_PHPNAME, $indexType)];

$this->wgs84\_lat = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgTelepulesKoordinataTableMap::translateFieldName('Wgs84Lob', TableMap::TYPE\_PHPNAME, $indexType)];

$this->wgs84\_lob = (null !== $col) ? (string) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 5; // 5 = StgTelepulesKoordinataTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgTelepulesKoordinata'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgTelepulesKoordinata The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgTelepulesKoordinataTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [eov\_x] column.

\*

\* @param string $v new value

\* @return $this|\StgTelepulesKoordinata The current object (for fluent API support)

\*/

public function setEovX($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->eov\_x !== $v) {

$this->eov\_x = $v;

$this->modifiedColumns[StgTelepulesKoordinataTableMap::COL\_EOV\_X] = true;

}

return $this;

} // setEovX()

/\*\*

\* Set the value of [eov\_y] column.

\*

\* @param string $v new value

\* @return $this|\StgTelepulesKoordinata The current object (for fluent API support)

\*/

public function setEovY($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->eov\_y !== $v) {

$this->eov\_y = $v;

$this->modifiedColumns[StgTelepulesKoordinataTableMap::COL\_EOV\_Y] = true;

}

return $this;

} // setEovY()

/\*\*

\* Set the value of [wgs84\_lat] column.

\*

\* @param string $v new value

\* @return $this|\StgTelepulesKoordinata The current object (for fluent API support)

\*/

public function setWgs84Lat($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->wgs84\_lat !== $v) {

$this->wgs84\_lat = $v;

$this->modifiedColumns[StgTelepulesKoordinataTableMap::COL\_WGS84\_LAT] = true;

}

return $this;

} // setWgs84Lat()

/\*\*

\* Set the value of [wgs84\_lob] column.

\*

\* @param string $v new value

\* @return $this|\StgTelepulesKoordinata The current object (for fluent API support)

\*/

public function setWgs84Lob($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->wgs84\_lob !== $v) {

$this->wgs84\_lob = $v;

$this->modifiedColumns[StgTelepulesKoordinataTableMap::COL\_WGS84\_LOB] = true;

}

return $this;

} // setWgs84Lob()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgTelepulesKoordinataTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgTelepulesKoordinataQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgTelepulesKoordinata::setDeleted()

\* @see StgTelepulesKoordinata::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTelepulesKoordinataTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgTelepulesKoordinataQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTelepulesKoordinataTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgTelepulesKoordinataTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgTelepulesKoordinataTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgTelepulesKoordinataTableMap::COL\_EOV\_X)) {

$modifiedColumns[':p' . $index++] = 'EOV\_X';

}

if ($this->isColumnModified(StgTelepulesKoordinataTableMap::COL\_EOV\_Y)) {

$modifiedColumns[':p' . $index++] = 'EOV\_Y';

}

if ($this->isColumnModified(StgTelepulesKoordinataTableMap::COL\_WGS84\_LAT)) {

$modifiedColumns[':p' . $index++] = 'WGS84\_LAT';

}

if ($this->isColumnModified(StgTelepulesKoordinataTableMap::COL\_WGS84\_LOB)) {

$modifiedColumns[':p' . $index++] = 'WGS84\_LOB';

}

$sql = sprintf(

'INSERT INTO stg\_telepules\_koordinata (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'EOV\_X':

$stmt->bindValue($identifier, $this->eov\_x, PDO::PARAM\_STR);

break;

case 'EOV\_Y':

$stmt->bindValue($identifier, $this->eov\_y, PDO::PARAM\_STR);

break;

case 'WGS84\_LAT':

$stmt->bindValue($identifier, $this->wgs84\_lat, PDO::PARAM\_STR);

break;

case 'WGS84\_LOB':

$stmt->bindValue($identifier, $this->wgs84\_lob, PDO::PARAM\_STR);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgTelepulesKoordinataTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getEovX();

break;

case 2:

return $this->getEovY();

break;

case 3:

return $this->getWgs84Lat();

break;

case 4:

return $this->getWgs84Lob();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgTelepulesKoordinata'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgTelepulesKoordinata'][$this->getPrimaryKey()] = true;

$keys = StgTelepulesKoordinataTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getEovX(),

$keys[2] => $this->getEovY(),

$keys[3] => $this->getWgs84Lat(),

$keys[4] => $this->getWgs84Lob(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgTelepulesKoordinata

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgTelepulesKoordinataTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgTelepulesKoordinata

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setEovX($value);

break;

case 2:

$this->setEovY($value);

break;

case 3:

$this->setWgs84Lat($value);

break;

case 4:

$this->setWgs84Lob($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgTelepulesKoordinataTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setEovX($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setEovY($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setWgs84Lat($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setWgs84Lob($arr[$keys[4]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgTelepulesKoordinata The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgTelepulesKoordinataTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgTelepulesKoordinataTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgTelepulesKoordinataTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgTelepulesKoordinataTableMap::COL\_EOV\_X)) {

$criteria->add(StgTelepulesKoordinataTableMap::COL\_EOV\_X, $this->eov\_x);

}

if ($this->isColumnModified(StgTelepulesKoordinataTableMap::COL\_EOV\_Y)) {

$criteria->add(StgTelepulesKoordinataTableMap::COL\_EOV\_Y, $this->eov\_y);

}

if ($this->isColumnModified(StgTelepulesKoordinataTableMap::COL\_WGS84\_LAT)) {

$criteria->add(StgTelepulesKoordinataTableMap::COL\_WGS84\_LAT, $this->wgs84\_lat);

}

if ($this->isColumnModified(StgTelepulesKoordinataTableMap::COL\_WGS84\_LOB)) {

$criteria->add(StgTelepulesKoordinataTableMap::COL\_WGS84\_LOB, $this->wgs84\_lob);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

$criteria = new Criteria(StgTelepulesKoordinataTableMap::DATABASE\_NAME);

$criteria->add(StgTelepulesKoordinataTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = null !== $this->getTelepulesNev();

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns the primary key for this object (row).

\* @return string

\*/

public function getPrimaryKey()

{

return $this->getTelepulesNev();

}

/\*\*

\* Generic method to set the primary key (telepules\_nev column).

\*

\* @param string $key Primary key.

\* @return void

\*/

public function setPrimaryKey($key)

{

$this->setTelepulesNev($key);

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return null === $this->getTelepulesNev();

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgTelepulesKoordinata (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setEovX($this->getEovX());

$copyObj->setEovY($this->getEovY());

$copyObj->setWgs84Lat($this->getWgs84Lat());

$copyObj->setWgs84Lob($this->getWgs84Lob());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgTelepulesKoordinata Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->eov\_x = null;

$this->eov\_y = null;

$this->wgs84\_lat = null;

$this->wgs84\_lob = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgTelepulesKoordinataTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgTelepulesKoordinataQuery.php

<?php

namespace Base;

use \StgTelepulesKoordinata as ChildStgTelepulesKoordinata;

use \StgTelepulesKoordinataQuery as ChildStgTelepulesKoordinataQuery;

use \Exception;

use \PDO;

use Map\StgTelepulesKoordinataTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_telepules\_koordinata' table.

\*

\*

\*

\* @method ChildStgTelepulesKoordinataQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgTelepulesKoordinataQuery orderByEovX($order = Criteria::ASC) Order by the EOV\_X column

\* @method ChildStgTelepulesKoordinataQuery orderByEovY($order = Criteria::ASC) Order by the EOV\_Y column

\* @method ChildStgTelepulesKoordinataQuery orderByWgs84Lat($order = Criteria::ASC) Order by the WGS84\_lat column

\* @method ChildStgTelepulesKoordinataQuery orderByWgs84Lob($order = Criteria::ASC) Order by the WGS84\_lob column

\*

\* @method ChildStgTelepulesKoordinataQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgTelepulesKoordinataQuery groupByEovX() Group by the EOV\_X column

\* @method ChildStgTelepulesKoordinataQuery groupByEovY() Group by the EOV\_Y column

\* @method ChildStgTelepulesKoordinataQuery groupByWgs84Lat() Group by the WGS84\_lat column

\* @method ChildStgTelepulesKoordinataQuery groupByWgs84Lob() Group by the WGS84\_lob column

\*

\* @method ChildStgTelepulesKoordinataQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgTelepulesKoordinataQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgTelepulesKoordinataQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgTelepulesKoordinata findOne(ConnectionInterface $con = null) Return the first ChildStgTelepulesKoordinata matching the query

\* @method ChildStgTelepulesKoordinata findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgTelepulesKoordinata matching the query, or a new ChildStgTelepulesKoordinata object populated from the query conditions when no match is found

\*

\* @method ChildStgTelepulesKoordinata findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgTelepulesKoordinata filtered by the telepules\_nev column

\* @method ChildStgTelepulesKoordinata findOneByEovX(string $EOV\_X) Return the first ChildStgTelepulesKoordinata filtered by the EOV\_X column

\* @method ChildStgTelepulesKoordinata findOneByEovY(string $EOV\_Y) Return the first ChildStgTelepulesKoordinata filtered by the EOV\_Y column

\* @method ChildStgTelepulesKoordinata findOneByWgs84Lat(string $WGS84\_lat) Return the first ChildStgTelepulesKoordinata filtered by the WGS84\_lat column

\* @method ChildStgTelepulesKoordinata findOneByWgs84Lob(string $WGS84\_lob) Return the first ChildStgTelepulesKoordinata filtered by the WGS84\_lob column

\*

\* @method ChildStgTelepulesKoordinata[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgTelepulesKoordinata objects based on current ModelCriteria

\* @method ChildStgTelepulesKoordinata[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgTelepulesKoordinata objects filtered by the telepules\_nev column

\* @method ChildStgTelepulesKoordinata[]|ObjectCollection findByEovX(string $EOV\_X) Return ChildStgTelepulesKoordinata objects filtered by the EOV\_X column

\* @method ChildStgTelepulesKoordinata[]|ObjectCollection findByEovY(string $EOV\_Y) Return ChildStgTelepulesKoordinata objects filtered by the EOV\_Y column

\* @method ChildStgTelepulesKoordinata[]|ObjectCollection findByWgs84Lat(string $WGS84\_lat) Return ChildStgTelepulesKoordinata objects filtered by the WGS84\_lat column

\* @method ChildStgTelepulesKoordinata[]|ObjectCollection findByWgs84Lob(string $WGS84\_lob) Return ChildStgTelepulesKoordinata objects filtered by the WGS84\_lob column

\* @method ChildStgTelepulesKoordinata[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgTelepulesKoordinataQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgTelepulesKoordinataQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgTelepulesKoordinata', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgTelepulesKoordinataQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgTelepulesKoordinataQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgTelepulesKoordinataQuery) {

return $criteria;

}

$query = new ChildStgTelepulesKoordinataQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgTelepulesKoordinata|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

if ($key === null) {

return null;

}

if ((null !== ($obj = StgTelepulesKoordinataTableMap::getInstanceFromPool((string) $key))) && !$this->formatter) {

// the object is already in the instance pool

return $obj;

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgTelepulesKoordinataTableMap::DATABASE\_NAME);

}

$this->basePreSelect($con);

if ($this->formatter || $this->modelAlias || $this->with || $this->select

|| $this->selectColumns || $this->asColumns || $this->selectModifiers

|| $this->map || $this->having || $this->joins) {

return $this->findPkComplex($key, $con);

} else {

return $this->findPkSimple($key, $con);

}

}

/\*\*

\* Find object by primary key using raw SQL to go fast.

\* Bypass doSelect() and the object formatter by using generated code.

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con A connection object

\*

\* @return ChildStgTelepulesKoordinata A model object, or null if the key is not found

\*/

protected function findPkSimple($key, ConnectionInterface $con)

{

$sql = 'SELECT TELEPULES\_NEV, EOV\_X, EOV\_Y, WGS84\_LAT, WGS84\_LOB FROM stg\_telepules\_koordinata WHERE TELEPULES\_NEV = :p0';

try {

$stmt = $con->prepare($sql);

$stmt->bindValue(':p0', $key, PDO::PARAM\_STR);

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute SELECT statement [%s]', $sql), 0, $e);

}

$obj = null;

if ($row = $stmt->fetch(\PDO::FETCH\_NUM)) {

/\*\* @var ChildStgTelepulesKoordinata $obj \*/

$obj = new ChildStgTelepulesKoordinata();

$obj->hydrate($row);

StgTelepulesKoordinataTableMap::addInstanceToPool($obj, (string) $key);

}

$stmt->closeCursor();

return $obj;

}

/\*\*

\* Find object by primary key.

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con A connection object

\*

\* @return ChildStgTelepulesKoordinata|array|mixed the result, formatted by the current formatter

\*/

protected function findPkComplex($key, ConnectionInterface $con)

{

// As the query uses a PK condition, no limit(1) is necessary.

$criteria = $this->isKeepQuery() ? clone $this : $this;

$dataFetcher = $criteria

->filterByPrimaryKey($key)

->doSelect($con);

return $criteria->getFormatter()->init($criteria)->formatOne($dataFetcher);

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(12, 56, 832), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getReadConnection($this->getDbName());

}

$this->basePreSelect($con);

$criteria = $this->isKeepQuery() ? clone $this : $this;

$dataFetcher = $criteria

->filterByPrimaryKeys($keys)

->doSelect($con);

return $criteria->getFormatter()->init($criteria)->format($dataFetcher);

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgTelepulesKoordinataQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

return $this->addUsingAlias(StgTelepulesKoordinataTableMap::COL\_TELEPULES\_NEV, $key, Criteria::EQUAL);

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgTelepulesKoordinataQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

return $this->addUsingAlias(StgTelepulesKoordinataTableMap::COL\_TELEPULES\_NEV, $keys, Criteria::IN);

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTelepulesKoordinataQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTelepulesKoordinataTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the EOV\_X column

\*

\* Example usage:

\* <code>

\* $query->filterByEovX('fooValue'); // WHERE EOV\_X = 'fooValue'

\* $query->filterByEovX('%fooValue%'); // WHERE EOV\_X LIKE '%fooValue%'

\* </code>

\*

\* @param string $eovX The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTelepulesKoordinataQuery The current query, for fluid interface

\*/

public function filterByEovX($eovX = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($eovX)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $eovX)) {

$eovX = str\_replace('\*', '%', $eovX);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTelepulesKoordinataTableMap::COL\_EOV\_X, $eovX, $comparison);

}

/\*\*

\* Filter the query on the EOV\_Y column

\*

\* Example usage:

\* <code>

\* $query->filterByEovY('fooValue'); // WHERE EOV\_Y = 'fooValue'

\* $query->filterByEovY('%fooValue%'); // WHERE EOV\_Y LIKE '%fooValue%'

\* </code>

\*

\* @param string $eovY The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTelepulesKoordinataQuery The current query, for fluid interface

\*/

public function filterByEovY($eovY = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($eovY)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $eovY)) {

$eovY = str\_replace('\*', '%', $eovY);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTelepulesKoordinataTableMap::COL\_EOV\_Y, $eovY, $comparison);

}

/\*\*

\* Filter the query on the WGS84\_lat column

\*

\* Example usage:

\* <code>

\* $query->filterByWgs84Lat('fooValue'); // WHERE WGS84\_lat = 'fooValue'

\* $query->filterByWgs84Lat('%fooValue%'); // WHERE WGS84\_lat LIKE '%fooValue%'

\* </code>

\*

\* @param string $wgs84Lat The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTelepulesKoordinataQuery The current query, for fluid interface

\*/

public function filterByWgs84Lat($wgs84Lat = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($wgs84Lat)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $wgs84Lat)) {

$wgs84Lat = str\_replace('\*', '%', $wgs84Lat);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTelepulesKoordinataTableMap::COL\_WGS84\_LAT, $wgs84Lat, $comparison);

}

/\*\*

\* Filter the query on the WGS84\_lob column

\*

\* Example usage:

\* <code>

\* $query->filterByWgs84Lob('fooValue'); // WHERE WGS84\_lob = 'fooValue'

\* $query->filterByWgs84Lob('%fooValue%'); // WHERE WGS84\_lob LIKE '%fooValue%'

\* </code>

\*

\* @param string $wgs84Lob The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTelepulesKoordinataQuery The current query, for fluid interface

\*/

public function filterByWgs84Lob($wgs84Lob = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($wgs84Lob)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $wgs84Lob)) {

$wgs84Lob = str\_replace('\*', '%', $wgs84Lob);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTelepulesKoordinataTableMap::COL\_WGS84\_LOB, $wgs84Lob, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgTelepulesKoordinata $stgTelepulesKoordinata Object to remove from the list of results

\*

\* @return $this|ChildStgTelepulesKoordinataQuery The current query, for fluid interface

\*/

public function prune($stgTelepulesKoordinata = null)

{

if ($stgTelepulesKoordinata) {

$this->addUsingAlias(StgTelepulesKoordinataTableMap::COL\_TELEPULES\_NEV, $stgTelepulesKoordinata->getTelepulesNev(), Criteria::NOT\_EQUAL);

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_telepules\_koordinata table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTelepulesKoordinataTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgTelepulesKoordinataTableMap::clearInstancePool();

StgTelepulesKoordinataTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTelepulesKoordinataTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgTelepulesKoordinataTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgTelepulesKoordinataTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgTelepulesKoordinataTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgTelepulesKoordinataQuery

#### StgTelepulesQuery.php

<?php

namespace Base;

use \StgTelepules as ChildStgTelepules;

use \StgTelepulesQuery as ChildStgTelepulesQuery;

use \Exception;

use \PDO;

use Map\StgTelepulesTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_telepules' table.

\*

\*

\*

\* @method ChildStgTelepulesQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgTelepulesQuery orderByKshKod($order = Criteria::ASC) Order by the KSH\_kod column

\* @method ChildStgTelepulesQuery orderByJogallas($order = Criteria::ASC) Order by the jogallas column

\* @method ChildStgTelepulesQuery orderByMegye($order = Criteria::ASC) Order by the megye column

\* @method ChildStgTelepulesQuery orderByKistersegKod($order = Criteria::ASC) Order by the kisterseg\_kod column

\* @method ChildStgTelepulesQuery orderByKistersegNev($order = Criteria::ASC) Order by the kisterseg\_nev column

\* @method ChildStgTelepulesQuery orderByKistersegSzekhelye($order = Criteria::ASC) Order by the kisterseg\_szekhelye column

\* @method ChildStgTelepulesQuery orderByEovX($order = Criteria::ASC) Order by the EOV\_X column

\* @method ChildStgTelepulesQuery orderByEovY($order = Criteria::ASC) Order by the EOV\_Y column

\* @method ChildStgTelepulesQuery orderByWgs84Lat($order = Criteria::ASC) Order by the WGS84\_lat column

\* @method ChildStgTelepulesQuery orderByWgs84Lob($order = Criteria::ASC) Order by the WGS84\_lob column

\*

\* @method ChildStgTelepulesQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgTelepulesQuery groupByKshKod() Group by the KSH\_kod column

\* @method ChildStgTelepulesQuery groupByJogallas() Group by the jogallas column

\* @method ChildStgTelepulesQuery groupByMegye() Group by the megye column

\* @method ChildStgTelepulesQuery groupByKistersegKod() Group by the kisterseg\_kod column

\* @method ChildStgTelepulesQuery groupByKistersegNev() Group by the kisterseg\_nev column

\* @method ChildStgTelepulesQuery groupByKistersegSzekhelye() Group by the kisterseg\_szekhelye column

\* @method ChildStgTelepulesQuery groupByEovX() Group by the EOV\_X column

\* @method ChildStgTelepulesQuery groupByEovY() Group by the EOV\_Y column

\* @method ChildStgTelepulesQuery groupByWgs84Lat() Group by the WGS84\_lat column

\* @method ChildStgTelepulesQuery groupByWgs84Lob() Group by the WGS84\_lob column

\*

\* @method ChildStgTelepulesQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgTelepulesQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgTelepulesQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgTelepules findOne(ConnectionInterface $con = null) Return the first ChildStgTelepules matching the query

\* @method ChildStgTelepules findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgTelepules matching the query, or a new ChildStgTelepules object populated from the query conditions when no match is found

\*

\* @method ChildStgTelepules findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgTelepules filtered by the telepules\_nev column

\* @method ChildStgTelepules findOneByKshKod(string $KSH\_kod) Return the first ChildStgTelepules filtered by the KSH\_kod column

\* @method ChildStgTelepules findOneByJogallas(string $jogallas) Return the first ChildStgTelepules filtered by the jogallas column

\* @method ChildStgTelepules findOneByMegye(string $megye) Return the first ChildStgTelepules filtered by the megye column

\* @method ChildStgTelepules findOneByKistersegKod(string $kisterseg\_kod) Return the first ChildStgTelepules filtered by the kisterseg\_kod column

\* @method ChildStgTelepules findOneByKistersegNev(string $kisterseg\_nev) Return the first ChildStgTelepules filtered by the kisterseg\_nev column

\* @method ChildStgTelepules findOneByKistersegSzekhelye(string $kisterseg\_szekhelye) Return the first ChildStgTelepules filtered by the kisterseg\_szekhelye column

\* @method ChildStgTelepules findOneByEovX(string $EOV\_X) Return the first ChildStgTelepules filtered by the EOV\_X column

\* @method ChildStgTelepules findOneByEovY(string $EOV\_Y) Return the first ChildStgTelepules filtered by the EOV\_Y column

\* @method ChildStgTelepules findOneByWgs84Lat(string $WGS84\_lat) Return the first ChildStgTelepules filtered by the WGS84\_lat column

\* @method ChildStgTelepules findOneByWgs84Lob(string $WGS84\_lob) Return the first ChildStgTelepules filtered by the WGS84\_lob column

\*

\* @method ChildStgTelepules[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgTelepules objects based on current ModelCriteria

\* @method ChildStgTelepules[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgTelepules objects filtered by the telepules\_nev column

\* @method ChildStgTelepules[]|ObjectCollection findByKshKod(string $KSH\_kod) Return ChildStgTelepules objects filtered by the KSH\_kod column

\* @method ChildStgTelepules[]|ObjectCollection findByJogallas(string $jogallas) Return ChildStgTelepules objects filtered by the jogallas column

\* @method ChildStgTelepules[]|ObjectCollection findByMegye(string $megye) Return ChildStgTelepules objects filtered by the megye column

\* @method ChildStgTelepules[]|ObjectCollection findByKistersegKod(string $kisterseg\_kod) Return ChildStgTelepules objects filtered by the kisterseg\_kod column

\* @method ChildStgTelepules[]|ObjectCollection findByKistersegNev(string $kisterseg\_nev) Return ChildStgTelepules objects filtered by the kisterseg\_nev column

\* @method ChildStgTelepules[]|ObjectCollection findByKistersegSzekhelye(string $kisterseg\_szekhelye) Return ChildStgTelepules objects filtered by the kisterseg\_szekhelye column

\* @method ChildStgTelepules[]|ObjectCollection findByEovX(string $EOV\_X) Return ChildStgTelepules objects filtered by the EOV\_X column

\* @method ChildStgTelepules[]|ObjectCollection findByEovY(string $EOV\_Y) Return ChildStgTelepules objects filtered by the EOV\_Y column

\* @method ChildStgTelepules[]|ObjectCollection findByWgs84Lat(string $WGS84\_lat) Return ChildStgTelepules objects filtered by the WGS84\_lat column

\* @method ChildStgTelepules[]|ObjectCollection findByWgs84Lob(string $WGS84\_lob) Return ChildStgTelepules objects filtered by the WGS84\_lob column

\* @method ChildStgTelepules[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgTelepulesQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgTelepulesQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgTelepules', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgTelepulesQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgTelepulesQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgTelepulesQuery) {

return $criteria;

}

$query = new ChildStgTelepulesQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgTelepules|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

if ($key === null) {

return null;

}

if ((null !== ($obj = StgTelepulesTableMap::getInstanceFromPool((string) $key))) && !$this->formatter) {

// the object is already in the instance pool

return $obj;

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgTelepulesTableMap::DATABASE\_NAME);

}

$this->basePreSelect($con);

if ($this->formatter || $this->modelAlias || $this->with || $this->select

|| $this->selectColumns || $this->asColumns || $this->selectModifiers

|| $this->map || $this->having || $this->joins) {

return $this->findPkComplex($key, $con);

} else {

return $this->findPkSimple($key, $con);

}

}

/\*\*

\* Find object by primary key using raw SQL to go fast.

\* Bypass doSelect() and the object formatter by using generated code.

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con A connection object

\*

\* @return ChildStgTelepules A model object, or null if the key is not found

\*/

protected function findPkSimple($key, ConnectionInterface $con)

{

$sql = 'SELECT TELEPULES\_NEV, KSH\_KOD, JOGALLAS, MEGYE, KISTERSEG\_KOD, KISTERSEG\_NEV, KISTERSEG\_SZEKHELYE, EOV\_X, EOV\_Y, WGS84\_LAT, WGS84\_LOB FROM stg\_telepules WHERE TELEPULES\_NEV = :p0';

try {

$stmt = $con->prepare($sql);

$stmt->bindValue(':p0', $key, PDO::PARAM\_STR);

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute SELECT statement [%s]', $sql), 0, $e);

}

$obj = null;

if ($row = $stmt->fetch(\PDO::FETCH\_NUM)) {

/\*\* @var ChildStgTelepules $obj \*/

$obj = new ChildStgTelepules();

$obj->hydrate($row);

StgTelepulesTableMap::addInstanceToPool($obj, (string) $key);

}

$stmt->closeCursor();

return $obj;

}

/\*\*

\* Find object by primary key.

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con A connection object

\*

\* @return ChildStgTelepules|array|mixed the result, formatted by the current formatter

\*/

protected function findPkComplex($key, ConnectionInterface $con)

{

// As the query uses a PK condition, no limit(1) is necessary.

$criteria = $this->isKeepQuery() ? clone $this : $this;

$dataFetcher = $criteria

->filterByPrimaryKey($key)

->doSelect($con);

return $criteria->getFormatter()->init($criteria)->formatOne($dataFetcher);

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(12, 56, 832), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getReadConnection($this->getDbName());

}

$this->basePreSelect($con);

$criteria = $this->isKeepQuery() ? clone $this : $this;

$dataFetcher = $criteria

->filterByPrimaryKeys($keys)

->doSelect($con);

return $criteria->getFormatter()->init($criteria)->format($dataFetcher);

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgTelepulesQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

return $this->addUsingAlias(StgTelepulesTableMap::COL\_TELEPULES\_NEV, $key, Criteria::EQUAL);

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgTelepulesQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

return $this->addUsingAlias(StgTelepulesTableMap::COL\_TELEPULES\_NEV, $keys, Criteria::IN);

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTelepulesQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTelepulesTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the KSH\_kod column

\*

\* Example usage:

\* <code>

\* $query->filterByKshKod('fooValue'); // WHERE KSH\_kod = 'fooValue'

\* $query->filterByKshKod('%fooValue%'); // WHERE KSH\_kod LIKE '%fooValue%'

\* </code>

\*

\* @param string $kshKod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTelepulesQuery The current query, for fluid interface

\*/

public function filterByKshKod($kshKod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($kshKod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $kshKod)) {

$kshKod = str\_replace('\*', '%', $kshKod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTelepulesTableMap::COL\_KSH\_KOD, $kshKod, $comparison);

}

/\*\*

\* Filter the query on the jogallas column

\*

\* Example usage:

\* <code>

\* $query->filterByJogallas('fooValue'); // WHERE jogallas = 'fooValue'

\* $query->filterByJogallas('%fooValue%'); // WHERE jogallas LIKE '%fooValue%'

\* </code>

\*

\* @param string $jogallas The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTelepulesQuery The current query, for fluid interface

\*/

public function filterByJogallas($jogallas = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($jogallas)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $jogallas)) {

$jogallas = str\_replace('\*', '%', $jogallas);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTelepulesTableMap::COL\_JOGALLAS, $jogallas, $comparison);

}

/\*\*

\* Filter the query on the megye column

\*

\* Example usage:

\* <code>

\* $query->filterByMegye('fooValue'); // WHERE megye = 'fooValue'

\* $query->filterByMegye('%fooValue%'); // WHERE megye LIKE '%fooValue%'

\* </code>

\*

\* @param string $megye The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTelepulesQuery The current query, for fluid interface

\*/

public function filterByMegye($megye = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($megye)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $megye)) {

$megye = str\_replace('\*', '%', $megye);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTelepulesTableMap::COL\_MEGYE, $megye, $comparison);

}

/\*\*

\* Filter the query on the kisterseg\_kod column

\*

\* Example usage:

\* <code>

\* $query->filterByKistersegKod('fooValue'); // WHERE kisterseg\_kod = 'fooValue'

\* $query->filterByKistersegKod('%fooValue%'); // WHERE kisterseg\_kod LIKE '%fooValue%'

\* </code>

\*

\* @param string $kistersegKod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTelepulesQuery The current query, for fluid interface

\*/

public function filterByKistersegKod($kistersegKod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($kistersegKod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $kistersegKod)) {

$kistersegKod = str\_replace('\*', '%', $kistersegKod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTelepulesTableMap::COL\_KISTERSEG\_KOD, $kistersegKod, $comparison);

}

/\*\*

\* Filter the query on the kisterseg\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByKistersegNev('fooValue'); // WHERE kisterseg\_nev = 'fooValue'

\* $query->filterByKistersegNev('%fooValue%'); // WHERE kisterseg\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $kistersegNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTelepulesQuery The current query, for fluid interface

\*/

public function filterByKistersegNev($kistersegNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($kistersegNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $kistersegNev)) {

$kistersegNev = str\_replace('\*', '%', $kistersegNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTelepulesTableMap::COL\_KISTERSEG\_NEV, $kistersegNev, $comparison);

}

/\*\*

\* Filter the query on the kisterseg\_szekhelye column

\*

\* Example usage:

\* <code>

\* $query->filterByKistersegSzekhelye('fooValue'); // WHERE kisterseg\_szekhelye = 'fooValue'

\* $query->filterByKistersegSzekhelye('%fooValue%'); // WHERE kisterseg\_szekhelye LIKE '%fooValue%'

\* </code>

\*

\* @param string $kistersegSzekhelye The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTelepulesQuery The current query, for fluid interface

\*/

public function filterByKistersegSzekhelye($kistersegSzekhelye = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($kistersegSzekhelye)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $kistersegSzekhelye)) {

$kistersegSzekhelye = str\_replace('\*', '%', $kistersegSzekhelye);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTelepulesTableMap::COL\_KISTERSEG\_SZEKHELYE, $kistersegSzekhelye, $comparison);

}

/\*\*

\* Filter the query on the EOV\_X column

\*

\* Example usage:

\* <code>

\* $query->filterByEovX('fooValue'); // WHERE EOV\_X = 'fooValue'

\* $query->filterByEovX('%fooValue%'); // WHERE EOV\_X LIKE '%fooValue%'

\* </code>

\*

\* @param string $eovX The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTelepulesQuery The current query, for fluid interface

\*/

public function filterByEovX($eovX = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($eovX)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $eovX)) {

$eovX = str\_replace('\*', '%', $eovX);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTelepulesTableMap::COL\_EOV\_X, $eovX, $comparison);

}

/\*\*

\* Filter the query on the EOV\_Y column

\*

\* Example usage:

\* <code>

\* $query->filterByEovY('fooValue'); // WHERE EOV\_Y = 'fooValue'

\* $query->filterByEovY('%fooValue%'); // WHERE EOV\_Y LIKE '%fooValue%'

\* </code>

\*

\* @param string $eovY The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTelepulesQuery The current query, for fluid interface

\*/

public function filterByEovY($eovY = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($eovY)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $eovY)) {

$eovY = str\_replace('\*', '%', $eovY);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTelepulesTableMap::COL\_EOV\_Y, $eovY, $comparison);

}

/\*\*

\* Filter the query on the WGS84\_lat column

\*

\* Example usage:

\* <code>

\* $query->filterByWgs84Lat('fooValue'); // WHERE WGS84\_lat = 'fooValue'

\* $query->filterByWgs84Lat('%fooValue%'); // WHERE WGS84\_lat LIKE '%fooValue%'

\* </code>

\*

\* @param string $wgs84Lat The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTelepulesQuery The current query, for fluid interface

\*/

public function filterByWgs84Lat($wgs84Lat = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($wgs84Lat)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $wgs84Lat)) {

$wgs84Lat = str\_replace('\*', '%', $wgs84Lat);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTelepulesTableMap::COL\_WGS84\_LAT, $wgs84Lat, $comparison);

}

/\*\*

\* Filter the query on the WGS84\_lob column

\*

\* Example usage:

\* <code>

\* $query->filterByWgs84Lob('fooValue'); // WHERE WGS84\_lob = 'fooValue'

\* $query->filterByWgs84Lob('%fooValue%'); // WHERE WGS84\_lob LIKE '%fooValue%'

\* </code>

\*

\* @param string $wgs84Lob The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTelepulesQuery The current query, for fluid interface

\*/

public function filterByWgs84Lob($wgs84Lob = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($wgs84Lob)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $wgs84Lob)) {

$wgs84Lob = str\_replace('\*', '%', $wgs84Lob);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTelepulesTableMap::COL\_WGS84\_LOB, $wgs84Lob, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgTelepules $stgTelepules Object to remove from the list of results

\*

\* @return $this|ChildStgTelepulesQuery The current query, for fluid interface

\*/

public function prune($stgTelepules = null)

{

if ($stgTelepules) {

$this->addUsingAlias(StgTelepulesTableMap::COL\_TELEPULES\_NEV, $stgTelepules->getTelepulesNev(), Criteria::NOT\_EQUAL);

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_telepules table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTelepulesTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgTelepulesTableMap::clearInstancePool();

StgTelepulesTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTelepulesTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgTelepulesTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgTelepulesTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgTelepulesTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgTelepulesQuery

#### StgTomegkozlekedes.php

<?php

namespace Base;

use \StgTomegkozlekedesQuery as ChildStgTomegkozlekedesQuery;

use \Exception;

use \PDO;

use Map\StgTomegkozlekedesTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgTomegkozlekedes implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgTomegkozlekedesTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the vasutallomas\_lete field.

\* @var boolean

\*/

protected $vasutallomas\_lete;

/\*\*

\* The value for the kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre field.

\* @var int

\*/

protected $kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgTomegkozlekedes object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgTomegkozlekedes</code> instance. If

\* <code>obj</code> is an instance of <code>StgTomegkozlekedes</code>, delegates to

\* <code>equals(StgTomegkozlekedes)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgTomegkozlekedes The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [vasutallomas\_lete] column value.

\*

\* @return boolean

\*/

public function getVasutallomasLete()

{

return $this->vasutallomas\_lete;

}

/\*\*

\* Get the [vasutallomas\_lete] column value.

\*

\* @return boolean

\*/

public function isVasutallomasLete()

{

return $this->getVasutallomasLete();

}

/\*\*

\* Get the [kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre] column value.

\*

\* @return int

\*/

public function getKozvetlenJaratokSzamaNapontaAMegyeszekhelyre()

{

return $this->kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgTomegkozlekedesTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgTomegkozlekedesTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgTomegkozlekedesTableMap::translateFieldName('VasutallomasLete', TableMap::TYPE\_PHPNAME, $indexType)];

$this->vasutallomas\_lete = (null !== $col) ? (boolean) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgTomegkozlekedesTableMap::translateFieldName('KozvetlenJaratokSzamaNapontaAMegyeszekhelyre', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 4; // 4 = StgTomegkozlekedesTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgTomegkozlekedes'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgTomegkozlekedes The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgTomegkozlekedesTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgTomegkozlekedes The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgTomegkozlekedesTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Sets the value of the [vasutallomas\_lete] column.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\*

\* @param boolean|integer|string $v The new value

\* @return $this|\StgTomegkozlekedes The current object (for fluent API support)

\*/

public function setVasutallomasLete($v)

{

if ($v !== null) {

if (is\_string($v)) {

$v = in\_array(strtolower($v), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

} else {

$v = (boolean) $v;

}

}

if ($this->vasutallomas\_lete !== $v) {

$this->vasutallomas\_lete = $v;

$this->modifiedColumns[StgTomegkozlekedesTableMap::COL\_VASUTALLOMAS\_LETE] = true;

}

return $this;

} // setVasutallomasLete()

/\*\*

\* Set the value of [kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre] column.

\*

\* @param int $v new value

\* @return $this|\StgTomegkozlekedes The current object (for fluent API support)

\*/

public function setKozvetlenJaratokSzamaNapontaAMegyeszekhelyre($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre !== $v) {

$this->kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre = $v;

$this->modifiedColumns[StgTomegkozlekedesTableMap::COL\_KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE] = true;

}

return $this;

} // setKozvetlenJaratokSzamaNapontaAMegyeszekhelyre()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgTomegkozlekedesTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgTomegkozlekedesQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgTomegkozlekedes::setDeleted()

\* @see StgTomegkozlekedes::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTomegkozlekedesTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgTomegkozlekedesQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTomegkozlekedesTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgTomegkozlekedesTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgTomegkozlekedesTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgTomegkozlekedesTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgTomegkozlekedesTableMap::COL\_VASUTALLOMAS\_LETE)) {

$modifiedColumns[':p' . $index++] = 'VASUTALLOMAS\_LETE';

}

if ($this->isColumnModified(StgTomegkozlekedesTableMap::COL\_KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE)) {

$modifiedColumns[':p' . $index++] = 'KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE';

}

$sql = sprintf(

'INSERT INTO stg\_tomegkozlekedes (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'VASUTALLOMAS\_LETE':

$stmt->bindValue($identifier, (int) $this->vasutallomas\_lete, PDO::PARAM\_INT);

break;

case 'KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE':

$stmt->bindValue($identifier, $this->kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgTomegkozlekedesTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getVasutallomasLete();

break;

case 3:

return $this->getKozvetlenJaratokSzamaNapontaAMegyeszekhelyre();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgTomegkozlekedes'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgTomegkozlekedes'][$this->getPrimaryKey()] = true;

$keys = StgTomegkozlekedesTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getVasutallomasLete(),

$keys[3] => $this->getKozvetlenJaratokSzamaNapontaAMegyeszekhelyre(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgTomegkozlekedes

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgTomegkozlekedesTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgTomegkozlekedes

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setVasutallomasLete($value);

break;

case 3:

$this->setKozvetlenJaratokSzamaNapontaAMegyeszekhelyre($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgTomegkozlekedesTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setVasutallomasLete($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setKozvetlenJaratokSzamaNapontaAMegyeszekhelyre($arr[$keys[3]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgTomegkozlekedes The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgTomegkozlekedesTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgTomegkozlekedesTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgTomegkozlekedesTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgTomegkozlekedesTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgTomegkozlekedesTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgTomegkozlekedesTableMap::COL\_VASUTALLOMAS\_LETE)) {

$criteria->add(StgTomegkozlekedesTableMap::COL\_VASUTALLOMAS\_LETE, $this->vasutallomas\_lete);

}

if ($this->isColumnModified(StgTomegkozlekedesTableMap::COL\_KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE)) {

$criteria->add(StgTomegkozlekedesTableMap::COL\_KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE, $this->kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgTomegkozlekedes object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgTomegkozlekedes (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setVasutallomasLete($this->getVasutallomasLete());

$copyObj->setKozvetlenJaratokSzamaNapontaAMegyeszekhelyre($this->getKozvetlenJaratokSzamaNapontaAMegyeszekhelyre());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgTomegkozlekedes Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->vasutallomas\_lete = null;

$this->kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgTomegkozlekedesTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgTomegkozlekedesQuery.php

<?php

namespace Base;

use \StgTomegkozlekedes as ChildStgTomegkozlekedes;

use \StgTomegkozlekedesQuery as ChildStgTomegkozlekedesQuery;

use \Exception;

use Map\StgTomegkozlekedesTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_tomegkozlekedes' table.

\*

\*

\*

\* @method ChildStgTomegkozlekedesQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgTomegkozlekedesQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgTomegkozlekedesQuery orderByVasutallomasLete($order = Criteria::ASC) Order by the vasutallomas\_lete column

\* @method ChildStgTomegkozlekedesQuery orderByKozvetlenJaratokSzamaNapontaAMegyeszekhelyre($order = Criteria::ASC) Order by the kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre column

\*

\* @method ChildStgTomegkozlekedesQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgTomegkozlekedesQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgTomegkozlekedesQuery groupByVasutallomasLete() Group by the vasutallomas\_lete column

\* @method ChildStgTomegkozlekedesQuery groupByKozvetlenJaratokSzamaNapontaAMegyeszekhelyre() Group by the kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre column

\*

\* @method ChildStgTomegkozlekedesQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgTomegkozlekedesQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgTomegkozlekedesQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgTomegkozlekedes findOne(ConnectionInterface $con = null) Return the first ChildStgTomegkozlekedes matching the query

\* @method ChildStgTomegkozlekedes findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgTomegkozlekedes matching the query, or a new ChildStgTomegkozlekedes object populated from the query conditions when no match is found

\*

\* @method ChildStgTomegkozlekedes findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgTomegkozlekedes filtered by the telepules\_nev column

\* @method ChildStgTomegkozlekedes findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgTomegkozlekedes filtered by the telepules\_KSHKOD column

\* @method ChildStgTomegkozlekedes findOneByVasutallomasLete(boolean $vasutallomas\_lete) Return the first ChildStgTomegkozlekedes filtered by the vasutallomas\_lete column

\* @method ChildStgTomegkozlekedes findOneByKozvetlenJaratokSzamaNapontaAMegyeszekhelyre(int $kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre) Return the first ChildStgTomegkozlekedes filtered by the kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre column

\*

\* @method ChildStgTomegkozlekedes[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgTomegkozlekedes objects based on current ModelCriteria

\* @method ChildStgTomegkozlekedes[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgTomegkozlekedes objects filtered by the telepules\_nev column

\* @method ChildStgTomegkozlekedes[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgTomegkozlekedes objects filtered by the telepules\_KSHKOD column

\* @method ChildStgTomegkozlekedes[]|ObjectCollection findByVasutallomasLete(boolean $vasutallomas\_lete) Return ChildStgTomegkozlekedes objects filtered by the vasutallomas\_lete column

\* @method ChildStgTomegkozlekedes[]|ObjectCollection findByKozvetlenJaratokSzamaNapontaAMegyeszekhelyre(int $kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre) Return ChildStgTomegkozlekedes objects filtered by the kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre column

\* @method ChildStgTomegkozlekedes[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgTomegkozlekedesQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgTomegkozlekedesQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgTomegkozlekedes', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgTomegkozlekedesQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgTomegkozlekedesQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgTomegkozlekedesQuery) {

return $criteria;

}

$query = new ChildStgTomegkozlekedesQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgTomegkozlekedes|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgTomegkozlekedes object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgTomegkozlekedes object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgTomegkozlekedesQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgTomegkozlekedes object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgTomegkozlekedesQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgTomegkozlekedes object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTomegkozlekedesQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTomegkozlekedesTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTomegkozlekedesQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgTomegkozlekedesTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the vasutallomas\_lete column

\*

\* Example usage:

\* <code>

\* $query->filterByVasutallomasLete(true); // WHERE vasutallomas\_lete = true

\* $query->filterByVasutallomasLete('yes'); // WHERE vasutallomas\_lete = true

\* </code>

\*

\* @param boolean|string $vasutallomasLete The value to use as filter.

\* Non-boolean arguments are converted using the following rules:

\* \* 1, '1', 'true', 'on', and 'yes' are converted to boolean true

\* \* 0, '0', 'false', 'off', and 'no' are converted to boolean false

\* Check on string values is case insensitive (so 'FaLsE' is seen as 'false').

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTomegkozlekedesQuery The current query, for fluid interface

\*/

public function filterByVasutallomasLete($vasutallomasLete = null, $comparison = null)

{

if (is\_string($vasutallomasLete)) {

$vasutallomasLete = in\_array(strtolower($vasutallomasLete), array('false', 'off', '-', 'no', 'n', '0', '')) ? false : true;

}

return $this->addUsingAlias(StgTomegkozlekedesTableMap::COL\_VASUTALLOMAS\_LETE, $vasutallomasLete, $comparison);

}

/\*\*

\* Filter the query on the kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre column

\*

\* Example usage:

\* <code>

\* $query->filterByKozvetlenJaratokSzamaNapontaAMegyeszekhelyre(1234); // WHERE kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre = 1234

\* $query->filterByKozvetlenJaratokSzamaNapontaAMegyeszekhelyre(array(12, 34)); // WHERE kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre IN (12, 34)

\* $query->filterByKozvetlenJaratokSzamaNapontaAMegyeszekhelyre(array('min' => 12)); // WHERE kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre > 12

\* </code>

\*

\* @param mixed $kozvetlenJaratokSzamaNapontaAMegyeszekhelyre The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgTomegkozlekedesQuery The current query, for fluid interface

\*/

public function filterByKozvetlenJaratokSzamaNapontaAMegyeszekhelyre($kozvetlenJaratokSzamaNapontaAMegyeszekhelyre = null, $comparison = null)

{

if (is\_array($kozvetlenJaratokSzamaNapontaAMegyeszekhelyre)) {

$useMinMax = false;

if (isset($kozvetlenJaratokSzamaNapontaAMegyeszekhelyre['min'])) {

$this->addUsingAlias(StgTomegkozlekedesTableMap::COL\_KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE, $kozvetlenJaratokSzamaNapontaAMegyeszekhelyre['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($kozvetlenJaratokSzamaNapontaAMegyeszekhelyre['max'])) {

$this->addUsingAlias(StgTomegkozlekedesTableMap::COL\_KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE, $kozvetlenJaratokSzamaNapontaAMegyeszekhelyre['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgTomegkozlekedesTableMap::COL\_KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE, $kozvetlenJaratokSzamaNapontaAMegyeszekhelyre, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgTomegkozlekedes $stgTomegkozlekedes Object to remove from the list of results

\*

\* @return $this|ChildStgTomegkozlekedesQuery The current query, for fluid interface

\*/

public function prune($stgTomegkozlekedes = null)

{

if ($stgTomegkozlekedes) {

throw new LogicException('StgTomegkozlekedes object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_tomegkozlekedes table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTomegkozlekedesTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgTomegkozlekedesTableMap::clearInstancePool();

StgTomegkozlekedesTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTomegkozlekedesTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgTomegkozlekedesTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgTomegkozlekedesTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgTomegkozlekedesTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgTomegkozlekedesQuery

#### StgUgyintezesAltalanosJellemzese.php

<?php

namespace Base;

use \StgUgyintezesAltalanosJellemzeseQuery as ChildStgUgyintezesAltalanosJellemzeseQuery;

use \Exception;

use \PDO;

use Map\StgUgyintezesAltalanosJellemzeseTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgUgyintezesAltalanosJellemzese implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgUgyintezesAltalanosJellemzeseTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the egeszsegugyi\_ellatasok\_szama field.

\* @var int

\*/

protected $egeszsegugyi\_ellatasok\_szama;

/\*\*

\* The value for the postai\_gyogyszertari\_szolgaltatasok\_szama field.

\* @var int

\*/

protected $postai\_gyogyszertari\_szolgaltatasok\_szama;

/\*\*

\* The value for the buntetoperek\_szama field.

\* @var int

\*/

protected $buntetoperek\_szama;

/\*\*

\* The value for the kereskedelmi\_szolgaltatasok\_szama field.

\* @var int

\*/

protected $kereskedelmi\_szolgaltatasok\_szama;

/\*\*

\* The value for the szocialis\_szolgaltatasok\_szama field.

\* @var int

\*/

protected $szocialis\_szolgaltatasok\_szama;

/\*\*

\* The value for the okmanyirodai\_ugyintezesek\_szama field.

\* @var int

\*/

protected $okmanyirodai\_ugyintezesek\_szama;

/\*\*

\* The value for the bankok\_szama field.

\* @var int

\*/

protected $bankok\_szama;

/\*\*

\* The value for the benzinkutak\_szama field.

\* @var int

\*/

protected $benzinkutak\_szama;

/\*\*

\* The value for the korjegyzosegek\_foldhivatalok\_szama field.

\* @var int

\*/

protected $korjegyzosegek\_foldhivatalok\_szama;

/\*\*

\* The value for the kozmuszolgaltatasok field.

\* @var int

\*/

protected $kozmuszolgaltatasok;

/\*\*

\* The value for the munkaugyi\_szolgaltatasok\_szama field.

\* @var int

\*/

protected $munkaugyi\_szolgaltatasok\_szama;

/\*\*

\* The value for the bolcsodei\_szolgaltatasok\_szama field.

\* @var int

\*/

protected $bolcsodei\_szolgaltatasok\_szama;

/\*\*

\* The value for the mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama field.

\* @var int

\*/

protected $mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgUgyintezesAltalanosJellemzese object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgUgyintezesAltalanosJellemzese</code> instance. If

\* <code>obj</code> is an instance of <code>StgUgyintezesAltalanosJellemzese</code>, delegates to

\* <code>equals(StgUgyintezesAltalanosJellemzese)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgUgyintezesAltalanosJellemzese The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [egeszsegugyi\_ellatasok\_szama] column value.

\*

\* @return int

\*/

public function getEgeszsegugyiEllatasokSzama()

{

return $this->egeszsegugyi\_ellatasok\_szama;

}

/\*\*

\* Get the [postai\_gyogyszertari\_szolgaltatasok\_szama] column value.

\*

\* @return int

\*/

public function getPostaiGyogyszertariSzolgaltatasokSzama()

{

return $this->postai\_gyogyszertari\_szolgaltatasok\_szama;

}

/\*\*

\* Get the [buntetoperek\_szama] column value.

\*

\* @return int

\*/

public function getBuntetoperekSzama()

{

return $this->buntetoperek\_szama;

}

/\*\*

\* Get the [kereskedelmi\_szolgaltatasok\_szama] column value.

\*

\* @return int

\*/

public function getKereskedelmiSzolgaltatasokSzama()

{

return $this->kereskedelmi\_szolgaltatasok\_szama;

}

/\*\*

\* Get the [szocialis\_szolgaltatasok\_szama] column value.

\*

\* @return int

\*/

public function getSzocialisSzolgaltatasokSzama()

{

return $this->szocialis\_szolgaltatasok\_szama;

}

/\*\*

\* Get the [okmanyirodai\_ugyintezesek\_szama] column value.

\*

\* @return int

\*/

public function getOkmanyirodaiUgyintezesekSzama()

{

return $this->okmanyirodai\_ugyintezesek\_szama;

}

/\*\*

\* Get the [bankok\_szama] column value.

\*

\* @return int

\*/

public function getBankokSzama()

{

return $this->bankok\_szama;

}

/\*\*

\* Get the [benzinkutak\_szama] column value.

\*

\* @return int

\*/

public function getBenzinkutakSzama()

{

return $this->benzinkutak\_szama;

}

/\*\*

\* Get the [korjegyzosegek\_foldhivatalok\_szama] column value.

\*

\* @return int

\*/

public function getKorjegyzosegekFoldhivatalokSzama()

{

return $this->korjegyzosegek\_foldhivatalok\_szama;

}

/\*\*

\* Get the [kozmuszolgaltatasok] column value.

\*

\* @return int

\*/

public function getKozmuszolgaltatasok()

{

return $this->kozmuszolgaltatasok;

}

/\*\*

\* Get the [munkaugyi\_szolgaltatasok\_szama] column value.

\*

\* @return int

\*/

public function getMunkaugyiSzolgaltatasokSzama()

{

return $this->munkaugyi\_szolgaltatasok\_szama;

}

/\*\*

\* Get the [bolcsodei\_szolgaltatasok\_szama] column value.

\*

\* @return int

\*/

public function getBolcsodeiSzolgaltatasokSzama()

{

return $this->bolcsodei\_szolgaltatasok\_szama;

}

/\*\*

\* Get the [mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama] column value.

\*

\* @return int

\*/

public function getMobiltelefonUgyintezesiSzolgaltatasokSzama()

{

return $this->mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgUgyintezesAltalanosJellemzeseTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgUgyintezesAltalanosJellemzeseTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgUgyintezesAltalanosJellemzeseTableMap::translateFieldName('EgeszsegugyiEllatasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->egeszsegugyi\_ellatasok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgUgyintezesAltalanosJellemzeseTableMap::translateFieldName('PostaiGyogyszertariSzolgaltatasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->postai\_gyogyszertari\_szolgaltatasok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgUgyintezesAltalanosJellemzeseTableMap::translateFieldName('BuntetoperekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->buntetoperek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 5 + $startcol : StgUgyintezesAltalanosJellemzeseTableMap::translateFieldName('KereskedelmiSzolgaltatasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kereskedelmi\_szolgaltatasok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 6 + $startcol : StgUgyintezesAltalanosJellemzeseTableMap::translateFieldName('SzocialisSzolgaltatasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->szocialis\_szolgaltatasok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 7 + $startcol : StgUgyintezesAltalanosJellemzeseTableMap::translateFieldName('OkmanyirodaiUgyintezesekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->okmanyirodai\_ugyintezesek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 8 + $startcol : StgUgyintezesAltalanosJellemzeseTableMap::translateFieldName('BankokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->bankok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 9 + $startcol : StgUgyintezesAltalanosJellemzeseTableMap::translateFieldName('BenzinkutakSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->benzinkutak\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 10 + $startcol : StgUgyintezesAltalanosJellemzeseTableMap::translateFieldName('KorjegyzosegekFoldhivatalokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->korjegyzosegek\_foldhivatalok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 11 + $startcol : StgUgyintezesAltalanosJellemzeseTableMap::translateFieldName('Kozmuszolgaltatasok', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kozmuszolgaltatasok = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 12 + $startcol : StgUgyintezesAltalanosJellemzeseTableMap::translateFieldName('MunkaugyiSzolgaltatasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->munkaugyi\_szolgaltatasok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 13 + $startcol : StgUgyintezesAltalanosJellemzeseTableMap::translateFieldName('BolcsodeiSzolgaltatasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->bolcsodei\_szolgaltatasok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 14 + $startcol : StgUgyintezesAltalanosJellemzeseTableMap::translateFieldName('MobiltelefonUgyintezesiSzolgaltatasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 15; // 15 = StgUgyintezesAltalanosJellemzeseTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgUgyintezesAltalanosJellemzese'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgUgyintezesAltalanosJellemzese The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgUgyintezesAltalanosJellemzeseTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgUgyintezesAltalanosJellemzese The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgUgyintezesAltalanosJellemzeseTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [egeszsegugyi\_ellatasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgUgyintezesAltalanosJellemzese The current object (for fluent API support)

\*/

public function setEgeszsegugyiEllatasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->egeszsegugyi\_ellatasok\_szama !== $v) {

$this->egeszsegugyi\_ellatasok\_szama = $v;

$this->modifiedColumns[StgUgyintezesAltalanosJellemzeseTableMap::COL\_EGESZSEGUGYI\_ELLATASOK\_SZAMA] = true;

}

return $this;

} // setEgeszsegugyiEllatasokSzama()

/\*\*

\* Set the value of [postai\_gyogyszertari\_szolgaltatasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgUgyintezesAltalanosJellemzese The current object (for fluent API support)

\*/

public function setPostaiGyogyszertariSzolgaltatasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->postai\_gyogyszertari\_szolgaltatasok\_szama !== $v) {

$this->postai\_gyogyszertari\_szolgaltatasok\_szama = $v;

$this->modifiedColumns[StgUgyintezesAltalanosJellemzeseTableMap::COL\_POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA] = true;

}

return $this;

} // setPostaiGyogyszertariSzolgaltatasokSzama()

/\*\*

\* Set the value of [buntetoperek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgUgyintezesAltalanosJellemzese The current object (for fluent API support)

\*/

public function setBuntetoperekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->buntetoperek\_szama !== $v) {

$this->buntetoperek\_szama = $v;

$this->modifiedColumns[StgUgyintezesAltalanosJellemzeseTableMap::COL\_BUNTETOPEREK\_SZAMA] = true;

}

return $this;

} // setBuntetoperekSzama()

/\*\*

\* Set the value of [kereskedelmi\_szolgaltatasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgUgyintezesAltalanosJellemzese The current object (for fluent API support)

\*/

public function setKereskedelmiSzolgaltatasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->kereskedelmi\_szolgaltatasok\_szama !== $v) {

$this->kereskedelmi\_szolgaltatasok\_szama = $v;

$this->modifiedColumns[StgUgyintezesAltalanosJellemzeseTableMap::COL\_KERESKEDELMI\_SZOLGALTATASOK\_SZAMA] = true;

}

return $this;

} // setKereskedelmiSzolgaltatasokSzama()

/\*\*

\* Set the value of [szocialis\_szolgaltatasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgUgyintezesAltalanosJellemzese The current object (for fluent API support)

\*/

public function setSzocialisSzolgaltatasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->szocialis\_szolgaltatasok\_szama !== $v) {

$this->szocialis\_szolgaltatasok\_szama = $v;

$this->modifiedColumns[StgUgyintezesAltalanosJellemzeseTableMap::COL\_SZOCIALIS\_SZOLGALTATASOK\_SZAMA] = true;

}

return $this;

} // setSzocialisSzolgaltatasokSzama()

/\*\*

\* Set the value of [okmanyirodai\_ugyintezesek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgUgyintezesAltalanosJellemzese The current object (for fluent API support)

\*/

public function setOkmanyirodaiUgyintezesekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->okmanyirodai\_ugyintezesek\_szama !== $v) {

$this->okmanyirodai\_ugyintezesek\_szama = $v;

$this->modifiedColumns[StgUgyintezesAltalanosJellemzeseTableMap::COL\_OKMANYIRODAI\_UGYINTEZESEK\_SZAMA] = true;

}

return $this;

} // setOkmanyirodaiUgyintezesekSzama()

/\*\*

\* Set the value of [bankok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgUgyintezesAltalanosJellemzese The current object (for fluent API support)

\*/

public function setBankokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->bankok\_szama !== $v) {

$this->bankok\_szama = $v;

$this->modifiedColumns[StgUgyintezesAltalanosJellemzeseTableMap::COL\_BANKOK\_SZAMA] = true;

}

return $this;

} // setBankokSzama()

/\*\*

\* Set the value of [benzinkutak\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgUgyintezesAltalanosJellemzese The current object (for fluent API support)

\*/

public function setBenzinkutakSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->benzinkutak\_szama !== $v) {

$this->benzinkutak\_szama = $v;

$this->modifiedColumns[StgUgyintezesAltalanosJellemzeseTableMap::COL\_BENZINKUTAK\_SZAMA] = true;

}

return $this;

} // setBenzinkutakSzama()

/\*\*

\* Set the value of [korjegyzosegek\_foldhivatalok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgUgyintezesAltalanosJellemzese The current object (for fluent API support)

\*/

public function setKorjegyzosegekFoldhivatalokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->korjegyzosegek\_foldhivatalok\_szama !== $v) {

$this->korjegyzosegek\_foldhivatalok\_szama = $v;

$this->modifiedColumns[StgUgyintezesAltalanosJellemzeseTableMap::COL\_KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA] = true;

}

return $this;

} // setKorjegyzosegekFoldhivatalokSzama()

/\*\*

\* Set the value of [kozmuszolgaltatasok] column.

\*

\* @param int $v new value

\* @return $this|\StgUgyintezesAltalanosJellemzese The current object (for fluent API support)

\*/

public function setKozmuszolgaltatasok($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->kozmuszolgaltatasok !== $v) {

$this->kozmuszolgaltatasok = $v;

$this->modifiedColumns[StgUgyintezesAltalanosJellemzeseTableMap::COL\_KOZMUSZOLGALTATASOK] = true;

}

return $this;

} // setKozmuszolgaltatasok()

/\*\*

\* Set the value of [munkaugyi\_szolgaltatasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgUgyintezesAltalanosJellemzese The current object (for fluent API support)

\*/

public function setMunkaugyiSzolgaltatasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->munkaugyi\_szolgaltatasok\_szama !== $v) {

$this->munkaugyi\_szolgaltatasok\_szama = $v;

$this->modifiedColumns[StgUgyintezesAltalanosJellemzeseTableMap::COL\_MUNKAUGYI\_SZOLGALTATASOK\_SZAMA] = true;

}

return $this;

} // setMunkaugyiSzolgaltatasokSzama()

/\*\*

\* Set the value of [bolcsodei\_szolgaltatasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgUgyintezesAltalanosJellemzese The current object (for fluent API support)

\*/

public function setBolcsodeiSzolgaltatasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->bolcsodei\_szolgaltatasok\_szama !== $v) {

$this->bolcsodei\_szolgaltatasok\_szama = $v;

$this->modifiedColumns[StgUgyintezesAltalanosJellemzeseTableMap::COL\_BOLCSODEI\_SZOLGALTATASOK\_SZAMA] = true;

}

return $this;

} // setBolcsodeiSzolgaltatasokSzama()

/\*\*

\* Set the value of [mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgUgyintezesAltalanosJellemzese The current object (for fluent API support)

\*/

public function setMobiltelefonUgyintezesiSzolgaltatasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama !== $v) {

$this->mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama = $v;

$this->modifiedColumns[StgUgyintezesAltalanosJellemzeseTableMap::COL\_MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA] = true;

}

return $this;

} // setMobiltelefonUgyintezesiSzolgaltatasokSzama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgUgyintezesAltalanosJellemzeseTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgUgyintezesAltalanosJellemzeseQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgUgyintezesAltalanosJellemzese::setDeleted()

\* @see StgUgyintezesAltalanosJellemzese::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgUgyintezesAltalanosJellemzeseTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgUgyintezesAltalanosJellemzeseQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgUgyintezesAltalanosJellemzeseTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgUgyintezesAltalanosJellemzeseTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_EGESZSEGUGYI\_ELLATASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'EGESZSEGUGYI\_ELLATASOK\_SZAMA';

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA';

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BUNTETOPEREK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'BUNTETOPEREK\_SZAMA';

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KERESKEDELMI\_SZOLGALTATASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'KERESKEDELMI\_SZOLGALTATASOK\_SZAMA';

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_SZOCIALIS\_SZOLGALTATASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'SZOCIALIS\_SZOLGALTATASOK\_SZAMA';

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_OKMANYIRODAI\_UGYINTEZESEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'OKMANYIRODAI\_UGYINTEZESEK\_SZAMA';

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BANKOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'BANKOK\_SZAMA';

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BENZINKUTAK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'BENZINKUTAK\_SZAMA';

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA';

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KOZMUSZOLGALTATASOK)) {

$modifiedColumns[':p' . $index++] = 'KOZMUSZOLGALTATASOK';

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_MUNKAUGYI\_SZOLGALTATASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'MUNKAUGYI\_SZOLGALTATASOK\_SZAMA';

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BOLCSODEI\_SZOLGALTATASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'BOLCSODEI\_SZOLGALTATASOK\_SZAMA';

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_ugyintezes\_altalanos\_jellemzese (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'EGESZSEGUGYI\_ELLATASOK\_SZAMA':

$stmt->bindValue($identifier, $this->egeszsegugyi\_ellatasok\_szama, PDO::PARAM\_INT);

break;

case 'POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA':

$stmt->bindValue($identifier, $this->postai\_gyogyszertari\_szolgaltatasok\_szama, PDO::PARAM\_INT);

break;

case 'BUNTETOPEREK\_SZAMA':

$stmt->bindValue($identifier, $this->buntetoperek\_szama, PDO::PARAM\_INT);

break;

case 'KERESKEDELMI\_SZOLGALTATASOK\_SZAMA':

$stmt->bindValue($identifier, $this->kereskedelmi\_szolgaltatasok\_szama, PDO::PARAM\_INT);

break;

case 'SZOCIALIS\_SZOLGALTATASOK\_SZAMA':

$stmt->bindValue($identifier, $this->szocialis\_szolgaltatasok\_szama, PDO::PARAM\_INT);

break;

case 'OKMANYIRODAI\_UGYINTEZESEK\_SZAMA':

$stmt->bindValue($identifier, $this->okmanyirodai\_ugyintezesek\_szama, PDO::PARAM\_INT);

break;

case 'BANKOK\_SZAMA':

$stmt->bindValue($identifier, $this->bankok\_szama, PDO::PARAM\_INT);

break;

case 'BENZINKUTAK\_SZAMA':

$stmt->bindValue($identifier, $this->benzinkutak\_szama, PDO::PARAM\_INT);

break;

case 'KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA':

$stmt->bindValue($identifier, $this->korjegyzosegek\_foldhivatalok\_szama, PDO::PARAM\_INT);

break;

case 'KOZMUSZOLGALTATASOK':

$stmt->bindValue($identifier, $this->kozmuszolgaltatasok, PDO::PARAM\_INT);

break;

case 'MUNKAUGYI\_SZOLGALTATASOK\_SZAMA':

$stmt->bindValue($identifier, $this->munkaugyi\_szolgaltatasok\_szama, PDO::PARAM\_INT);

break;

case 'BOLCSODEI\_SZOLGALTATASOK\_SZAMA':

$stmt->bindValue($identifier, $this->bolcsodei\_szolgaltatasok\_szama, PDO::PARAM\_INT);

break;

case 'MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA':

$stmt->bindValue($identifier, $this->mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgUgyintezesAltalanosJellemzeseTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getEgeszsegugyiEllatasokSzama();

break;

case 3:

return $this->getPostaiGyogyszertariSzolgaltatasokSzama();

break;

case 4:

return $this->getBuntetoperekSzama();

break;

case 5:

return $this->getKereskedelmiSzolgaltatasokSzama();

break;

case 6:

return $this->getSzocialisSzolgaltatasokSzama();

break;

case 7:

return $this->getOkmanyirodaiUgyintezesekSzama();

break;

case 8:

return $this->getBankokSzama();

break;

case 9:

return $this->getBenzinkutakSzama();

break;

case 10:

return $this->getKorjegyzosegekFoldhivatalokSzama();

break;

case 11:

return $this->getKozmuszolgaltatasok();

break;

case 12:

return $this->getMunkaugyiSzolgaltatasokSzama();

break;

case 13:

return $this->getBolcsodeiSzolgaltatasokSzama();

break;

case 14:

return $this->getMobiltelefonUgyintezesiSzolgaltatasokSzama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgUgyintezesAltalanosJellemzese'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgUgyintezesAltalanosJellemzese'][$this->getPrimaryKey()] = true;

$keys = StgUgyintezesAltalanosJellemzeseTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getEgeszsegugyiEllatasokSzama(),

$keys[3] => $this->getPostaiGyogyszertariSzolgaltatasokSzama(),

$keys[4] => $this->getBuntetoperekSzama(),

$keys[5] => $this->getKereskedelmiSzolgaltatasokSzama(),

$keys[6] => $this->getSzocialisSzolgaltatasokSzama(),

$keys[7] => $this->getOkmanyirodaiUgyintezesekSzama(),

$keys[8] => $this->getBankokSzama(),

$keys[9] => $this->getBenzinkutakSzama(),

$keys[10] => $this->getKorjegyzosegekFoldhivatalokSzama(),

$keys[11] => $this->getKozmuszolgaltatasok(),

$keys[12] => $this->getMunkaugyiSzolgaltatasokSzama(),

$keys[13] => $this->getBolcsodeiSzolgaltatasokSzama(),

$keys[14] => $this->getMobiltelefonUgyintezesiSzolgaltatasokSzama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgUgyintezesAltalanosJellemzese

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgUgyintezesAltalanosJellemzeseTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgUgyintezesAltalanosJellemzese

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setEgeszsegugyiEllatasokSzama($value);

break;

case 3:

$this->setPostaiGyogyszertariSzolgaltatasokSzama($value);

break;

case 4:

$this->setBuntetoperekSzama($value);

break;

case 5:

$this->setKereskedelmiSzolgaltatasokSzama($value);

break;

case 6:

$this->setSzocialisSzolgaltatasokSzama($value);

break;

case 7:

$this->setOkmanyirodaiUgyintezesekSzama($value);

break;

case 8:

$this->setBankokSzama($value);

break;

case 9:

$this->setBenzinkutakSzama($value);

break;

case 10:

$this->setKorjegyzosegekFoldhivatalokSzama($value);

break;

case 11:

$this->setKozmuszolgaltatasok($value);

break;

case 12:

$this->setMunkaugyiSzolgaltatasokSzama($value);

break;

case 13:

$this->setBolcsodeiSzolgaltatasokSzama($value);

break;

case 14:

$this->setMobiltelefonUgyintezesiSzolgaltatasokSzama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgUgyintezesAltalanosJellemzeseTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setEgeszsegugyiEllatasokSzama($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setPostaiGyogyszertariSzolgaltatasokSzama($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setBuntetoperekSzama($arr[$keys[4]]);

}

if (array\_key\_exists($keys[5], $arr)) {

$this->setKereskedelmiSzolgaltatasokSzama($arr[$keys[5]]);

}

if (array\_key\_exists($keys[6], $arr)) {

$this->setSzocialisSzolgaltatasokSzama($arr[$keys[6]]);

}

if (array\_key\_exists($keys[7], $arr)) {

$this->setOkmanyirodaiUgyintezesekSzama($arr[$keys[7]]);

}

if (array\_key\_exists($keys[8], $arr)) {

$this->setBankokSzama($arr[$keys[8]]);

}

if (array\_key\_exists($keys[9], $arr)) {

$this->setBenzinkutakSzama($arr[$keys[9]]);

}

if (array\_key\_exists($keys[10], $arr)) {

$this->setKorjegyzosegekFoldhivatalokSzama($arr[$keys[10]]);

}

if (array\_key\_exists($keys[11], $arr)) {

$this->setKozmuszolgaltatasok($arr[$keys[11]]);

}

if (array\_key\_exists($keys[12], $arr)) {

$this->setMunkaugyiSzolgaltatasokSzama($arr[$keys[12]]);

}

if (array\_key\_exists($keys[13], $arr)) {

$this->setBolcsodeiSzolgaltatasokSzama($arr[$keys[13]]);

}

if (array\_key\_exists($keys[14], $arr)) {

$this->setMobiltelefonUgyintezesiSzolgaltatasokSzama($arr[$keys[14]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgUgyintezesAltalanosJellemzese The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgUgyintezesAltalanosJellemzeseTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgUgyintezesAltalanosJellemzeseTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgUgyintezesAltalanosJellemzeseTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_EGESZSEGUGYI\_ELLATASOK\_SZAMA)) {

$criteria->add(StgUgyintezesAltalanosJellemzeseTableMap::COL\_EGESZSEGUGYI\_ELLATASOK\_SZAMA, $this->egeszsegugyi\_ellatasok\_szama);

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA)) {

$criteria->add(StgUgyintezesAltalanosJellemzeseTableMap::COL\_POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA, $this->postai\_gyogyszertari\_szolgaltatasok\_szama);

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BUNTETOPEREK\_SZAMA)) {

$criteria->add(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BUNTETOPEREK\_SZAMA, $this->buntetoperek\_szama);

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KERESKEDELMI\_SZOLGALTATASOK\_SZAMA)) {

$criteria->add(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KERESKEDELMI\_SZOLGALTATASOK\_SZAMA, $this->kereskedelmi\_szolgaltatasok\_szama);

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_SZOCIALIS\_SZOLGALTATASOK\_SZAMA)) {

$criteria->add(StgUgyintezesAltalanosJellemzeseTableMap::COL\_SZOCIALIS\_SZOLGALTATASOK\_SZAMA, $this->szocialis\_szolgaltatasok\_szama);

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_OKMANYIRODAI\_UGYINTEZESEK\_SZAMA)) {

$criteria->add(StgUgyintezesAltalanosJellemzeseTableMap::COL\_OKMANYIRODAI\_UGYINTEZESEK\_SZAMA, $this->okmanyirodai\_ugyintezesek\_szama);

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BANKOK\_SZAMA)) {

$criteria->add(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BANKOK\_SZAMA, $this->bankok\_szama);

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BENZINKUTAK\_SZAMA)) {

$criteria->add(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BENZINKUTAK\_SZAMA, $this->benzinkutak\_szama);

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA)) {

$criteria->add(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA, $this->korjegyzosegek\_foldhivatalok\_szama);

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KOZMUSZOLGALTATASOK)) {

$criteria->add(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KOZMUSZOLGALTATASOK, $this->kozmuszolgaltatasok);

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_MUNKAUGYI\_SZOLGALTATASOK\_SZAMA)) {

$criteria->add(StgUgyintezesAltalanosJellemzeseTableMap::COL\_MUNKAUGYI\_SZOLGALTATASOK\_SZAMA, $this->munkaugyi\_szolgaltatasok\_szama);

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BOLCSODEI\_SZOLGALTATASOK\_SZAMA)) {

$criteria->add(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BOLCSODEI\_SZOLGALTATASOK\_SZAMA, $this->bolcsodei\_szolgaltatasok\_szama);

}

if ($this->isColumnModified(StgUgyintezesAltalanosJellemzeseTableMap::COL\_MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA)) {

$criteria->add(StgUgyintezesAltalanosJellemzeseTableMap::COL\_MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA, $this->mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgUgyintezesAltalanosJellemzese object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgUgyintezesAltalanosJellemzese (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setEgeszsegugyiEllatasokSzama($this->getEgeszsegugyiEllatasokSzama());

$copyObj->setPostaiGyogyszertariSzolgaltatasokSzama($this->getPostaiGyogyszertariSzolgaltatasokSzama());

$copyObj->setBuntetoperekSzama($this->getBuntetoperekSzama());

$copyObj->setKereskedelmiSzolgaltatasokSzama($this->getKereskedelmiSzolgaltatasokSzama());

$copyObj->setSzocialisSzolgaltatasokSzama($this->getSzocialisSzolgaltatasokSzama());

$copyObj->setOkmanyirodaiUgyintezesekSzama($this->getOkmanyirodaiUgyintezesekSzama());

$copyObj->setBankokSzama($this->getBankokSzama());

$copyObj->setBenzinkutakSzama($this->getBenzinkutakSzama());

$copyObj->setKorjegyzosegekFoldhivatalokSzama($this->getKorjegyzosegekFoldhivatalokSzama());

$copyObj->setKozmuszolgaltatasok($this->getKozmuszolgaltatasok());

$copyObj->setMunkaugyiSzolgaltatasokSzama($this->getMunkaugyiSzolgaltatasokSzama());

$copyObj->setBolcsodeiSzolgaltatasokSzama($this->getBolcsodeiSzolgaltatasokSzama());

$copyObj->setMobiltelefonUgyintezesiSzolgaltatasokSzama($this->getMobiltelefonUgyintezesiSzolgaltatasokSzama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgUgyintezesAltalanosJellemzese Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->egeszsegugyi\_ellatasok\_szama = null;

$this->postai\_gyogyszertari\_szolgaltatasok\_szama = null;

$this->buntetoperek\_szama = null;

$this->kereskedelmi\_szolgaltatasok\_szama = null;

$this->szocialis\_szolgaltatasok\_szama = null;

$this->okmanyirodai\_ugyintezesek\_szama = null;

$this->bankok\_szama = null;

$this->benzinkutak\_szama = null;

$this->korjegyzosegek\_foldhivatalok\_szama = null;

$this->kozmuszolgaltatasok = null;

$this->munkaugyi\_szolgaltatasok\_szama = null;

$this->bolcsodei\_szolgaltatasok\_szama = null;

$this->mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgUgyintezesAltalanosJellemzeseTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgUgyintezesAltalanosJellemzeseQuery.php

<?php

namespace Base;

use \StgUgyintezesAltalanosJellemzese as ChildStgUgyintezesAltalanosJellemzese;

use \StgUgyintezesAltalanosJellemzeseQuery as ChildStgUgyintezesAltalanosJellemzeseQuery;

use \Exception;

use Map\StgUgyintezesAltalanosJellemzeseTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_ugyintezes\_altalanos\_jellemzese' table.

\*

\*

\*

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery orderByEgeszsegugyiEllatasokSzama($order = Criteria::ASC) Order by the egeszsegugyi\_ellatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery orderByPostaiGyogyszertariSzolgaltatasokSzama($order = Criteria::ASC) Order by the postai\_gyogyszertari\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery orderByBuntetoperekSzama($order = Criteria::ASC) Order by the buntetoperek\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery orderByKereskedelmiSzolgaltatasokSzama($order = Criteria::ASC) Order by the kereskedelmi\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery orderBySzocialisSzolgaltatasokSzama($order = Criteria::ASC) Order by the szocialis\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery orderByOkmanyirodaiUgyintezesekSzama($order = Criteria::ASC) Order by the okmanyirodai\_ugyintezesek\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery orderByBankokSzama($order = Criteria::ASC) Order by the bankok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery orderByBenzinkutakSzama($order = Criteria::ASC) Order by the benzinkutak\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery orderByKorjegyzosegekFoldhivatalokSzama($order = Criteria::ASC) Order by the korjegyzosegek\_foldhivatalok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery orderByKozmuszolgaltatasok($order = Criteria::ASC) Order by the kozmuszolgaltatasok column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery orderByMunkaugyiSzolgaltatasokSzama($order = Criteria::ASC) Order by the munkaugyi\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery orderByBolcsodeiSzolgaltatasokSzama($order = Criteria::ASC) Order by the bolcsodei\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery orderByMobiltelefonUgyintezesiSzolgaltatasokSzama($order = Criteria::ASC) Order by the mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama column

\*

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery groupByEgeszsegugyiEllatasokSzama() Group by the egeszsegugyi\_ellatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery groupByPostaiGyogyszertariSzolgaltatasokSzama() Group by the postai\_gyogyszertari\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery groupByBuntetoperekSzama() Group by the buntetoperek\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery groupByKereskedelmiSzolgaltatasokSzama() Group by the kereskedelmi\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery groupBySzocialisSzolgaltatasokSzama() Group by the szocialis\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery groupByOkmanyirodaiUgyintezesekSzama() Group by the okmanyirodai\_ugyintezesek\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery groupByBankokSzama() Group by the bankok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery groupByBenzinkutakSzama() Group by the benzinkutak\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery groupByKorjegyzosegekFoldhivatalokSzama() Group by the korjegyzosegek\_foldhivatalok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery groupByKozmuszolgaltatasok() Group by the kozmuszolgaltatasok column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery groupByMunkaugyiSzolgaltatasokSzama() Group by the munkaugyi\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery groupByBolcsodeiSzolgaltatasokSzama() Group by the bolcsodei\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery groupByMobiltelefonUgyintezesiSzolgaltatasokSzama() Group by the mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama column

\*

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgUgyintezesAltalanosJellemzeseQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgUgyintezesAltalanosJellemzese findOne(ConnectionInterface $con = null) Return the first ChildStgUgyintezesAltalanosJellemzese matching the query

\* @method ChildStgUgyintezesAltalanosJellemzese findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgUgyintezesAltalanosJellemzese matching the query, or a new ChildStgUgyintezesAltalanosJellemzese object populated from the query conditions when no match is found

\*

\* @method ChildStgUgyintezesAltalanosJellemzese findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgUgyintezesAltalanosJellemzese filtered by the telepules\_nev column

\* @method ChildStgUgyintezesAltalanosJellemzese findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgUgyintezesAltalanosJellemzese filtered by the telepules\_KSHKOD column

\* @method ChildStgUgyintezesAltalanosJellemzese findOneByEgeszsegugyiEllatasokSzama(int $egeszsegugyi\_ellatasok\_szama) Return the first ChildStgUgyintezesAltalanosJellemzese filtered by the egeszsegugyi\_ellatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese findOneByPostaiGyogyszertariSzolgaltatasokSzama(int $postai\_gyogyszertari\_szolgaltatasok\_szama) Return the first ChildStgUgyintezesAltalanosJellemzese filtered by the postai\_gyogyszertari\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese findOneByBuntetoperekSzama(int $buntetoperek\_szama) Return the first ChildStgUgyintezesAltalanosJellemzese filtered by the buntetoperek\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese findOneByKereskedelmiSzolgaltatasokSzama(int $kereskedelmi\_szolgaltatasok\_szama) Return the first ChildStgUgyintezesAltalanosJellemzese filtered by the kereskedelmi\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese findOneBySzocialisSzolgaltatasokSzama(int $szocialis\_szolgaltatasok\_szama) Return the first ChildStgUgyintezesAltalanosJellemzese filtered by the szocialis\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese findOneByOkmanyirodaiUgyintezesekSzama(int $okmanyirodai\_ugyintezesek\_szama) Return the first ChildStgUgyintezesAltalanosJellemzese filtered by the okmanyirodai\_ugyintezesek\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese findOneByBankokSzama(int $bankok\_szama) Return the first ChildStgUgyintezesAltalanosJellemzese filtered by the bankok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese findOneByBenzinkutakSzama(int $benzinkutak\_szama) Return the first ChildStgUgyintezesAltalanosJellemzese filtered by the benzinkutak\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese findOneByKorjegyzosegekFoldhivatalokSzama(int $korjegyzosegek\_foldhivatalok\_szama) Return the first ChildStgUgyintezesAltalanosJellemzese filtered by the korjegyzosegek\_foldhivatalok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese findOneByKozmuszolgaltatasok(int $kozmuszolgaltatasok) Return the first ChildStgUgyintezesAltalanosJellemzese filtered by the kozmuszolgaltatasok column

\* @method ChildStgUgyintezesAltalanosJellemzese findOneByMunkaugyiSzolgaltatasokSzama(int $munkaugyi\_szolgaltatasok\_szama) Return the first ChildStgUgyintezesAltalanosJellemzese filtered by the munkaugyi\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese findOneByBolcsodeiSzolgaltatasokSzama(int $bolcsodei\_szolgaltatasok\_szama) Return the first ChildStgUgyintezesAltalanosJellemzese filtered by the bolcsodei\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese findOneByMobiltelefonUgyintezesiSzolgaltatasokSzama(int $mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama) Return the first ChildStgUgyintezesAltalanosJellemzese filtered by the mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama column

\*

\* @method ChildStgUgyintezesAltalanosJellemzese[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgUgyintezesAltalanosJellemzese objects based on current ModelCriteria

\* @method ChildStgUgyintezesAltalanosJellemzese[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgUgyintezesAltalanosJellemzese objects filtered by the telepules\_nev column

\* @method ChildStgUgyintezesAltalanosJellemzese[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgUgyintezesAltalanosJellemzese objects filtered by the telepules\_KSHKOD column

\* @method ChildStgUgyintezesAltalanosJellemzese[]|ObjectCollection findByEgeszsegugyiEllatasokSzama(int $egeszsegugyi\_ellatasok\_szama) Return ChildStgUgyintezesAltalanosJellemzese objects filtered by the egeszsegugyi\_ellatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese[]|ObjectCollection findByPostaiGyogyszertariSzolgaltatasokSzama(int $postai\_gyogyszertari\_szolgaltatasok\_szama) Return ChildStgUgyintezesAltalanosJellemzese objects filtered by the postai\_gyogyszertari\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese[]|ObjectCollection findByBuntetoperekSzama(int $buntetoperek\_szama) Return ChildStgUgyintezesAltalanosJellemzese objects filtered by the buntetoperek\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese[]|ObjectCollection findByKereskedelmiSzolgaltatasokSzama(int $kereskedelmi\_szolgaltatasok\_szama) Return ChildStgUgyintezesAltalanosJellemzese objects filtered by the kereskedelmi\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese[]|ObjectCollection findBySzocialisSzolgaltatasokSzama(int $szocialis\_szolgaltatasok\_szama) Return ChildStgUgyintezesAltalanosJellemzese objects filtered by the szocialis\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese[]|ObjectCollection findByOkmanyirodaiUgyintezesekSzama(int $okmanyirodai\_ugyintezesek\_szama) Return ChildStgUgyintezesAltalanosJellemzese objects filtered by the okmanyirodai\_ugyintezesek\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese[]|ObjectCollection findByBankokSzama(int $bankok\_szama) Return ChildStgUgyintezesAltalanosJellemzese objects filtered by the bankok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese[]|ObjectCollection findByBenzinkutakSzama(int $benzinkutak\_szama) Return ChildStgUgyintezesAltalanosJellemzese objects filtered by the benzinkutak\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese[]|ObjectCollection findByKorjegyzosegekFoldhivatalokSzama(int $korjegyzosegek\_foldhivatalok\_szama) Return ChildStgUgyintezesAltalanosJellemzese objects filtered by the korjegyzosegek\_foldhivatalok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese[]|ObjectCollection findByKozmuszolgaltatasok(int $kozmuszolgaltatasok) Return ChildStgUgyintezesAltalanosJellemzese objects filtered by the kozmuszolgaltatasok column

\* @method ChildStgUgyintezesAltalanosJellemzese[]|ObjectCollection findByMunkaugyiSzolgaltatasokSzama(int $munkaugyi\_szolgaltatasok\_szama) Return ChildStgUgyintezesAltalanosJellemzese objects filtered by the munkaugyi\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese[]|ObjectCollection findByBolcsodeiSzolgaltatasokSzama(int $bolcsodei\_szolgaltatasok\_szama) Return ChildStgUgyintezesAltalanosJellemzese objects filtered by the bolcsodei\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese[]|ObjectCollection findByMobiltelefonUgyintezesiSzolgaltatasokSzama(int $mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama) Return ChildStgUgyintezesAltalanosJellemzese objects filtered by the mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama column

\* @method ChildStgUgyintezesAltalanosJellemzese[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgUgyintezesAltalanosJellemzeseQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgUgyintezesAltalanosJellemzeseQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgUgyintezesAltalanosJellemzese', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgUgyintezesAltalanosJellemzeseQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgUgyintezesAltalanosJellemzeseQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgUgyintezesAltalanosJellemzeseQuery) {

return $criteria;

}

$query = new ChildStgUgyintezesAltalanosJellemzeseQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgUgyintezesAltalanosJellemzese|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgUgyintezesAltalanosJellemzese object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgUgyintezesAltalanosJellemzese object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgUgyintezesAltalanosJellemzese object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgUgyintezesAltalanosJellemzese object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the egeszsegugyi\_ellatasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByEgeszsegugyiEllatasokSzama(1234); // WHERE egeszsegugyi\_ellatasok\_szama = 1234

\* $query->filterByEgeszsegugyiEllatasokSzama(array(12, 34)); // WHERE egeszsegugyi\_ellatasok\_szama IN (12, 34)

\* $query->filterByEgeszsegugyiEllatasokSzama(array('min' => 12)); // WHERE egeszsegugyi\_ellatasok\_szama > 12

\* </code>

\*

\* @param mixed $egeszsegugyiEllatasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByEgeszsegugyiEllatasokSzama($egeszsegugyiEllatasokSzama = null, $comparison = null)

{

if (is\_array($egeszsegugyiEllatasokSzama)) {

$useMinMax = false;

if (isset($egeszsegugyiEllatasokSzama['min'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_EGESZSEGUGYI\_ELLATASOK\_SZAMA, $egeszsegugyiEllatasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($egeszsegugyiEllatasokSzama['max'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_EGESZSEGUGYI\_ELLATASOK\_SZAMA, $egeszsegugyiEllatasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_EGESZSEGUGYI\_ELLATASOK\_SZAMA, $egeszsegugyiEllatasokSzama, $comparison);

}

/\*\*

\* Filter the query on the postai\_gyogyszertari\_szolgaltatasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByPostaiGyogyszertariSzolgaltatasokSzama(1234); // WHERE postai\_gyogyszertari\_szolgaltatasok\_szama = 1234

\* $query->filterByPostaiGyogyszertariSzolgaltatasokSzama(array(12, 34)); // WHERE postai\_gyogyszertari\_szolgaltatasok\_szama IN (12, 34)

\* $query->filterByPostaiGyogyszertariSzolgaltatasokSzama(array('min' => 12)); // WHERE postai\_gyogyszertari\_szolgaltatasok\_szama > 12

\* </code>

\*

\* @param mixed $postaiGyogyszertariSzolgaltatasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByPostaiGyogyszertariSzolgaltatasokSzama($postaiGyogyszertariSzolgaltatasokSzama = null, $comparison = null)

{

if (is\_array($postaiGyogyszertariSzolgaltatasokSzama)) {

$useMinMax = false;

if (isset($postaiGyogyszertariSzolgaltatasokSzama['min'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA, $postaiGyogyszertariSzolgaltatasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($postaiGyogyszertariSzolgaltatasokSzama['max'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA, $postaiGyogyszertariSzolgaltatasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA, $postaiGyogyszertariSzolgaltatasokSzama, $comparison);

}

/\*\*

\* Filter the query on the buntetoperek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByBuntetoperekSzama(1234); // WHERE buntetoperek\_szama = 1234

\* $query->filterByBuntetoperekSzama(array(12, 34)); // WHERE buntetoperek\_szama IN (12, 34)

\* $query->filterByBuntetoperekSzama(array('min' => 12)); // WHERE buntetoperek\_szama > 12

\* </code>

\*

\* @param mixed $buntetoperekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByBuntetoperekSzama($buntetoperekSzama = null, $comparison = null)

{

if (is\_array($buntetoperekSzama)) {

$useMinMax = false;

if (isset($buntetoperekSzama['min'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BUNTETOPEREK\_SZAMA, $buntetoperekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($buntetoperekSzama['max'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BUNTETOPEREK\_SZAMA, $buntetoperekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BUNTETOPEREK\_SZAMA, $buntetoperekSzama, $comparison);

}

/\*\*

\* Filter the query on the kereskedelmi\_szolgaltatasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByKereskedelmiSzolgaltatasokSzama(1234); // WHERE kereskedelmi\_szolgaltatasok\_szama = 1234

\* $query->filterByKereskedelmiSzolgaltatasokSzama(array(12, 34)); // WHERE kereskedelmi\_szolgaltatasok\_szama IN (12, 34)

\* $query->filterByKereskedelmiSzolgaltatasokSzama(array('min' => 12)); // WHERE kereskedelmi\_szolgaltatasok\_szama > 12

\* </code>

\*

\* @param mixed $kereskedelmiSzolgaltatasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByKereskedelmiSzolgaltatasokSzama($kereskedelmiSzolgaltatasokSzama = null, $comparison = null)

{

if (is\_array($kereskedelmiSzolgaltatasokSzama)) {

$useMinMax = false;

if (isset($kereskedelmiSzolgaltatasokSzama['min'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KERESKEDELMI\_SZOLGALTATASOK\_SZAMA, $kereskedelmiSzolgaltatasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($kereskedelmiSzolgaltatasokSzama['max'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KERESKEDELMI\_SZOLGALTATASOK\_SZAMA, $kereskedelmiSzolgaltatasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KERESKEDELMI\_SZOLGALTATASOK\_SZAMA, $kereskedelmiSzolgaltatasokSzama, $comparison);

}

/\*\*

\* Filter the query on the szocialis\_szolgaltatasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterBySzocialisSzolgaltatasokSzama(1234); // WHERE szocialis\_szolgaltatasok\_szama = 1234

\* $query->filterBySzocialisSzolgaltatasokSzama(array(12, 34)); // WHERE szocialis\_szolgaltatasok\_szama IN (12, 34)

\* $query->filterBySzocialisSzolgaltatasokSzama(array('min' => 12)); // WHERE szocialis\_szolgaltatasok\_szama > 12

\* </code>

\*

\* @param mixed $szocialisSzolgaltatasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterBySzocialisSzolgaltatasokSzama($szocialisSzolgaltatasokSzama = null, $comparison = null)

{

if (is\_array($szocialisSzolgaltatasokSzama)) {

$useMinMax = false;

if (isset($szocialisSzolgaltatasokSzama['min'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_SZOCIALIS\_SZOLGALTATASOK\_SZAMA, $szocialisSzolgaltatasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($szocialisSzolgaltatasokSzama['max'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_SZOCIALIS\_SZOLGALTATASOK\_SZAMA, $szocialisSzolgaltatasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_SZOCIALIS\_SZOLGALTATASOK\_SZAMA, $szocialisSzolgaltatasokSzama, $comparison);

}

/\*\*

\* Filter the query on the okmanyirodai\_ugyintezesek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByOkmanyirodaiUgyintezesekSzama(1234); // WHERE okmanyirodai\_ugyintezesek\_szama = 1234

\* $query->filterByOkmanyirodaiUgyintezesekSzama(array(12, 34)); // WHERE okmanyirodai\_ugyintezesek\_szama IN (12, 34)

\* $query->filterByOkmanyirodaiUgyintezesekSzama(array('min' => 12)); // WHERE okmanyirodai\_ugyintezesek\_szama > 12

\* </code>

\*

\* @param mixed $okmanyirodaiUgyintezesekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByOkmanyirodaiUgyintezesekSzama($okmanyirodaiUgyintezesekSzama = null, $comparison = null)

{

if (is\_array($okmanyirodaiUgyintezesekSzama)) {

$useMinMax = false;

if (isset($okmanyirodaiUgyintezesekSzama['min'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_OKMANYIRODAI\_UGYINTEZESEK\_SZAMA, $okmanyirodaiUgyintezesekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($okmanyirodaiUgyintezesekSzama['max'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_OKMANYIRODAI\_UGYINTEZESEK\_SZAMA, $okmanyirodaiUgyintezesekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_OKMANYIRODAI\_UGYINTEZESEK\_SZAMA, $okmanyirodaiUgyintezesekSzama, $comparison);

}

/\*\*

\* Filter the query on the bankok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByBankokSzama(1234); // WHERE bankok\_szama = 1234

\* $query->filterByBankokSzama(array(12, 34)); // WHERE bankok\_szama IN (12, 34)

\* $query->filterByBankokSzama(array('min' => 12)); // WHERE bankok\_szama > 12

\* </code>

\*

\* @param mixed $bankokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByBankokSzama($bankokSzama = null, $comparison = null)

{

if (is\_array($bankokSzama)) {

$useMinMax = false;

if (isset($bankokSzama['min'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BANKOK\_SZAMA, $bankokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($bankokSzama['max'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BANKOK\_SZAMA, $bankokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BANKOK\_SZAMA, $bankokSzama, $comparison);

}

/\*\*

\* Filter the query on the benzinkutak\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByBenzinkutakSzama(1234); // WHERE benzinkutak\_szama = 1234

\* $query->filterByBenzinkutakSzama(array(12, 34)); // WHERE benzinkutak\_szama IN (12, 34)

\* $query->filterByBenzinkutakSzama(array('min' => 12)); // WHERE benzinkutak\_szama > 12

\* </code>

\*

\* @param mixed $benzinkutakSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByBenzinkutakSzama($benzinkutakSzama = null, $comparison = null)

{

if (is\_array($benzinkutakSzama)) {

$useMinMax = false;

if (isset($benzinkutakSzama['min'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BENZINKUTAK\_SZAMA, $benzinkutakSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($benzinkutakSzama['max'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BENZINKUTAK\_SZAMA, $benzinkutakSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BENZINKUTAK\_SZAMA, $benzinkutakSzama, $comparison);

}

/\*\*

\* Filter the query on the korjegyzosegek\_foldhivatalok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByKorjegyzosegekFoldhivatalokSzama(1234); // WHERE korjegyzosegek\_foldhivatalok\_szama = 1234

\* $query->filterByKorjegyzosegekFoldhivatalokSzama(array(12, 34)); // WHERE korjegyzosegek\_foldhivatalok\_szama IN (12, 34)

\* $query->filterByKorjegyzosegekFoldhivatalokSzama(array('min' => 12)); // WHERE korjegyzosegek\_foldhivatalok\_szama > 12

\* </code>

\*

\* @param mixed $korjegyzosegekFoldhivatalokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByKorjegyzosegekFoldhivatalokSzama($korjegyzosegekFoldhivatalokSzama = null, $comparison = null)

{

if (is\_array($korjegyzosegekFoldhivatalokSzama)) {

$useMinMax = false;

if (isset($korjegyzosegekFoldhivatalokSzama['min'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA, $korjegyzosegekFoldhivatalokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($korjegyzosegekFoldhivatalokSzama['max'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA, $korjegyzosegekFoldhivatalokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA, $korjegyzosegekFoldhivatalokSzama, $comparison);

}

/\*\*

\* Filter the query on the kozmuszolgaltatasok column

\*

\* Example usage:

\* <code>

\* $query->filterByKozmuszolgaltatasok(1234); // WHERE kozmuszolgaltatasok = 1234

\* $query->filterByKozmuszolgaltatasok(array(12, 34)); // WHERE kozmuszolgaltatasok IN (12, 34)

\* $query->filterByKozmuszolgaltatasok(array('min' => 12)); // WHERE kozmuszolgaltatasok > 12

\* </code>

\*

\* @param mixed $kozmuszolgaltatasok The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByKozmuszolgaltatasok($kozmuszolgaltatasok = null, $comparison = null)

{

if (is\_array($kozmuszolgaltatasok)) {

$useMinMax = false;

if (isset($kozmuszolgaltatasok['min'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KOZMUSZOLGALTATASOK, $kozmuszolgaltatasok['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($kozmuszolgaltatasok['max'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KOZMUSZOLGALTATASOK, $kozmuszolgaltatasok['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KOZMUSZOLGALTATASOK, $kozmuszolgaltatasok, $comparison);

}

/\*\*

\* Filter the query on the munkaugyi\_szolgaltatasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByMunkaugyiSzolgaltatasokSzama(1234); // WHERE munkaugyi\_szolgaltatasok\_szama = 1234

\* $query->filterByMunkaugyiSzolgaltatasokSzama(array(12, 34)); // WHERE munkaugyi\_szolgaltatasok\_szama IN (12, 34)

\* $query->filterByMunkaugyiSzolgaltatasokSzama(array('min' => 12)); // WHERE munkaugyi\_szolgaltatasok\_szama > 12

\* </code>

\*

\* @param mixed $munkaugyiSzolgaltatasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByMunkaugyiSzolgaltatasokSzama($munkaugyiSzolgaltatasokSzama = null, $comparison = null)

{

if (is\_array($munkaugyiSzolgaltatasokSzama)) {

$useMinMax = false;

if (isset($munkaugyiSzolgaltatasokSzama['min'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_MUNKAUGYI\_SZOLGALTATASOK\_SZAMA, $munkaugyiSzolgaltatasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($munkaugyiSzolgaltatasokSzama['max'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_MUNKAUGYI\_SZOLGALTATASOK\_SZAMA, $munkaugyiSzolgaltatasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_MUNKAUGYI\_SZOLGALTATASOK\_SZAMA, $munkaugyiSzolgaltatasokSzama, $comparison);

}

/\*\*

\* Filter the query on the bolcsodei\_szolgaltatasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByBolcsodeiSzolgaltatasokSzama(1234); // WHERE bolcsodei\_szolgaltatasok\_szama = 1234

\* $query->filterByBolcsodeiSzolgaltatasokSzama(array(12, 34)); // WHERE bolcsodei\_szolgaltatasok\_szama IN (12, 34)

\* $query->filterByBolcsodeiSzolgaltatasokSzama(array('min' => 12)); // WHERE bolcsodei\_szolgaltatasok\_szama > 12

\* </code>

\*

\* @param mixed $bolcsodeiSzolgaltatasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByBolcsodeiSzolgaltatasokSzama($bolcsodeiSzolgaltatasokSzama = null, $comparison = null)

{

if (is\_array($bolcsodeiSzolgaltatasokSzama)) {

$useMinMax = false;

if (isset($bolcsodeiSzolgaltatasokSzama['min'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BOLCSODEI\_SZOLGALTATASOK\_SZAMA, $bolcsodeiSzolgaltatasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($bolcsodeiSzolgaltatasokSzama['max'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BOLCSODEI\_SZOLGALTATASOK\_SZAMA, $bolcsodeiSzolgaltatasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BOLCSODEI\_SZOLGALTATASOK\_SZAMA, $bolcsodeiSzolgaltatasokSzama, $comparison);

}

/\*\*

\* Filter the query on the mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByMobiltelefonUgyintezesiSzolgaltatasokSzama(1234); // WHERE mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama = 1234

\* $query->filterByMobiltelefonUgyintezesiSzolgaltatasokSzama(array(12, 34)); // WHERE mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama IN (12, 34)

\* $query->filterByMobiltelefonUgyintezesiSzolgaltatasokSzama(array('min' => 12)); // WHERE mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama > 12

\* </code>

\*

\* @param mixed $mobiltelefonUgyintezesiSzolgaltatasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function filterByMobiltelefonUgyintezesiSzolgaltatasokSzama($mobiltelefonUgyintezesiSzolgaltatasokSzama = null, $comparison = null)

{

if (is\_array($mobiltelefonUgyintezesiSzolgaltatasokSzama)) {

$useMinMax = false;

if (isset($mobiltelefonUgyintezesiSzolgaltatasokSzama['min'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA, $mobiltelefonUgyintezesiSzolgaltatasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($mobiltelefonUgyintezesiSzolgaltatasokSzama['max'])) {

$this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA, $mobiltelefonUgyintezesiSzolgaltatasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUgyintezesAltalanosJellemzeseTableMap::COL\_MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA, $mobiltelefonUgyintezesiSzolgaltatasokSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgUgyintezesAltalanosJellemzese $stgUgyintezesAltalanosJellemzese Object to remove from the list of results

\*

\* @return $this|ChildStgUgyintezesAltalanosJellemzeseQuery The current query, for fluid interface

\*/

public function prune($stgUgyintezesAltalanosJellemzese = null)

{

if ($stgUgyintezesAltalanosJellemzese) {

throw new LogicException('StgUgyintezesAltalanosJellemzese object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_ugyintezes\_altalanos\_jellemzese table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgUgyintezesAltalanosJellemzeseTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgUgyintezesAltalanosJellemzeseTableMap::clearInstancePool();

StgUgyintezesAltalanosJellemzeseTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgUgyintezesAltalanosJellemzeseTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgUgyintezesAltalanosJellemzeseTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgUgyintezesAltalanosJellemzeseTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgUgyintezesAltalanosJellemzeseTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgUgyintezesAltalanosJellemzeseQuery

#### StgUzletekVendeglatohelyekGyogyszertarak.php

<?php

namespace Base;

use \StgUzletekVendeglatohelyekGyogyszertarakQuery as ChildStgUzletekVendeglatohelyekGyogyszertarakQuery;

use \Exception;

use \PDO;

use Map\StgUzletekVendeglatohelyekGyogyszertarakTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgUzletekVendeglatohelyekGyogyszertarak implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgUzletekVendeglatohelyekGyogyszertarakTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the elelmiszer\_vegyesuzletek\_aruhazak\_szama field.

\* @var int

\*/

protected $elelmiszer\_vegyesuzletek\_aruhazak\_szama;

/\*\*

\* The value for the kiskereskedelmi\_uzletek\_szama field.

\* @var int

\*/

protected $kiskereskedelmi\_uzletek\_szama;

/\*\*

\* The value for the vendeglatohelyek\_szama field.

\* @var int

\*/

protected $vendeglatohelyek\_szama;

/\*\*

\* The value for the fiokgyogyszertarak\_szama field.

\* @var int

\*/

protected $fiokgyogyszertarak\_szama;

/\*\*

\* The value for the gyogyszertarak\_szama field.

\* @var int

\*/

protected $gyogyszertarak\_szama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgUzletekVendeglatohelyekGyogyszertarak object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgUzletekVendeglatohelyekGyogyszertarak</code> instance. If

\* <code>obj</code> is an instance of <code>StgUzletekVendeglatohelyekGyogyszertarak</code>, delegates to

\* <code>equals(StgUzletekVendeglatohelyekGyogyszertarak)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgUzletekVendeglatohelyekGyogyszertarak The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [elelmiszer\_vegyesuzletek\_aruhazak\_szama] column value.

\*

\* @return int

\*/

public function getElelmiszerVegyesuzletekAruhazakSzama()

{

return $this->elelmiszer\_vegyesuzletek\_aruhazak\_szama;

}

/\*\*

\* Get the [kiskereskedelmi\_uzletek\_szama] column value.

\*

\* @return int

\*/

public function getKiskereskedelmiUzletekSzama()

{

return $this->kiskereskedelmi\_uzletek\_szama;

}

/\*\*

\* Get the [vendeglatohelyek\_szama] column value.

\*

\* @return int

\*/

public function getVendeglatohelyekSzama()

{

return $this->vendeglatohelyek\_szama;

}

/\*\*

\* Get the [fiokgyogyszertarak\_szama] column value.

\*

\* @return int

\*/

public function getFiokgyogyszertarakSzama()

{

return $this->fiokgyogyszertarak\_szama;

}

/\*\*

\* Get the [gyogyszertarak\_szama] column value.

\*

\* @return int

\*/

public function getGyogyszertarakSzama()

{

return $this->gyogyszertarak\_szama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgUzletekVendeglatohelyekGyogyszertarakTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgUzletekVendeglatohelyekGyogyszertarakTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgUzletekVendeglatohelyekGyogyszertarakTableMap::translateFieldName('ElelmiszerVegyesuzletekAruhazakSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->elelmiszer\_vegyesuzletek\_aruhazak\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgUzletekVendeglatohelyekGyogyszertarakTableMap::translateFieldName('KiskereskedelmiUzletekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->kiskereskedelmi\_uzletek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgUzletekVendeglatohelyekGyogyszertarakTableMap::translateFieldName('VendeglatohelyekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->vendeglatohelyek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 5 + $startcol : StgUzletekVendeglatohelyekGyogyszertarakTableMap::translateFieldName('FiokgyogyszertarakSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->fiokgyogyszertarak\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 6 + $startcol : StgUzletekVendeglatohelyekGyogyszertarakTableMap::translateFieldName('GyogyszertarakSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->gyogyszertarak\_szama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 7; // 7 = StgUzletekVendeglatohelyekGyogyszertarakTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgUzletekVendeglatohelyekGyogyszertarak'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgUzletekVendeglatohelyekGyogyszertarak The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgUzletekVendeglatohelyekGyogyszertarak The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [elelmiszer\_vegyesuzletek\_aruhazak\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgUzletekVendeglatohelyekGyogyszertarak The current object (for fluent API support)

\*/

public function setElelmiszerVegyesuzletekAruhazakSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->elelmiszer\_vegyesuzletek\_aruhazak\_szama !== $v) {

$this->elelmiszer\_vegyesuzletek\_aruhazak\_szama = $v;

$this->modifiedColumns[StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA] = true;

}

return $this;

} // setElelmiszerVegyesuzletekAruhazakSzama()

/\*\*

\* Set the value of [kiskereskedelmi\_uzletek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgUzletekVendeglatohelyekGyogyszertarak The current object (for fluent API support)

\*/

public function setKiskereskedelmiUzletekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->kiskereskedelmi\_uzletek\_szama !== $v) {

$this->kiskereskedelmi\_uzletek\_szama = $v;

$this->modifiedColumns[StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA] = true;

}

return $this;

} // setKiskereskedelmiUzletekSzama()

/\*\*

\* Set the value of [vendeglatohelyek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgUzletekVendeglatohelyekGyogyszertarak The current object (for fluent API support)

\*/

public function setVendeglatohelyekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->vendeglatohelyek\_szama !== $v) {

$this->vendeglatohelyek\_szama = $v;

$this->modifiedColumns[StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_VENDEGLATOHELYEK\_SZAMA] = true;

}

return $this;

} // setVendeglatohelyekSzama()

/\*\*

\* Set the value of [fiokgyogyszertarak\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgUzletekVendeglatohelyekGyogyszertarak The current object (for fluent API support)

\*/

public function setFiokgyogyszertarakSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->fiokgyogyszertarak\_szama !== $v) {

$this->fiokgyogyszertarak\_szama = $v;

$this->modifiedColumns[StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_FIOKGYOGYSZERTARAK\_SZAMA] = true;

}

return $this;

} // setFiokgyogyszertarakSzama()

/\*\*

\* Set the value of [gyogyszertarak\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgUzletekVendeglatohelyekGyogyszertarak The current object (for fluent API support)

\*/

public function setGyogyszertarakSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->gyogyszertarak\_szama !== $v) {

$this->gyogyszertarak\_szama = $v;

$this->modifiedColumns[StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_GYOGYSZERTARAK\_SZAMA] = true;

}

return $this;

} // setGyogyszertarakSzama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgUzletekVendeglatohelyekGyogyszertarakTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgUzletekVendeglatohelyekGyogyszertarakQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgUzletekVendeglatohelyekGyogyszertarak::setDeleted()

\* @see StgUzletekVendeglatohelyekGyogyszertarak::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgUzletekVendeglatohelyekGyogyszertarakTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgUzletekVendeglatohelyekGyogyszertarakQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgUzletekVendeglatohelyekGyogyszertarakTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgUzletekVendeglatohelyekGyogyszertarakTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA';

}

if ($this->isColumnModified(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'KISKERESKEDELMI\_UZLETEK\_SZAMA';

}

if ($this->isColumnModified(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_VENDEGLATOHELYEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'VENDEGLATOHELYEK\_SZAMA';

}

if ($this->isColumnModified(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_FIOKGYOGYSZERTARAK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'FIOKGYOGYSZERTARAK\_SZAMA';

}

if ($this->isColumnModified(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_GYOGYSZERTARAK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'GYOGYSZERTARAK\_SZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_uzletek\_vendeglatohelyek\_gyogyszertarak (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA':

$stmt->bindValue($identifier, $this->elelmiszer\_vegyesuzletek\_aruhazak\_szama, PDO::PARAM\_INT);

break;

case 'KISKERESKEDELMI\_UZLETEK\_SZAMA':

$stmt->bindValue($identifier, $this->kiskereskedelmi\_uzletek\_szama, PDO::PARAM\_INT);

break;

case 'VENDEGLATOHELYEK\_SZAMA':

$stmt->bindValue($identifier, $this->vendeglatohelyek\_szama, PDO::PARAM\_INT);

break;

case 'FIOKGYOGYSZERTARAK\_SZAMA':

$stmt->bindValue($identifier, $this->fiokgyogyszertarak\_szama, PDO::PARAM\_INT);

break;

case 'GYOGYSZERTARAK\_SZAMA':

$stmt->bindValue($identifier, $this->gyogyszertarak\_szama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgUzletekVendeglatohelyekGyogyszertarakTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getElelmiszerVegyesuzletekAruhazakSzama();

break;

case 3:

return $this->getKiskereskedelmiUzletekSzama();

break;

case 4:

return $this->getVendeglatohelyekSzama();

break;

case 5:

return $this->getFiokgyogyszertarakSzama();

break;

case 6:

return $this->getGyogyszertarakSzama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgUzletekVendeglatohelyekGyogyszertarak'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgUzletekVendeglatohelyekGyogyszertarak'][$this->getPrimaryKey()] = true;

$keys = StgUzletekVendeglatohelyekGyogyszertarakTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getElelmiszerVegyesuzletekAruhazakSzama(),

$keys[3] => $this->getKiskereskedelmiUzletekSzama(),

$keys[4] => $this->getVendeglatohelyekSzama(),

$keys[5] => $this->getFiokgyogyszertarakSzama(),

$keys[6] => $this->getGyogyszertarakSzama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgUzletekVendeglatohelyekGyogyszertarak

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgUzletekVendeglatohelyekGyogyszertarakTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgUzletekVendeglatohelyekGyogyszertarak

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setElelmiszerVegyesuzletekAruhazakSzama($value);

break;

case 3:

$this->setKiskereskedelmiUzletekSzama($value);

break;

case 4:

$this->setVendeglatohelyekSzama($value);

break;

case 5:

$this->setFiokgyogyszertarakSzama($value);

break;

case 6:

$this->setGyogyszertarakSzama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgUzletekVendeglatohelyekGyogyszertarakTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setElelmiszerVegyesuzletekAruhazakSzama($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setKiskereskedelmiUzletekSzama($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setVendeglatohelyekSzama($arr[$keys[4]]);

}

if (array\_key\_exists($keys[5], $arr)) {

$this->setFiokgyogyszertarakSzama($arr[$keys[5]]);

}

if (array\_key\_exists($keys[6], $arr)) {

$this->setGyogyszertarakSzama($arr[$keys[6]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgUzletekVendeglatohelyekGyogyszertarak The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgUzletekVendeglatohelyekGyogyszertarakTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA)) {

$criteria->add(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA, $this->elelmiszer\_vegyesuzletek\_aruhazak\_szama);

}

if ($this->isColumnModified(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA)) {

$criteria->add(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA, $this->kiskereskedelmi\_uzletek\_szama);

}

if ($this->isColumnModified(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_VENDEGLATOHELYEK\_SZAMA)) {

$criteria->add(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_VENDEGLATOHELYEK\_SZAMA, $this->vendeglatohelyek\_szama);

}

if ($this->isColumnModified(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_FIOKGYOGYSZERTARAK\_SZAMA)) {

$criteria->add(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_FIOKGYOGYSZERTARAK\_SZAMA, $this->fiokgyogyszertarak\_szama);

}

if ($this->isColumnModified(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_GYOGYSZERTARAK\_SZAMA)) {

$criteria->add(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_GYOGYSZERTARAK\_SZAMA, $this->gyogyszertarak\_szama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgUzletekVendeglatohelyekGyogyszertarak object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgUzletekVendeglatohelyekGyogyszertarak (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setElelmiszerVegyesuzletekAruhazakSzama($this->getElelmiszerVegyesuzletekAruhazakSzama());

$copyObj->setKiskereskedelmiUzletekSzama($this->getKiskereskedelmiUzletekSzama());

$copyObj->setVendeglatohelyekSzama($this->getVendeglatohelyekSzama());

$copyObj->setFiokgyogyszertarakSzama($this->getFiokgyogyszertarakSzama());

$copyObj->setGyogyszertarakSzama($this->getGyogyszertarakSzama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgUzletekVendeglatohelyekGyogyszertarak Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->elelmiszer\_vegyesuzletek\_aruhazak\_szama = null;

$this->kiskereskedelmi\_uzletek\_szama = null;

$this->vendeglatohelyek\_szama = null;

$this->fiokgyogyszertarak\_szama = null;

$this->gyogyszertarak\_szama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgUzletekVendeglatohelyekGyogyszertarakTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgUzletekVendeglatohelyekGyogyszertarakQuery.php

<?php

namespace Base;

use \StgUzletekVendeglatohelyekGyogyszertarak as ChildStgUzletekVendeglatohelyekGyogyszertarak;

use \StgUzletekVendeglatohelyekGyogyszertarakQuery as ChildStgUzletekVendeglatohelyekGyogyszertarakQuery;

use \Exception;

use Map\StgUzletekVendeglatohelyekGyogyszertarakTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_uzletek\_vendeglatohelyek\_gyogyszertarak' table.

\*

\*

\*

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarakQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarakQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarakQuery orderByElelmiszerVegyesuzletekAruhazakSzama($order = Criteria::ASC) Order by the elelmiszer\_vegyesuzletek\_aruhazak\_szama column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarakQuery orderByKiskereskedelmiUzletekSzama($order = Criteria::ASC) Order by the kiskereskedelmi\_uzletek\_szama column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarakQuery orderByVendeglatohelyekSzama($order = Criteria::ASC) Order by the vendeglatohelyek\_szama column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarakQuery orderByFiokgyogyszertarakSzama($order = Criteria::ASC) Order by the fiokgyogyszertarak\_szama column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarakQuery orderByGyogyszertarakSzama($order = Criteria::ASC) Order by the gyogyszertarak\_szama column

\*

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarakQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarakQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarakQuery groupByElelmiszerVegyesuzletekAruhazakSzama() Group by the elelmiszer\_vegyesuzletek\_aruhazak\_szama column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarakQuery groupByKiskereskedelmiUzletekSzama() Group by the kiskereskedelmi\_uzletek\_szama column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarakQuery groupByVendeglatohelyekSzama() Group by the vendeglatohelyek\_szama column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarakQuery groupByFiokgyogyszertarakSzama() Group by the fiokgyogyszertarak\_szama column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarakQuery groupByGyogyszertarakSzama() Group by the gyogyszertarak\_szama column

\*

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarakQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarakQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarakQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak findOne(ConnectionInterface $con = null) Return the first ChildStgUzletekVendeglatohelyekGyogyszertarak matching the query

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgUzletekVendeglatohelyekGyogyszertarak matching the query, or a new ChildStgUzletekVendeglatohelyekGyogyszertarak object populated from the query conditions when no match is found

\*

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgUzletekVendeglatohelyekGyogyszertarak filtered by the telepules\_nev column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgUzletekVendeglatohelyekGyogyszertarak filtered by the telepules\_KSHKOD column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak findOneByElelmiszerVegyesuzletekAruhazakSzama(int $elelmiszer\_vegyesuzletek\_aruhazak\_szama) Return the first ChildStgUzletekVendeglatohelyekGyogyszertarak filtered by the elelmiszer\_vegyesuzletek\_aruhazak\_szama column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak findOneByKiskereskedelmiUzletekSzama(int $kiskereskedelmi\_uzletek\_szama) Return the first ChildStgUzletekVendeglatohelyekGyogyszertarak filtered by the kiskereskedelmi\_uzletek\_szama column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak findOneByVendeglatohelyekSzama(int $vendeglatohelyek\_szama) Return the first ChildStgUzletekVendeglatohelyekGyogyszertarak filtered by the vendeglatohelyek\_szama column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak findOneByFiokgyogyszertarakSzama(int $fiokgyogyszertarak\_szama) Return the first ChildStgUzletekVendeglatohelyekGyogyszertarak filtered by the fiokgyogyszertarak\_szama column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak findOneByGyogyszertarakSzama(int $gyogyszertarak\_szama) Return the first ChildStgUzletekVendeglatohelyekGyogyszertarak filtered by the gyogyszertarak\_szama column

\*

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgUzletekVendeglatohelyekGyogyszertarak objects based on current ModelCriteria

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgUzletekVendeglatohelyekGyogyszertarak objects filtered by the telepules\_nev column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgUzletekVendeglatohelyekGyogyszertarak objects filtered by the telepules\_KSHKOD column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak[]|ObjectCollection findByElelmiszerVegyesuzletekAruhazakSzama(int $elelmiszer\_vegyesuzletek\_aruhazak\_szama) Return ChildStgUzletekVendeglatohelyekGyogyszertarak objects filtered by the elelmiszer\_vegyesuzletek\_aruhazak\_szama column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak[]|ObjectCollection findByKiskereskedelmiUzletekSzama(int $kiskereskedelmi\_uzletek\_szama) Return ChildStgUzletekVendeglatohelyekGyogyszertarak objects filtered by the kiskereskedelmi\_uzletek\_szama column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak[]|ObjectCollection findByVendeglatohelyekSzama(int $vendeglatohelyek\_szama) Return ChildStgUzletekVendeglatohelyekGyogyszertarak objects filtered by the vendeglatohelyek\_szama column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak[]|ObjectCollection findByFiokgyogyszertarakSzama(int $fiokgyogyszertarak\_szama) Return ChildStgUzletekVendeglatohelyekGyogyszertarak objects filtered by the fiokgyogyszertarak\_szama column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak[]|ObjectCollection findByGyogyszertarakSzama(int $gyogyszertarak\_szama) Return ChildStgUzletekVendeglatohelyekGyogyszertarak objects filtered by the gyogyszertarak\_szama column

\* @method ChildStgUzletekVendeglatohelyekGyogyszertarak[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgUzletekVendeglatohelyekGyogyszertarakQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgUzletekVendeglatohelyekGyogyszertarakQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgUzletekVendeglatohelyekGyogyszertarak', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgUzletekVendeglatohelyekGyogyszertarakQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgUzletekVendeglatohelyekGyogyszertarakQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgUzletekVendeglatohelyekGyogyszertarakQuery) {

return $criteria;

}

$query = new ChildStgUzletekVendeglatohelyekGyogyszertarakQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgUzletekVendeglatohelyekGyogyszertarak|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgUzletekVendeglatohelyekGyogyszertarak object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgUzletekVendeglatohelyekGyogyszertarak object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgUzletekVendeglatohelyekGyogyszertarakQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgUzletekVendeglatohelyekGyogyszertarak object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgUzletekVendeglatohelyekGyogyszertarakQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgUzletekVendeglatohelyekGyogyszertarak object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUzletekVendeglatohelyekGyogyszertarakQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUzletekVendeglatohelyekGyogyszertarakQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the elelmiszer\_vegyesuzletek\_aruhazak\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByElelmiszerVegyesuzletekAruhazakSzama(1234); // WHERE elelmiszer\_vegyesuzletek\_aruhazak\_szama = 1234

\* $query->filterByElelmiszerVegyesuzletekAruhazakSzama(array(12, 34)); // WHERE elelmiszer\_vegyesuzletek\_aruhazak\_szama IN (12, 34)

\* $query->filterByElelmiszerVegyesuzletekAruhazakSzama(array('min' => 12)); // WHERE elelmiszer\_vegyesuzletek\_aruhazak\_szama > 12

\* </code>

\*

\* @param mixed $elelmiszerVegyesuzletekAruhazakSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUzletekVendeglatohelyekGyogyszertarakQuery The current query, for fluid interface

\*/

public function filterByElelmiszerVegyesuzletekAruhazakSzama($elelmiszerVegyesuzletekAruhazakSzama = null, $comparison = null)

{

if (is\_array($elelmiszerVegyesuzletekAruhazakSzama)) {

$useMinMax = false;

if (isset($elelmiszerVegyesuzletekAruhazakSzama['min'])) {

$this->addUsingAlias(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA, $elelmiszerVegyesuzletekAruhazakSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($elelmiszerVegyesuzletekAruhazakSzama['max'])) {

$this->addUsingAlias(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA, $elelmiszerVegyesuzletekAruhazakSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA, $elelmiszerVegyesuzletekAruhazakSzama, $comparison);

}

/\*\*

\* Filter the query on the kiskereskedelmi\_uzletek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByKiskereskedelmiUzletekSzama(1234); // WHERE kiskereskedelmi\_uzletek\_szama = 1234

\* $query->filterByKiskereskedelmiUzletekSzama(array(12, 34)); // WHERE kiskereskedelmi\_uzletek\_szama IN (12, 34)

\* $query->filterByKiskereskedelmiUzletekSzama(array('min' => 12)); // WHERE kiskereskedelmi\_uzletek\_szama > 12

\* </code>

\*

\* @param mixed $kiskereskedelmiUzletekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUzletekVendeglatohelyekGyogyszertarakQuery The current query, for fluid interface

\*/

public function filterByKiskereskedelmiUzletekSzama($kiskereskedelmiUzletekSzama = null, $comparison = null)

{

if (is\_array($kiskereskedelmiUzletekSzama)) {

$useMinMax = false;

if (isset($kiskereskedelmiUzletekSzama['min'])) {

$this->addUsingAlias(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA, $kiskereskedelmiUzletekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($kiskereskedelmiUzletekSzama['max'])) {

$this->addUsingAlias(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA, $kiskereskedelmiUzletekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA, $kiskereskedelmiUzletekSzama, $comparison);

}

/\*\*

\* Filter the query on the vendeglatohelyek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByVendeglatohelyekSzama(1234); // WHERE vendeglatohelyek\_szama = 1234

\* $query->filterByVendeglatohelyekSzama(array(12, 34)); // WHERE vendeglatohelyek\_szama IN (12, 34)

\* $query->filterByVendeglatohelyekSzama(array('min' => 12)); // WHERE vendeglatohelyek\_szama > 12

\* </code>

\*

\* @param mixed $vendeglatohelyekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUzletekVendeglatohelyekGyogyszertarakQuery The current query, for fluid interface

\*/

public function filterByVendeglatohelyekSzama($vendeglatohelyekSzama = null, $comparison = null)

{

if (is\_array($vendeglatohelyekSzama)) {

$useMinMax = false;

if (isset($vendeglatohelyekSzama['min'])) {

$this->addUsingAlias(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_VENDEGLATOHELYEK\_SZAMA, $vendeglatohelyekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($vendeglatohelyekSzama['max'])) {

$this->addUsingAlias(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_VENDEGLATOHELYEK\_SZAMA, $vendeglatohelyekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_VENDEGLATOHELYEK\_SZAMA, $vendeglatohelyekSzama, $comparison);

}

/\*\*

\* Filter the query on the fiokgyogyszertarak\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByFiokgyogyszertarakSzama(1234); // WHERE fiokgyogyszertarak\_szama = 1234

\* $query->filterByFiokgyogyszertarakSzama(array(12, 34)); // WHERE fiokgyogyszertarak\_szama IN (12, 34)

\* $query->filterByFiokgyogyszertarakSzama(array('min' => 12)); // WHERE fiokgyogyszertarak\_szama > 12

\* </code>

\*

\* @param mixed $fiokgyogyszertarakSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUzletekVendeglatohelyekGyogyszertarakQuery The current query, for fluid interface

\*/

public function filterByFiokgyogyszertarakSzama($fiokgyogyszertarakSzama = null, $comparison = null)

{

if (is\_array($fiokgyogyszertarakSzama)) {

$useMinMax = false;

if (isset($fiokgyogyszertarakSzama['min'])) {

$this->addUsingAlias(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_FIOKGYOGYSZERTARAK\_SZAMA, $fiokgyogyszertarakSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($fiokgyogyszertarakSzama['max'])) {

$this->addUsingAlias(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_FIOKGYOGYSZERTARAK\_SZAMA, $fiokgyogyszertarakSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_FIOKGYOGYSZERTARAK\_SZAMA, $fiokgyogyszertarakSzama, $comparison);

}

/\*\*

\* Filter the query on the gyogyszertarak\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByGyogyszertarakSzama(1234); // WHERE gyogyszertarak\_szama = 1234

\* $query->filterByGyogyszertarakSzama(array(12, 34)); // WHERE gyogyszertarak\_szama IN (12, 34)

\* $query->filterByGyogyszertarakSzama(array('min' => 12)); // WHERE gyogyszertarak\_szama > 12

\* </code>

\*

\* @param mixed $gyogyszertarakSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgUzletekVendeglatohelyekGyogyszertarakQuery The current query, for fluid interface

\*/

public function filterByGyogyszertarakSzama($gyogyszertarakSzama = null, $comparison = null)

{

if (is\_array($gyogyszertarakSzama)) {

$useMinMax = false;

if (isset($gyogyszertarakSzama['min'])) {

$this->addUsingAlias(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_GYOGYSZERTARAK\_SZAMA, $gyogyszertarakSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($gyogyszertarakSzama['max'])) {

$this->addUsingAlias(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_GYOGYSZERTARAK\_SZAMA, $gyogyszertarakSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_GYOGYSZERTARAK\_SZAMA, $gyogyszertarakSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgUzletekVendeglatohelyekGyogyszertarak $stgUzletekVendeglatohelyekGyogyszertarak Object to remove from the list of results

\*

\* @return $this|ChildStgUzletekVendeglatohelyekGyogyszertarakQuery The current query, for fluid interface

\*/

public function prune($stgUzletekVendeglatohelyekGyogyszertarak = null)

{

if ($stgUzletekVendeglatohelyekGyogyszertarak) {

throw new LogicException('StgUzletekVendeglatohelyekGyogyszertarak object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_uzletek\_vendeglatohelyek\_gyogyszertarak table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgUzletekVendeglatohelyekGyogyszertarakTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgUzletekVendeglatohelyekGyogyszertarakTableMap::clearInstancePool();

StgUzletekVendeglatohelyekGyogyszertarakTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgUzletekVendeglatohelyekGyogyszertarakTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgUzletekVendeglatohelyekGyogyszertarakTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgUzletekVendeglatohelyekGyogyszertarakTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgUzletekVendeglatohelyekGyogyszertarakTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgUzletekVendeglatohelyekGyogyszertarakQuery

#### StgVandorlasiEgyenlegAranya.php

<?php

namespace Base;

use \StgVandorlasiEgyenlegAranyaQuery as ChildStgVandorlasiEgyenlegAranyaQuery;

use \Exception;

use \PDO;

use Map\StgVandorlasiEgyenlegAranyaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\ActiveRecord\ActiveRecordInterface;

use Propel\Runtime\Collection\Collection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\BadMethodCallException;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Parser\AbstractParser;

abstract class StgVandorlasiEgyenlegAranya implements ActiveRecordInterface

{

/\*\*

\* TableMap class name

\*/

const TABLE\_MAP = '\\Map\\StgVandorlasiEgyenlegAranyaTableMap';

/\*\*

\* attribute to determine if this object has previously been saved.

\* @var boolean

\*/

protected $new = true;

/\*\*

\* attribute to determine whether this object has been deleted.

\* @var boolean

\*/

protected $deleted = false;

/\*\*

\* The columns that have been modified in current object.

\* Tracking modified columns allows us to only update modified columns.

\* @var array

\*/

protected $modifiedColumns = array();

/\*\*

\* The (virtual) columns that are added at runtime

\* The formatters can add supplementary columns based on a resultset

\* @var array

\*/

protected $virtualColumns = array();

/\*\*

\* The value for the telepules\_nev field.

\* @var string

\*/

protected $telepules\_nev;

/\*\*

\* The value for the telepules\_kshkod field.

\* @var string

\*/

protected $telepules\_kshkod;

/\*\*

\* The value for the odavandorlasok\_szama field.

\* @var int

\*/

protected $odavandorlasok\_szama;

/\*\*

\* The value for the elvandorlasok\_szama field.

\* @var int

\*/

protected $elvandorlasok\_szama;

/\*\*

\* The value for the elveszuletesek\_szama field.

\* @var int

\*/

protected $elveszuletesek\_szama;

/\*\*

\* The value for the halalozasok\_szama field.

\* @var int

\*/

protected $halalozasok\_szama;

/\*\*

\* Flag to prevent endless save loop, if this object is referenced

\* by another object which falls in this transaction.

\*

\* @var boolean

\*/

protected $alreadyInSave = false;

/\*\*

\* Initializes internal state of Base\StgVandorlasiEgyenlegAranya object.

\*/

public function \_\_construct()

{

}

/\*\*

\* Returns whether the object has been modified.

\*

\* @return boolean True if the object has been modified.

\*/

public function isModified()

{

return !!$this->modifiedColumns;

}

/\*\*

\* Has specified column been modified?

\*

\* @param string $col column fully qualified name (TableMap::TYPE\_COLNAME), e.g. Book::AUTHOR\_ID

\* @return boolean True if $col has been modified.

\*/

public function isColumnModified($col)

{

return $this->modifiedColumns && isset($this->modifiedColumns[$col]);

}

/\*\*

\* Get the columns that have been modified in this object.

\* @return array A unique list of the modified column names for this object.

\*/

public function getModifiedColumns()

{

return $this->modifiedColumns ? array\_keys($this->modifiedColumns) : [];

}

/\*\*

\* Returns whether the object has ever been saved. This will

\* be false, if the object was retrieved from storage or was created

\* and then saved.

\*

\* @return boolean true, if the object has never been persisted.

\*/

public function isNew()

{

return $this->new;

}

/\*\*

\* Setter for the isNew attribute. This method will be called

\* by Propel-generated children and objects.

\*

\* @param boolean $b the state of the object.

\*/

public function setNew($b)

{

$this->new = (boolean) $b;

}

/\*\*

\* Whether this object has been deleted.

\* @return boolean The deleted state of this object.

\*/

public function isDeleted()

{

return $this->deleted;

}

/\*\*

\* Specify whether this object has been deleted.

\* @param boolean $b The deleted state of this object.

\* @return void

\*/

public function setDeleted($b)

{

$this->deleted = (boolean) $b;

}

/\*\*

\* Sets the modified state for the object to be false.

\* @param string $col If supplied, only the specified column is reset.

\* @return void

\*/

public function resetModified($col = null)

{

if (null !== $col) {

if (isset($this->modifiedColumns[$col])) {

unset($this->modifiedColumns[$col]);

}

} else {

$this->modifiedColumns = array();

}

}

/\*\*

\* Compares this with another <code>StgVandorlasiEgyenlegAranya</code> instance. If

\* <code>obj</code> is an instance of <code>StgVandorlasiEgyenlegAranya</code>, delegates to

\* <code>equals(StgVandorlasiEgyenlegAranya)</code>. Otherwise, returns <code>false</code>.

\*

\* @param mixed $obj The object to compare to.

\* @return boolean Whether equal to the object specified.

\*/

public function equals($obj)

{

if (!$obj instanceof static) {

return false;

}

if ($this === $obj) {

return true;

}

if (null === $this->getPrimaryKey() || null === $obj->getPrimaryKey()) {

return false;

}

return $this->getPrimaryKey() === $obj->getPrimaryKey();

}

/\*\*

\* Get the associative array of the virtual columns in this object

\*

\* @return array

\*/

public function getVirtualColumns()

{

return $this->virtualColumns;

}

/\*\*

\* Checks the existence of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return boolean

\*/

public function hasVirtualColumn($name)

{

return array\_key\_exists($name, $this->virtualColumns);

}

/\*\*

\* Get the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @return mixed

\*

\* @throws PropelException

\*/

public function getVirtualColumn($name)

{

if (!$this->hasVirtualColumn($name)) {

throw new PropelException(sprintf('Cannot get value of inexistent virtual column %s.', $name));

}

return $this->virtualColumns[$name];

}

/\*\*

\* Set the value of a virtual column in this object

\*

\* @param string $name The virtual column name

\* @param mixed $value The value to give to the virtual column

\*

\* @return $this|StgVandorlasiEgyenlegAranya The current object, for fluid interface

\*/

public function setVirtualColumn($name, $value)

{

$this->virtualColumns[$name] = $value;

return $this;

}

/\*\*

\* Logs a message using Propel::log().

\*

\* @param string $msg

\* @param int $priority One of the Propel::LOG\_\* logging levels

\* @return boolean

\*/

protected function log($msg, $priority = Propel::LOG\_INFO)

{

return Propel::log(get\_class($this) . ': ' . $msg, $priority);

}

/\*\*

\* Export the current object properties to a string, using a given parser format

\* <code>

\* $book = BookQuery::create()->findPk(9012);

\* echo $book->exportTo('JSON');

\* => {"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance, or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy load(ed) columns. Defaults to TRUE.

\* @return string The exported data

\*/

public function exportTo($parser, $includeLazyLoadColumns = true)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

return $parser->fromArray($this->toArray(TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns, array(), true));

}

/\*\*

\* Clean up internal collections prior to serializing

\* Avoids recursive loops that turn into segmentation faults when serializing

\*/

public function \_\_sleep()

{

$this->clearAllReferences();

return array\_keys(get\_object\_vars($this));

}

/\*\*

\* Get the [telepules\_nev] column value.

\*

\* @return string

\*/

public function getTelepulesNev()

{

return $this->telepules\_nev;

}

/\*\*

\* Get the [telepules\_kshkod] column value.

\*

\* @return string

\*/

public function getTelepulesKshkod()

{

return $this->telepules\_kshkod;

}

/\*\*

\* Get the [odavandorlasok\_szama] column value.

\*

\* @return int

\*/

public function getOdavandorlasokSzama()

{

return $this->odavandorlasok\_szama;

}

/\*\*

\* Get the [elvandorlasok\_szama] column value.

\*

\* @return int

\*/

public function getElvandorlasokSzama()

{

return $this->elvandorlasok\_szama;

}

/\*\*

\* Get the [elveszuletesek\_szama] column value.

\*

\* @return int

\*/

public function getElveszuletesekSzama()

{

return $this->elveszuletesek\_szama;

}

/\*\*

\* Get the [halalozasok\_szama] column value.

\*

\* @return int

\*/

public function getHalalozasokSzama()

{

return $this->halalozasok\_szama;

}

/\*\*

\* Indicates whether the columns in this object are only set to default values.

\*

\* This method can be used in conjunction with isModified() to indicate whether an object is both

\* modified \_and\_ has some values set which are non-default.

\*

\* @return boolean Whether the columns in this object are only been set with default values.

\*/

public function hasOnlyDefaultValues()

{

// otherwise, everything was equal, so return TRUE

return true;

} // hasOnlyDefaultValues()

/\*\*

\* Hydrates (populates) the object variables with values from the database resultset.

\*

\* An offset (0-based "start column") is specified so that objects can be hydrated

\* with a subset of the columns in the resultset rows. This is needed, for example,

\* for results of JOIN queries where the resultset row includes columns from two or

\* more tables.

\*

\* @param array $row The row returned by DataFetcher->fetch().

\* @param int $startcol 0-based offset column which indicates which restultset column to start with.

\* @param boolean $rehydrate Whether this object is being re-hydrated from the database.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @return int next starting column

\* @throws PropelException - Any caught Exception will be rewrapped as a PropelException.

\*/

public function hydrate($row, $startcol = 0, $rehydrate = false, $indexType = TableMap::TYPE\_NUM)

{

try {

$col = $row[TableMap::TYPE\_NUM == $indexType ? 0 + $startcol : StgVandorlasiEgyenlegAranyaTableMap::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_nev = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 1 + $startcol : StgVandorlasiEgyenlegAranyaTableMap::translateFieldName('TelepulesKshkod', TableMap::TYPE\_PHPNAME, $indexType)];

$this->telepules\_kshkod = (null !== $col) ? (string) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 2 + $startcol : StgVandorlasiEgyenlegAranyaTableMap::translateFieldName('OdavandorlasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->odavandorlasok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 3 + $startcol : StgVandorlasiEgyenlegAranyaTableMap::translateFieldName('ElvandorlasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->elvandorlasok\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 4 + $startcol : StgVandorlasiEgyenlegAranyaTableMap::translateFieldName('ElveszuletesekSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->elveszuletesek\_szama = (null !== $col) ? (int) $col : null;

$col = $row[TableMap::TYPE\_NUM == $indexType ? 5 + $startcol : StgVandorlasiEgyenlegAranyaTableMap::translateFieldName('HalalozasokSzama', TableMap::TYPE\_PHPNAME, $indexType)];

$this->halalozasok\_szama = (null !== $col) ? (int) $col : null;

$this->resetModified();

$this->setNew(false);

if ($rehydrate) {

$this->ensureConsistency();

}

return $startcol + 6; // 6 = StgVandorlasiEgyenlegAranyaTableMap::NUM\_HYDRATE\_COLUMNS.

} catch (Exception $e) {

throw new PropelException(sprintf('Error populating %s object', '\\StgVandorlasiEgyenlegAranya'), 0, $e);

}

}

/\*\*

\* Checks and repairs the internal consistency of the object.

\*

\* This method is executed after an already-instantiated object is re-hydrated

\* from the database. It exists to check any foreign keys to make sure that

\* the objects related to the current object are correct based on foreign key.

\*

\* You can override this method in the stub class, but you should always invoke

\* the base method from the overridden method (i.e. parent::ensureConsistency()),

\* in case your model changes.

\*

\* @throws PropelException

\*/

public function ensureConsistency()

{

} // ensureConsistency

/\*\*

\* Set the value of [telepules\_nev] column.

\*

\* @param string $v new value

\* @return $this|\StgVandorlasiEgyenlegAranya The current object (for fluent API support)

\*/

public function setTelepulesNev($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_nev !== $v) {

$this->telepules\_nev = $v;

$this->modifiedColumns[StgVandorlasiEgyenlegAranyaTableMap::COL\_TELEPULES\_NEV] = true;

}

return $this;

} // setTelepulesNev()

/\*\*

\* Set the value of [telepules\_kshkod] column.

\*

\* @param string $v new value

\* @return $this|\StgVandorlasiEgyenlegAranya The current object (for fluent API support)

\*/

public function setTelepulesKshkod($v)

{

if ($v !== null) {

$v = (string) $v;

}

if ($this->telepules\_kshkod !== $v) {

$this->telepules\_kshkod = $v;

$this->modifiedColumns[StgVandorlasiEgyenlegAranyaTableMap::COL\_TELEPULES\_KSHKOD] = true;

}

return $this;

} // setTelepulesKshkod()

/\*\*

\* Set the value of [odavandorlasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgVandorlasiEgyenlegAranya The current object (for fluent API support)

\*/

public function setOdavandorlasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->odavandorlasok\_szama !== $v) {

$this->odavandorlasok\_szama = $v;

$this->modifiedColumns[StgVandorlasiEgyenlegAranyaTableMap::COL\_ODAVANDORLASOK\_SZAMA] = true;

}

return $this;

} // setOdavandorlasokSzama()

/\*\*

\* Set the value of [elvandorlasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgVandorlasiEgyenlegAranya The current object (for fluent API support)

\*/

public function setElvandorlasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->elvandorlasok\_szama !== $v) {

$this->elvandorlasok\_szama = $v;

$this->modifiedColumns[StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVANDORLASOK\_SZAMA] = true;

}

return $this;

} // setElvandorlasokSzama()

/\*\*

\* Set the value of [elveszuletesek\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgVandorlasiEgyenlegAranya The current object (for fluent API support)

\*/

public function setElveszuletesekSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->elveszuletesek\_szama !== $v) {

$this->elveszuletesek\_szama = $v;

$this->modifiedColumns[StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVESZULETESEK\_SZAMA] = true;

}

return $this;

} // setElveszuletesekSzama()

/\*\*

\* Set the value of [halalozasok\_szama] column.

\*

\* @param int $v new value

\* @return $this|\StgVandorlasiEgyenlegAranya The current object (for fluent API support)

\*/

public function setHalalozasokSzama($v)

{

if ($v !== null) {

$v = (int) $v;

}

if ($this->halalozasok\_szama !== $v) {

$this->halalozasok\_szama = $v;

$this->modifiedColumns[StgVandorlasiEgyenlegAranyaTableMap::COL\_HALALOZASOK\_SZAMA] = true;

}

return $this;

} // setHalalozasokSzama()

/\*\*

\* Reloads this object from datastore based on primary key and (optionally) resets all associated objects.

\*

\* This will only work if the object has been saved and has a valid primary key set.

\*

\* @param boolean $deep (optional) Whether to also de-associated any related objects.

\* @param ConnectionInterface $con (optional) The ConnectionInterface connection to use.

\* @return void

\* @throws PropelException - if this object is deleted, unsaved or doesn't have pk match in db

\*/

public function reload($deep = false, ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("Cannot reload a deleted object.");

}

if ($this->isNew()) {

throw new PropelException("Cannot reload an unsaved object.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getReadConnection(StgVandorlasiEgyenlegAranyaTableMap::DATABASE\_NAME);

}

// We don't need to alter the object instance pool; we're just modifying this instance

// already in the pool.

$dataFetcher = ChildStgVandorlasiEgyenlegAranyaQuery::create(null, $this->buildPkeyCriteria())->setFormatter(ModelCriteria::FORMAT\_STATEMENT)->find($con);

$row = $dataFetcher->fetch();

$dataFetcher->close();

if (!$row) {

throw new PropelException('Cannot find matching row in the database to reload object values.');

}

$this->hydrate($row, 0, true, $dataFetcher->getIndexType()); // rehydrate

if ($deep) { // also de-associate any related objects?

} // if (deep)

}

/\*\*

\* Removes this object from datastore and sets delete attribute.

\*

\* @param ConnectionInterface $con

\* @return void

\* @throws PropelException

\* @see StgVandorlasiEgyenlegAranya::setDeleted()

\* @see StgVandorlasiEgyenlegAranya::isDeleted()

\*/

public function delete(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("This object has already been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgVandorlasiEgyenlegAranyaTableMap::DATABASE\_NAME);

}

$con->transaction(function () use ($con) {

$deleteQuery = ChildStgVandorlasiEgyenlegAranyaQuery::create()

->filterByPrimaryKey($this->getPrimaryKey());

$ret = $this->preDelete($con);

if ($ret) {

$deleteQuery->delete($con);

$this->postDelete($con);

$this->setDeleted(true);

}

});

}

/\*\*

\* Persists this object to the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All modified related objects will also be persisted in the doSave()

\* method. This method wraps all precipitate database operations in a

\* single transaction.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see doSave()

\*/

public function save(ConnectionInterface $con = null)

{

if ($this->isDeleted()) {

throw new PropelException("You cannot save an object that has been deleted.");

}

if ($con === null) {

$con = Propel::getServiceContainer()->getWriteConnection(StgVandorlasiEgyenlegAranyaTableMap::DATABASE\_NAME);

}

return $con->transaction(function () use ($con) {

$isInsert = $this->isNew();

$ret = $this->preSave($con);

if ($isInsert) {

$ret = $ret && $this->preInsert($con);

} else {

$ret = $ret && $this->preUpdate($con);

}

if ($ret) {

$affectedRows = $this->doSave($con);

if ($isInsert) {

$this->postInsert($con);

} else {

$this->postUpdate($con);

}

$this->postSave($con);

StgVandorlasiEgyenlegAranyaTableMap::addInstanceToPool($this);

} else {

$affectedRows = 0;

}

return $affectedRows;

});

}

/\*\*

\* Performs the work of inserting or updating the row in the database.

\*

\* If the object is new, it inserts it; otherwise an update is performed.

\* All related objects are also updated in this method.

\*

\* @param ConnectionInterface $con

\* @return int The number of rows affected by this insert/update and any referring fk objects' save() operations.

\* @throws PropelException

\* @see save()

\*/

protected function doSave(ConnectionInterface $con)

{

$affectedRows = 0; // initialize var to track total num of affected rows

if (!$this->alreadyInSave) {

$this->alreadyInSave = true;

if ($this->isNew() || $this->isModified()) {

// persist changes

if ($this->isNew()) {

$this->doInsert($con);

} else {

$this->doUpdate($con);

}

$affectedRows += 1;

$this->resetModified();

}

$this->alreadyInSave = false;

}

return $affectedRows;

} // doSave()

/\*\*

\* Insert the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @throws PropelException

\* @see doSave()

\*/

protected function doInsert(ConnectionInterface $con)

{

$modifiedColumns = array();

$index = 0;

// check the columns in natural order for more readable SQL queries

if ($this->isColumnModified(StgVandorlasiEgyenlegAranyaTableMap::COL\_TELEPULES\_NEV)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_NEV';

}

if ($this->isColumnModified(StgVandorlasiEgyenlegAranyaTableMap::COL\_TELEPULES\_KSHKOD)) {

$modifiedColumns[':p' . $index++] = 'TELEPULES\_KSHKOD';

}

if ($this->isColumnModified(StgVandorlasiEgyenlegAranyaTableMap::COL\_ODAVANDORLASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'ODAVANDORLASOK\_SZAMA';

}

if ($this->isColumnModified(StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVANDORLASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'ELVANDORLASOK\_SZAMA';

}

if ($this->isColumnModified(StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVESZULETESEK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'ELVESZULETESEK\_SZAMA';

}

if ($this->isColumnModified(StgVandorlasiEgyenlegAranyaTableMap::COL\_HALALOZASOK\_SZAMA)) {

$modifiedColumns[':p' . $index++] = 'HALALOZASOK\_SZAMA';

}

$sql = sprintf(

'INSERT INTO stg\_vandorlasi\_egyenleg\_aranya (%s) VALUES (%s)',

implode(', ', $modifiedColumns),

implode(', ', array\_keys($modifiedColumns))

);

try {

$stmt = $con->prepare($sql);

foreach ($modifiedColumns as $identifier => $columnName) {

switch ($columnName) {

case 'TELEPULES\_NEV':

$stmt->bindValue($identifier, $this->telepules\_nev, PDO::PARAM\_STR);

break;

case 'TELEPULES\_KSHKOD':

$stmt->bindValue($identifier, $this->telepules\_kshkod, PDO::PARAM\_STR);

break;

case 'ODAVANDORLASOK\_SZAMA':

$stmt->bindValue($identifier, $this->odavandorlasok\_szama, PDO::PARAM\_INT);

break;

case 'ELVANDORLASOK\_SZAMA':

$stmt->bindValue($identifier, $this->elvandorlasok\_szama, PDO::PARAM\_INT);

break;

case 'ELVESZULETESEK\_SZAMA':

$stmt->bindValue($identifier, $this->elveszuletesek\_szama, PDO::PARAM\_INT);

break;

case 'HALALOZASOK\_SZAMA':

$stmt->bindValue($identifier, $this->halalozasok\_szama, PDO::PARAM\_INT);

break;

}

}

$stmt->execute();

} catch (Exception $e) {

Propel::log($e->getMessage(), Propel::LOG\_ERR);

throw new PropelException(sprintf('Unable to execute INSERT statement [%s]', $sql), 0, $e);

}

$this->setNew(false);

}

/\*\*

\* Update the row in the database.

\*

\* @param ConnectionInterface $con

\*

\* @return Integer Number of updated rows

\* @see doSave()

\*/

protected function doUpdate(ConnectionInterface $con)

{

$selectCriteria = $this->buildPkeyCriteria();

$valuesCriteria = $this->buildCriteria();

return $selectCriteria->doUpdate($valuesCriteria, $con);

}

/\*\*

\* Retrieves a field from the object by name passed in as a string.

\*

\* @param string $name name

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return mixed Value of field.

\*/

public function getByName($name, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgVandorlasiEgyenlegAranyaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

$field = $this->getByPosition($pos);

return $field;

}

/\*\*

\* Retrieves a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @return mixed Value of field at $pos

\*/

public function getByPosition($pos)

{

switch ($pos) {

case 0:

return $this->getTelepulesNev();

break;

case 1:

return $this->getTelepulesKshkod();

break;

case 2:

return $this->getOdavandorlasokSzama();

break;

case 3:

return $this->getElvandorlasokSzama();

break;

case 4:

return $this->getElveszuletesekSzama();

break;

case 5:

return $this->getHalalozasokSzama();

break;

default:

return null;

break;

} // switch()

}

/\*\*

\* Exports the object as an array.

\*

\* You can specify the key type of the array by passing one of the class

\* type constants.

\*

\* @param string $keyType (optional) One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @param boolean $includeLazyLoadColumns (optional) Whether to include lazy loaded columns. Defaults to TRUE.

\* @param array $alreadyDumpedObjects List of objects to skip to avoid recursion

\*

\* @return array an associative array containing the field names (as keys) and field values

\*/

public function toArray($keyType = TableMap::TYPE\_PHPNAME, $includeLazyLoadColumns = true, $alreadyDumpedObjects = array())

{

if (isset($alreadyDumpedObjects['StgVandorlasiEgyenlegAranya'][$this->getPrimaryKey()])) {

return '\*RECURSION\*';

}

$alreadyDumpedObjects['StgVandorlasiEgyenlegAranya'][$this->getPrimaryKey()] = true;

$keys = StgVandorlasiEgyenlegAranyaTableMap::getFieldNames($keyType);

$result = array(

$keys[0] => $this->getTelepulesNev(),

$keys[1] => $this->getTelepulesKshkod(),

$keys[2] => $this->getOdavandorlasokSzama(),

$keys[3] => $this->getElvandorlasokSzama(),

$keys[4] => $this->getElveszuletesekSzama(),

$keys[5] => $this->getHalalozasokSzama(),

);

$virtualColumns = $this->virtualColumns;

foreach ($virtualColumns as $key => $virtualColumn) {

$result[$key] = $virtualColumn;

}

return $result;

}

/\*\*

\* Sets a field from the object by name passed in as a string.

\*

\* @param string $name

\* @param mixed $value field value

\* @param string $type The type of fieldname the $name is of:

\* one of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* Defaults to TableMap::TYPE\_PHPNAME.

\* @return $this|\StgVandorlasiEgyenlegAranya

\*/

public function setByName($name, $value, $type = TableMap::TYPE\_PHPNAME)

{

$pos = StgVandorlasiEgyenlegAranyaTableMap::translateFieldName($name, $type, TableMap::TYPE\_NUM);

return $this->setByPosition($pos, $value);

}

/\*\*

\* Sets a field from the object by Position as specified in the xml schema.

\* Zero-based.

\*

\* @param int $pos position in xml schema

\* @param mixed $value field value

\* @return $this|\StgVandorlasiEgyenlegAranya

\*/

public function setByPosition($pos, $value)

{

switch ($pos) {

case 0:

$this->setTelepulesNev($value);

break;

case 1:

$this->setTelepulesKshkod($value);

break;

case 2:

$this->setOdavandorlasokSzama($value);

break;

case 3:

$this->setElvandorlasokSzama($value);

break;

case 4:

$this->setElveszuletesekSzama($value);

break;

case 5:

$this->setHalalozasokSzama($value);

break;

} // switch()

return $this;

}

/\*\*

\* Populates the object using an array.

\*

\* This is particularly useful when populating an object from one of the

\* request arrays (e.g. $\_POST). This method goes through the column

\* names, checking to see whether a matching key exists in populated

\* array. If so the setByName() method is called for that column.

\*

\* You can specify the key type of the array by additionally passing one

\* of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME,

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\* The default key type is the column's TableMap::TYPE\_PHPNAME.

\*

\* @param array $arr An array to populate the object from.

\* @param string $keyType The type of keys the array uses.

\* @return void

\*/

public function fromArray($arr, $keyType = TableMap::TYPE\_PHPNAME)

{

$keys = StgVandorlasiEgyenlegAranyaTableMap::getFieldNames($keyType);

if (array\_key\_exists($keys[0], $arr)) {

$this->setTelepulesNev($arr[$keys[0]]);

}

if (array\_key\_exists($keys[1], $arr)) {

$this->setTelepulesKshkod($arr[$keys[1]]);

}

if (array\_key\_exists($keys[2], $arr)) {

$this->setOdavandorlasokSzama($arr[$keys[2]]);

}

if (array\_key\_exists($keys[3], $arr)) {

$this->setElvandorlasokSzama($arr[$keys[3]]);

}

if (array\_key\_exists($keys[4], $arr)) {

$this->setElveszuletesekSzama($arr[$keys[4]]);

}

if (array\_key\_exists($keys[5], $arr)) {

$this->setHalalozasokSzama($arr[$keys[5]]);

}

}

/\*\*

\* Populate the current object from a string, using a given parser format

\* <code>

\* $book = new Book();

\* $book->importFrom('JSON', '{"Id":9012,"Title":"Don Juan","ISBN":"0140422161","Price":12.99,"PublisherId":1234,"AuthorId":5678}');

\* </code>

\*

\* @param mixed $parser A AbstractParser instance,

\* or a format name ('XML', 'YAML', 'JSON', 'CSV')

\* @param string $data The source data to import from

\*

\* @return $this|\StgVandorlasiEgyenlegAranya The current object, for fluid interface

\*/

public function importFrom($parser, $data)

{

if (!$parser instanceof AbstractParser) {

$parser = AbstractParser::getParser($parser);

}

$this->fromArray($parser->toArray($data), TableMap::TYPE\_PHPNAME);

return $this;

}

/\*\*

\* Build a Criteria object containing the values of all modified columns in this object.

\*

\* @return Criteria The Criteria object containing all modified values.

\*/

public function buildCriteria()

{

$criteria = new Criteria(StgVandorlasiEgyenlegAranyaTableMap::DATABASE\_NAME);

if ($this->isColumnModified(StgVandorlasiEgyenlegAranyaTableMap::COL\_TELEPULES\_NEV)) {

$criteria->add(StgVandorlasiEgyenlegAranyaTableMap::COL\_TELEPULES\_NEV, $this->telepules\_nev);

}

if ($this->isColumnModified(StgVandorlasiEgyenlegAranyaTableMap::COL\_TELEPULES\_KSHKOD)) {

$criteria->add(StgVandorlasiEgyenlegAranyaTableMap::COL\_TELEPULES\_KSHKOD, $this->telepules\_kshkod);

}

if ($this->isColumnModified(StgVandorlasiEgyenlegAranyaTableMap::COL\_ODAVANDORLASOK\_SZAMA)) {

$criteria->add(StgVandorlasiEgyenlegAranyaTableMap::COL\_ODAVANDORLASOK\_SZAMA, $this->odavandorlasok\_szama);

}

if ($this->isColumnModified(StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVANDORLASOK\_SZAMA)) {

$criteria->add(StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVANDORLASOK\_SZAMA, $this->elvandorlasok\_szama);

}

if ($this->isColumnModified(StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVESZULETESEK\_SZAMA)) {

$criteria->add(StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVESZULETESEK\_SZAMA, $this->elveszuletesek\_szama);

}

if ($this->isColumnModified(StgVandorlasiEgyenlegAranyaTableMap::COL\_HALALOZASOK\_SZAMA)) {

$criteria->add(StgVandorlasiEgyenlegAranyaTableMap::COL\_HALALOZASOK\_SZAMA, $this->halalozasok\_szama);

}

return $criteria;

}

/\*\*

\* Builds a Criteria object containing the primary key for this object.

\*

\* Unlike buildCriteria() this method includes the primary key values regardless

\* of whether or not they have been modified.

\*

\* @throws LogicException if no primary key is defined

\*

\* @return Criteria The Criteria object containing value(s) for primary key(s).

\*/

public function buildPkeyCriteria()

{

throw new LogicException('The StgVandorlasiEgyenlegAranya object has no primary key');

return $criteria;

}

/\*\*

\* If the primary key is not null, return the hashcode of the

\* primary key. Otherwise, return the hash code of the object.

\*

\* @return int Hashcode

\*/

public function hashCode()

{

$validPk = false;

$validPrimaryKeyFKs = 0;

$primaryKeyFKs = [];

if ($validPk) {

return crc32(json\_encode($this->getPrimaryKey(), JSON\_UNESCAPED\_UNICODE));

} elseif ($validPrimaryKeyFKs) {

return crc32(json\_encode($primaryKeyFKs, JSON\_UNESCAPED\_UNICODE));

}

return spl\_object\_hash($this);

}

/\*\*

\* Returns NULL since this table doesn't have a primary key.

\* This method exists only for BC and is deprecated!

\* @return null

\*/

public function getPrimaryKey()

{

return null;

}

/\*\*

\* Dummy primary key setter.

\*

\* This function only exists to preserve backwards compatibility. It is no longer

\* needed or required by the Persistent interface. It will be removed in next BC-breaking

\* release of Propel.

\*

\* @deprecated

\*/

public function setPrimaryKey($pk)

{

// do nothing, because this object doesn't have any primary keys

}

/\*\*

\* Returns true if the primary key for this object is null.

\* @return boolean

\*/

public function isPrimaryKeyNull()

{

return ;

}

/\*\*

\* Sets contents of passed object to values from current object.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param object $copyObj An object of \StgVandorlasiEgyenlegAranya (or compatible) type.

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @param boolean $makeNew Whether to reset autoincrement PKs and make the object new.

\* @throws PropelException

\*/

public function copyInto($copyObj, $deepCopy = false, $makeNew = true)

{

$copyObj->setTelepulesNev($this->getTelepulesNev());

$copyObj->setTelepulesKshkod($this->getTelepulesKshkod());

$copyObj->setOdavandorlasokSzama($this->getOdavandorlasokSzama());

$copyObj->setElvandorlasokSzama($this->getElvandorlasokSzama());

$copyObj->setElveszuletesekSzama($this->getElveszuletesekSzama());

$copyObj->setHalalozasokSzama($this->getHalalozasokSzama());

if ($makeNew) {

$copyObj->setNew(true);

}

}

/\*\*

\* Makes a copy of this object that will be inserted as a new row in table when saved.

\* It creates a new object filling in the simple attributes, but skipping any primary

\* keys that are defined for the table.

\*

\* If desired, this method can also make copies of all associated (fkey referrers)

\* objects.

\*

\* @param boolean $deepCopy Whether to also copy all rows that refer (by fkey) to the current row.

\* @return \StgVandorlasiEgyenlegAranya Clone of current object.

\* @throws PropelException

\*/

public function copy($deepCopy = false)

{

// we use get\_class(), because this might be a subclass

$clazz = get\_class($this);

$copyObj = new $clazz();

$this->copyInto($copyObj, $deepCopy);

return $copyObj;

}

/\*\*

\* Clears the current object, sets all attributes to their default values and removes

\* outgoing references as well as back-references (from other objects to this one. Results probably in a database

\* change of those foreign objects when you call `save` there).

\*/

public function clear()

{

$this->telepules\_nev = null;

$this->telepules\_kshkod = null;

$this->odavandorlasok\_szama = null;

$this->elvandorlasok\_szama = null;

$this->elveszuletesek\_szama = null;

$this->halalozasok\_szama = null;

$this->alreadyInSave = false;

$this->clearAllReferences();

$this->resetModified();

$this->setNew(true);

$this->setDeleted(false);

}

/\*\*

\* Resets all references and back-references to other model objects or collections of model objects.

\*

\* This method is used to reset all php object references (not the actual reference in the database).

\* Necessary for object serialisation.

\*

\* @param boolean $deep Whether to also clear the references on all referrer objects.

\*/

public function clearAllReferences($deep = false)

{

if ($deep) {

} // if ($deep)

}

/\*\*

\* Return the string representation of this object

\*

\* @return string

\*/

public function \_\_toString()

{

return (string) $this->exportTo(StgVandorlasiEgyenlegAranyaTableMap::DEFAULT\_STRING\_FORMAT);

}

/\*\*

\* Code to be run before persisting the object

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preSave(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after persisting the object

\* @param ConnectionInterface $con

\*/

public function postSave(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before inserting to database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preInsert(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after inserting to database

\* @param ConnectionInterface $con

\*/

public function postInsert(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before updating the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preUpdate(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after updating the object in database

\* @param ConnectionInterface $con

\*/

public function postUpdate(ConnectionInterface $con = null)

{

}

/\*\*

\* Code to be run before deleting the object in database

\* @param ConnectionInterface $con

\* @return boolean

\*/

public function preDelete(ConnectionInterface $con = null)

{

return true;

}

/\*\*

\* Code to be run after deleting the object in database

\* @param ConnectionInterface $con

\*/

public function postDelete(ConnectionInterface $con = null)

{

}

/\*\*

\* Derived method to catches calls to undefined methods.

\*

\* Provides magic import/export method support (fromXML()/toXML(), fromYAML()/toYAML(), etc.).

\* Allows to define default \_\_call() behavior if you overwrite \_\_call()

\*

\* @param string $name

\* @param mixed $params

\*

\* @return array|string

\*/

public function \_\_call($name, $params)

{

if (0 === strpos($name, 'get')) {

$virtualColumn = substr($name, 3);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

$virtualColumn = lcfirst($virtualColumn);

if ($this->hasVirtualColumn($virtualColumn)) {

return $this->getVirtualColumn($virtualColumn);

}

}

if (0 === strpos($name, 'from')) {

$format = substr($name, 4);

return $this->importFrom($format, reset($params));

}

if (0 === strpos($name, 'to')) {

$format = substr($name, 2);

$includeLazyLoadColumns = isset($params[0]) ? $params[0] : true;

return $this->exportTo($format, $includeLazyLoadColumns);

}

throw new BadMethodCallException(sprintf('Call to undefined method: %s.', $name));

}

}

#### StgVandorlasiEgyenlegAranyaQuery.php

<?php

namespace Base;

use \StgVandorlasiEgyenlegAranya as ChildStgVandorlasiEgyenlegAranya;

use \StgVandorlasiEgyenlegAranyaQuery as ChildStgVandorlasiEgyenlegAranyaQuery;

use \Exception;

use Map\StgVandorlasiEgyenlegAranyaTableMap;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\ModelCriteria;

use Propel\Runtime\Collection\ObjectCollection;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

/\*\*

\* Base class that represents a query for the 'stg\_vandorlasi\_egyenleg\_aranya' table.

\*

\*

\*

\* @method ChildStgVandorlasiEgyenlegAranyaQuery orderByTelepulesNev($order = Criteria::ASC) Order by the telepules\_nev column

\* @method ChildStgVandorlasiEgyenlegAranyaQuery orderByTelepulesKshkod($order = Criteria::ASC) Order by the telepules\_KSHKOD column

\* @method ChildStgVandorlasiEgyenlegAranyaQuery orderByOdavandorlasokSzama($order = Criteria::ASC) Order by the odavandorlasok\_szama column

\* @method ChildStgVandorlasiEgyenlegAranyaQuery orderByElvandorlasokSzama($order = Criteria::ASC) Order by the elvandorlasok\_szama column

\* @method ChildStgVandorlasiEgyenlegAranyaQuery orderByElveszuletesekSzama($order = Criteria::ASC) Order by the elveszuletesek\_szama column

\* @method ChildStgVandorlasiEgyenlegAranyaQuery orderByHalalozasokSzama($order = Criteria::ASC) Order by the halalozasok\_szama column

\*

\* @method ChildStgVandorlasiEgyenlegAranyaQuery groupByTelepulesNev() Group by the telepules\_nev column

\* @method ChildStgVandorlasiEgyenlegAranyaQuery groupByTelepulesKshkod() Group by the telepules\_KSHKOD column

\* @method ChildStgVandorlasiEgyenlegAranyaQuery groupByOdavandorlasokSzama() Group by the odavandorlasok\_szama column

\* @method ChildStgVandorlasiEgyenlegAranyaQuery groupByElvandorlasokSzama() Group by the elvandorlasok\_szama column

\* @method ChildStgVandorlasiEgyenlegAranyaQuery groupByElveszuletesekSzama() Group by the elveszuletesek\_szama column

\* @method ChildStgVandorlasiEgyenlegAranyaQuery groupByHalalozasokSzama() Group by the halalozasok\_szama column

\*

\* @method ChildStgVandorlasiEgyenlegAranyaQuery leftJoin($relation) Adds a LEFT JOIN clause to the query

\* @method ChildStgVandorlasiEgyenlegAranyaQuery rightJoin($relation) Adds a RIGHT JOIN clause to the query

\* @method ChildStgVandorlasiEgyenlegAranyaQuery innerJoin($relation) Adds a INNER JOIN clause to the query

\*

\* @method ChildStgVandorlasiEgyenlegAranya findOne(ConnectionInterface $con = null) Return the first ChildStgVandorlasiEgyenlegAranya matching the query

\* @method ChildStgVandorlasiEgyenlegAranya findOneOrCreate(ConnectionInterface $con = null) Return the first ChildStgVandorlasiEgyenlegAranya matching the query, or a new ChildStgVandorlasiEgyenlegAranya object populated from the query conditions when no match is found

\*

\* @method ChildStgVandorlasiEgyenlegAranya findOneByTelepulesNev(string $telepules\_nev) Return the first ChildStgVandorlasiEgyenlegAranya filtered by the telepules\_nev column

\* @method ChildStgVandorlasiEgyenlegAranya findOneByTelepulesKshkod(string $telepules\_KSHKOD) Return the first ChildStgVandorlasiEgyenlegAranya filtered by the telepules\_KSHKOD column

\* @method ChildStgVandorlasiEgyenlegAranya findOneByOdavandorlasokSzama(int $odavandorlasok\_szama) Return the first ChildStgVandorlasiEgyenlegAranya filtered by the odavandorlasok\_szama column

\* @method ChildStgVandorlasiEgyenlegAranya findOneByElvandorlasokSzama(int $elvandorlasok\_szama) Return the first ChildStgVandorlasiEgyenlegAranya filtered by the elvandorlasok\_szama column

\* @method ChildStgVandorlasiEgyenlegAranya findOneByElveszuletesekSzama(int $elveszuletesek\_szama) Return the first ChildStgVandorlasiEgyenlegAranya filtered by the elveszuletesek\_szama column

\* @method ChildStgVandorlasiEgyenlegAranya findOneByHalalozasokSzama(int $halalozasok\_szama) Return the first ChildStgVandorlasiEgyenlegAranya filtered by the halalozasok\_szama column

\*

\* @method ChildStgVandorlasiEgyenlegAranya[]|ObjectCollection find(ConnectionInterface $con = null) Return ChildStgVandorlasiEgyenlegAranya objects based on current ModelCriteria

\* @method ChildStgVandorlasiEgyenlegAranya[]|ObjectCollection findByTelepulesNev(string $telepules\_nev) Return ChildStgVandorlasiEgyenlegAranya objects filtered by the telepules\_nev column

\* @method ChildStgVandorlasiEgyenlegAranya[]|ObjectCollection findByTelepulesKshkod(string $telepules\_KSHKOD) Return ChildStgVandorlasiEgyenlegAranya objects filtered by the telepules\_KSHKOD column

\* @method ChildStgVandorlasiEgyenlegAranya[]|ObjectCollection findByOdavandorlasokSzama(int $odavandorlasok\_szama) Return ChildStgVandorlasiEgyenlegAranya objects filtered by the odavandorlasok\_szama column

\* @method ChildStgVandorlasiEgyenlegAranya[]|ObjectCollection findByElvandorlasokSzama(int $elvandorlasok\_szama) Return ChildStgVandorlasiEgyenlegAranya objects filtered by the elvandorlasok\_szama column

\* @method ChildStgVandorlasiEgyenlegAranya[]|ObjectCollection findByElveszuletesekSzama(int $elveszuletesek\_szama) Return ChildStgVandorlasiEgyenlegAranya objects filtered by the elveszuletesek\_szama column

\* @method ChildStgVandorlasiEgyenlegAranya[]|ObjectCollection findByHalalozasokSzama(int $halalozasok\_szama) Return ChildStgVandorlasiEgyenlegAranya objects filtered by the halalozasok\_szama column

\* @method ChildStgVandorlasiEgyenlegAranya[]|\Propel\Runtime\Util\PropelModelPager paginate($page = 1, $maxPerPage = 10, ConnectionInterface $con = null) Issue a SELECT query based on the current ModelCriteria and uses a page and a maximum number of results per page to compute an offset and a limit

\*

\*/

abstract class StgVandorlasiEgyenlegAranyaQuery extends ModelCriteria

{

/\*\*

\* Initializes internal state of \Base\StgVandorlasiEgyenlegAranyaQuery object.

\*

\* @param string $dbName The database name

\* @param string $modelName The phpName of a model, e.g. 'Book'

\* @param string $modelAlias The alias for the model in this query, e.g. 'b'

\*/

public function \_\_construct($dbName = 'etraffic\_v1', $modelName = '\\StgVandorlasiEgyenlegAranya', $modelAlias = null)

{

parent::\_\_construct($dbName, $modelName, $modelAlias);

}

/\*\*

\* Returns a new ChildStgVandorlasiEgyenlegAranyaQuery object.

\*

\* @param string $modelAlias The alias of a model in the query

\* @param Criteria $criteria Optional Criteria to build the query from

\*

\* @return ChildStgVandorlasiEgyenlegAranyaQuery

\*/

public static function create($modelAlias = null, Criteria $criteria = null)

{

if ($criteria instanceof ChildStgVandorlasiEgyenlegAranyaQuery) {

return $criteria;

}

$query = new ChildStgVandorlasiEgyenlegAranyaQuery();

if (null !== $modelAlias) {

$query->setModelAlias($modelAlias);

}

if ($criteria instanceof Criteria) {

$query->mergeWith($criteria);

}

return $query;

}

/\*\*

\* Find object by primary key.

\* Propel uses the instance pool to skip the database if the object exists.

\* Go fast if the query is untouched.

\*

\* <code>

\* $obj = $c->findPk(12, $con);

\* </code>

\*

\* @param mixed $key Primary key to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ChildStgVandorlasiEgyenlegAranya|array|mixed the result, formatted by the current formatter

\*/

public function findPk($key, ConnectionInterface $con = null)

{

throw new LogicException('The StgVandorlasiEgyenlegAranya object has no primary key');

}

/\*\*

\* Find objects by primary key

\* <code>

\* $objs = $c->findPks(array(array(12, 56), array(832, 123), array(123, 456)), $con);

\* </code>

\* @param array $keys Primary keys to use for the query

\* @param ConnectionInterface $con an optional connection object

\*

\* @return ObjectCollection|array|mixed the list of results, formatted by the current formatter

\*/

public function findPks($keys, ConnectionInterface $con = null)

{

throw new LogicException('The StgVandorlasiEgyenlegAranya object has no primary key');

}

/\*\*

\* Filter the query by primary key

\*

\* @param mixed $key Primary key to use for the query

\*

\* @return $this|ChildStgVandorlasiEgyenlegAranyaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKey($key)

{

throw new LogicException('The StgVandorlasiEgyenlegAranya object has no primary key');

}

/\*\*

\* Filter the query by a list of primary keys

\*

\* @param array $keys The list of primary key to use for the query

\*

\* @return $this|ChildStgVandorlasiEgyenlegAranyaQuery The current query, for fluid interface

\*/

public function filterByPrimaryKeys($keys)

{

throw new LogicException('The StgVandorlasiEgyenlegAranya object has no primary key');

}

/\*\*

\* Filter the query on the telepules\_nev column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesNev('fooValue'); // WHERE telepules\_nev = 'fooValue'

\* $query->filterByTelepulesNev('%fooValue%'); // WHERE telepules\_nev LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesNev The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgVandorlasiEgyenlegAranyaQuery The current query, for fluid interface

\*/

public function filterByTelepulesNev($telepulesNev = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesNev)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesNev)) {

$telepulesNev = str\_replace('\*', '%', $telepulesNev);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgVandorlasiEgyenlegAranyaTableMap::COL\_TELEPULES\_NEV, $telepulesNev, $comparison);

}

/\*\*

\* Filter the query on the telepules\_KSHKOD column

\*

\* Example usage:

\* <code>

\* $query->filterByTelepulesKshkod('fooValue'); // WHERE telepules\_KSHKOD = 'fooValue'

\* $query->filterByTelepulesKshkod('%fooValue%'); // WHERE telepules\_KSHKOD LIKE '%fooValue%'

\* </code>

\*

\* @param string $telepulesKshkod The value to use as filter.

\* Accepts wildcards (\* and % trigger a LIKE)

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgVandorlasiEgyenlegAranyaQuery The current query, for fluid interface

\*/

public function filterByTelepulesKshkod($telepulesKshkod = null, $comparison = null)

{

if (null === $comparison) {

if (is\_array($telepulesKshkod)) {

$comparison = Criteria::IN;

} elseif (preg\_match('/[\%\\*]/', $telepulesKshkod)) {

$telepulesKshkod = str\_replace('\*', '%', $telepulesKshkod);

$comparison = Criteria::LIKE;

}

}

return $this->addUsingAlias(StgVandorlasiEgyenlegAranyaTableMap::COL\_TELEPULES\_KSHKOD, $telepulesKshkod, $comparison);

}

/\*\*

\* Filter the query on the odavandorlasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByOdavandorlasokSzama(1234); // WHERE odavandorlasok\_szama = 1234

\* $query->filterByOdavandorlasokSzama(array(12, 34)); // WHERE odavandorlasok\_szama IN (12, 34)

\* $query->filterByOdavandorlasokSzama(array('min' => 12)); // WHERE odavandorlasok\_szama > 12

\* </code>

\*

\* @param mixed $odavandorlasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgVandorlasiEgyenlegAranyaQuery The current query, for fluid interface

\*/

public function filterByOdavandorlasokSzama($odavandorlasokSzama = null, $comparison = null)

{

if (is\_array($odavandorlasokSzama)) {

$useMinMax = false;

if (isset($odavandorlasokSzama['min'])) {

$this->addUsingAlias(StgVandorlasiEgyenlegAranyaTableMap::COL\_ODAVANDORLASOK\_SZAMA, $odavandorlasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($odavandorlasokSzama['max'])) {

$this->addUsingAlias(StgVandorlasiEgyenlegAranyaTableMap::COL\_ODAVANDORLASOK\_SZAMA, $odavandorlasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgVandorlasiEgyenlegAranyaTableMap::COL\_ODAVANDORLASOK\_SZAMA, $odavandorlasokSzama, $comparison);

}

/\*\*

\* Filter the query on the elvandorlasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByElvandorlasokSzama(1234); // WHERE elvandorlasok\_szama = 1234

\* $query->filterByElvandorlasokSzama(array(12, 34)); // WHERE elvandorlasok\_szama IN (12, 34)

\* $query->filterByElvandorlasokSzama(array('min' => 12)); // WHERE elvandorlasok\_szama > 12

\* </code>

\*

\* @param mixed $elvandorlasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgVandorlasiEgyenlegAranyaQuery The current query, for fluid interface

\*/

public function filterByElvandorlasokSzama($elvandorlasokSzama = null, $comparison = null)

{

if (is\_array($elvandorlasokSzama)) {

$useMinMax = false;

if (isset($elvandorlasokSzama['min'])) {

$this->addUsingAlias(StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVANDORLASOK\_SZAMA, $elvandorlasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($elvandorlasokSzama['max'])) {

$this->addUsingAlias(StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVANDORLASOK\_SZAMA, $elvandorlasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVANDORLASOK\_SZAMA, $elvandorlasokSzama, $comparison);

}

/\*\*

\* Filter the query on the elveszuletesek\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByElveszuletesekSzama(1234); // WHERE elveszuletesek\_szama = 1234

\* $query->filterByElveszuletesekSzama(array(12, 34)); // WHERE elveszuletesek\_szama IN (12, 34)

\* $query->filterByElveszuletesekSzama(array('min' => 12)); // WHERE elveszuletesek\_szama > 12

\* </code>

\*

\* @param mixed $elveszuletesekSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgVandorlasiEgyenlegAranyaQuery The current query, for fluid interface

\*/

public function filterByElveszuletesekSzama($elveszuletesekSzama = null, $comparison = null)

{

if (is\_array($elveszuletesekSzama)) {

$useMinMax = false;

if (isset($elveszuletesekSzama['min'])) {

$this->addUsingAlias(StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVESZULETESEK\_SZAMA, $elveszuletesekSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($elveszuletesekSzama['max'])) {

$this->addUsingAlias(StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVESZULETESEK\_SZAMA, $elveszuletesekSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVESZULETESEK\_SZAMA, $elveszuletesekSzama, $comparison);

}

/\*\*

\* Filter the query on the halalozasok\_szama column

\*

\* Example usage:

\* <code>

\* $query->filterByHalalozasokSzama(1234); // WHERE halalozasok\_szama = 1234

\* $query->filterByHalalozasokSzama(array(12, 34)); // WHERE halalozasok\_szama IN (12, 34)

\* $query->filterByHalalozasokSzama(array('min' => 12)); // WHERE halalozasok\_szama > 12

\* </code>

\*

\* @param mixed $halalozasokSzama The value to use as filter.

\* Use scalar values for equality.

\* Use array values for in\_array() equivalent.

\* Use associative array('min' => $minValue, 'max' => $maxValue) for intervals.

\* @param string $comparison Operator to use for the column comparison, defaults to Criteria::EQUAL

\*

\* @return $this|ChildStgVandorlasiEgyenlegAranyaQuery The current query, for fluid interface

\*/

public function filterByHalalozasokSzama($halalozasokSzama = null, $comparison = null)

{

if (is\_array($halalozasokSzama)) {

$useMinMax = false;

if (isset($halalozasokSzama['min'])) {

$this->addUsingAlias(StgVandorlasiEgyenlegAranyaTableMap::COL\_HALALOZASOK\_SZAMA, $halalozasokSzama['min'], Criteria::GREATER\_EQUAL);

$useMinMax = true;

}

if (isset($halalozasokSzama['max'])) {

$this->addUsingAlias(StgVandorlasiEgyenlegAranyaTableMap::COL\_HALALOZASOK\_SZAMA, $halalozasokSzama['max'], Criteria::LESS\_EQUAL);

$useMinMax = true;

}

if ($useMinMax) {

return $this;

}

if (null === $comparison) {

$comparison = Criteria::IN;

}

}

return $this->addUsingAlias(StgVandorlasiEgyenlegAranyaTableMap::COL\_HALALOZASOK\_SZAMA, $halalozasokSzama, $comparison);

}

/\*\*

\* Exclude object from result

\*

\* @param ChildStgVandorlasiEgyenlegAranya $stgVandorlasiEgyenlegAranya Object to remove from the list of results

\*

\* @return $this|ChildStgVandorlasiEgyenlegAranyaQuery The current query, for fluid interface

\*/

public function prune($stgVandorlasiEgyenlegAranya = null)

{

if ($stgVandorlasiEgyenlegAranya) {

throw new LogicException('StgVandorlasiEgyenlegAranya object has no primary key');

}

return $this;

}

/\*\*

\* Deletes all rows from the stg\_vandorlasi\_egyenleg\_aranya table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public function doDeleteAll(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgVandorlasiEgyenlegAranyaTableMap::DATABASE\_NAME);

}

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con) {

$affectedRows = 0; // initialize var to track total num of affected rows

$affectedRows += parent::doDeleteAll($con);

// Because this db requires some delete cascade/set null emulation, we have to

// clear the cached instance \*after\* the emulation has happened (since

// instances get re-added by the select statement contained therein).

StgVandorlasiEgyenlegAranyaTableMap::clearInstancePool();

StgVandorlasiEgyenlegAranyaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

/\*\*

\* Performs a DELETE on the database based on the current ModelCriteria

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public function delete(ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgVandorlasiEgyenlegAranyaTableMap::DATABASE\_NAME);

}

$criteria = $this;

// Set the correct dbName

$criteria->setDbName(StgVandorlasiEgyenlegAranyaTableMap::DATABASE\_NAME);

// use transaction because $criteria could contain info

// for more than one table or we could emulating ON DELETE CASCADE, etc.

return $con->transaction(function () use ($con, $criteria) {

$affectedRows = 0; // initialize var to track total num of affected rows

StgVandorlasiEgyenlegAranyaTableMap::removeInstanceFromPool($criteria);

$affectedRows += ModelCriteria::delete($con);

StgVandorlasiEgyenlegAranyaTableMap::clearRelatedInstancePool();

return $affectedRows;

});

}

} // StgVandorlasiEgyenlegAranyaQuery

## Table maping

### Stg10FonelNagyobbSzervezetekTableMap.php

<?php

namespace Map;

use \Stg10FonelNagyobbSzervezetek;

use \Stg10FonelNagyobbSzervezetekQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_10\_fonel\_nagyobb\_szervezetek' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class Stg10FonelNagyobbSzervezetekTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.Stg10FonelNagyobbSzervezetekTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_10\_fonel\_nagyobb\_szervezetek';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\Stg10FonelNagyobbSzervezetek';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'Stg10FonelNagyobbSzervezetek';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 4;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 4;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_10\_fonel\_nagyobb\_szervezetek.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_10\_fonel\_nagyobb\_szervezetek.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the MUKODO\_VALLALATOK\_SZAMA field

\*/

const COL\_MUKODO\_VALLALATOK\_SZAMA = 'stg\_10\_fonel\_nagyobb\_szervezetek.MUKODO\_VALLALATOK\_SZAMA';

/\*\*

\* the column name for the VALLALATOK\_ATLAGOS\_LETSZAMA field

\*/

const COL\_VALLALATOK\_ATLAGOS\_LETSZAMA = 'stg\_10\_fonel\_nagyobb\_szervezetek.VALLALATOK\_ATLAGOS\_LETSZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'MukodoVallalatokSzama', 'VallalatokAtlagosLetszama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'mukodoVallalatokSzama', 'vallalatokAtlagosLetszama', ),

self::TYPE\_COLNAME => array(Stg10FonelNagyobbSzervezetekTableMap::COL\_TELEPULES\_NEV, Stg10FonelNagyobbSzervezetekTableMap::COL\_TELEPULES\_KSHKOD, Stg10FonelNagyobbSzervezetekTableMap::COL\_MUKODO\_VALLALATOK\_SZAMA, Stg10FonelNagyobbSzervezetekTableMap::COL\_VALLALATOK\_ATLAGOS\_LETSZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_MUKODO\_VALLALATOK\_SZAMA', 'COL\_VALLALATOK\_ATLAGOS\_LETSZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'mukodo\_vallalatok\_szama', 'vallalatok\_atlagos\_letszama', ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'MukodoVallalatokSzama' => 2, 'VallalatokAtlagosLetszama' => 3, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'mukodoVallalatokSzama' => 2, 'vallalatokAtlagosLetszama' => 3, ),

self::TYPE\_COLNAME => array(Stg10FonelNagyobbSzervezetekTableMap::COL\_TELEPULES\_NEV => 0, Stg10FonelNagyobbSzervezetekTableMap::COL\_TELEPULES\_KSHKOD => 1, Stg10FonelNagyobbSzervezetekTableMap::COL\_MUKODO\_VALLALATOK\_SZAMA => 2, Stg10FonelNagyobbSzervezetekTableMap::COL\_VALLALATOK\_ATLAGOS\_LETSZAMA => 3, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_MUKODO\_VALLALATOK\_SZAMA' => 2, 'COL\_VALLALATOK\_ATLAGOS\_LETSZAMA' => 3, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'mukodo\_vallalatok\_szama' => 2, 'vallalatok\_atlagos\_letszama' => 3, ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_10\_fonel\_nagyobb\_szervezetek');

$this->setPhpName('Stg10FonelNagyobbSzervezetek');

$this->setClassName('\\Stg10FonelNagyobbSzervezetek');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('MUKODO\_VALLALATOK\_SZAMA', 'MukodoVallalatokSzama', 'INTEGER', false, null, null);

$this->addColumn('VALLALATOK\_ATLAGOS\_LETSZAMA', 'VallalatokAtlagosLetszama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? Stg10FonelNagyobbSzervezetekTableMap::CLASS\_DEFAULT : Stg10FonelNagyobbSzervezetekTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (Stg10FonelNagyobbSzervezetek object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = Stg10FonelNagyobbSzervezetekTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = Stg10FonelNagyobbSzervezetekTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + Stg10FonelNagyobbSzervezetekTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = Stg10FonelNagyobbSzervezetekTableMap::OM\_CLASS;

/\*\* @var Stg10FonelNagyobbSzervezetek $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

Stg10FonelNagyobbSzervezetekTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = Stg10FonelNagyobbSzervezetekTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = Stg10FonelNagyobbSzervezetekTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var Stg10FonelNagyobbSzervezetek $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

Stg10FonelNagyobbSzervezetekTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(Stg10FonelNagyobbSzervezetekTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(Stg10FonelNagyobbSzervezetekTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(Stg10FonelNagyobbSzervezetekTableMap::COL\_MUKODO\_VALLALATOK\_SZAMA);

$criteria->addSelectColumn(Stg10FonelNagyobbSzervezetekTableMap::COL\_VALLALATOK\_ATLAGOS\_LETSZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.MUKODO\_VALLALATOK\_SZAMA');

$criteria->addSelectColumn($alias . '.VALLALATOK\_ATLAGOS\_LETSZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(Stg10FonelNagyobbSzervezetekTableMap::DATABASE\_NAME)->getTable(Stg10FonelNagyobbSzervezetekTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(Stg10FonelNagyobbSzervezetekTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(Stg10FonelNagyobbSzervezetekTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new Stg10FonelNagyobbSzervezetekTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a Stg10FonelNagyobbSzervezetek or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or Stg10FonelNagyobbSzervezetek object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(Stg10FonelNagyobbSzervezetekTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \Stg10FonelNagyobbSzervezetek) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The Stg10FonelNagyobbSzervezetek object has no primary key');

}

$query = Stg10FonelNagyobbSzervezetekQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

Stg10FonelNagyobbSzervezetekTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

Stg10FonelNagyobbSzervezetekTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_10\_fonel\_nagyobb\_szervezetek table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return Stg10FonelNagyobbSzervezetekQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a Stg10FonelNagyobbSzervezetek or Criteria object.

\*

\* @param mixed $criteria Criteria or Stg10FonelNagyobbSzervezetek object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(Stg10FonelNagyobbSzervezetekTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from Stg10FonelNagyobbSzervezetek object

}

// Set the correct dbName

$query = Stg10FonelNagyobbSzervezetekQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // Stg10FonelNagyobbSzervezetekTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

Stg10FonelNagyobbSzervezetekTableMap::buildTableMap();

### StgBelfoldiOsszesJovedelemTableMap.php

<?php

namespace Map;

use \StgBelfoldiOsszesJovedelem;

use \StgBelfoldiOsszesJovedelemQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_belfoldi\_osszes\_jovedelem' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgBelfoldiOsszesJovedelemTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgBelfoldiOsszesJovedelemTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_belfoldi\_osszes\_jovedelem';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgBelfoldiOsszesJovedelem';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgBelfoldiOsszesJovedelem';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 3;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 3;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_belfoldi\_osszes\_jovedelem.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_belfoldi\_osszes\_jovedelem.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the JOVEDELEM field

\*/

const COL\_JOVEDELEM = 'stg\_belfoldi\_osszes\_jovedelem.JOVEDELEM';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'Jovedelem', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'jovedelem', ),

self::TYPE\_COLNAME => array(StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_NEV, StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_KSHKOD, StgBelfoldiOsszesJovedelemTableMap::COL\_JOVEDELEM, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_JOVEDELEM', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'jovedelem', ),

self::TYPE\_NUM => array(0, 1, 2, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'Jovedelem' => 2, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'jovedelem' => 2, ),

self::TYPE\_COLNAME => array(StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_NEV => 0, StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_KSHKOD => 1, StgBelfoldiOsszesJovedelemTableMap::COL\_JOVEDELEM => 2, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_JOVEDELEM' => 2, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'jovedelem' => 2, ),

self::TYPE\_NUM => array(0, 1, 2, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_belfoldi\_osszes\_jovedelem');

$this->setPhpName('StgBelfoldiOsszesJovedelem');

$this->setClassName('\\StgBelfoldiOsszesJovedelem');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addPrimaryKey('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('JOVEDELEM', 'Jovedelem', 'DOUBLE', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

// If the PK cannot be derived from the row, return NULL.

if ($row[TableMap::TYPE\_NUM == $indexType ? 0 + $offset : static::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)] === null) {

return null;

}

return (string) $row[TableMap::TYPE\_NUM == $indexType ? 0 + $offset : static::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return (string) $row[

$indexType == TableMap::TYPE\_NUM

? 0 + $offset

: self::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)

];

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgBelfoldiOsszesJovedelemTableMap::CLASS\_DEFAULT : StgBelfoldiOsszesJovedelemTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgBelfoldiOsszesJovedelem object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgBelfoldiOsszesJovedelemTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgBelfoldiOsszesJovedelemTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgBelfoldiOsszesJovedelemTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgBelfoldiOsszesJovedelemTableMap::OM\_CLASS;

/\*\* @var StgBelfoldiOsszesJovedelem $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgBelfoldiOsszesJovedelemTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgBelfoldiOsszesJovedelemTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgBelfoldiOsszesJovedelemTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgBelfoldiOsszesJovedelem $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgBelfoldiOsszesJovedelemTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgBelfoldiOsszesJovedelemTableMap::COL\_JOVEDELEM);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.JOVEDELEM');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgBelfoldiOsszesJovedelemTableMap::DATABASE\_NAME)->getTable(StgBelfoldiOsszesJovedelemTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgBelfoldiOsszesJovedelemTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgBelfoldiOsszesJovedelemTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgBelfoldiOsszesJovedelemTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgBelfoldiOsszesJovedelem or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgBelfoldiOsszesJovedelem object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgBelfoldiOsszesJovedelemTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgBelfoldiOsszesJovedelem) { // it's a model object

// create criteria based on pk values

$criteria = $values->buildPkeyCriteria();

} else { // it's a primary key, or an array of pks

$criteria = new Criteria(StgBelfoldiOsszesJovedelemTableMap::DATABASE\_NAME);

$criteria->add(StgBelfoldiOsszesJovedelemTableMap::COL\_TELEPULES\_NEV, (array) $values, Criteria::IN);

}

$query = StgBelfoldiOsszesJovedelemQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgBelfoldiOsszesJovedelemTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgBelfoldiOsszesJovedelemTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_belfoldi\_osszes\_jovedelem table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgBelfoldiOsszesJovedelemQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgBelfoldiOsszesJovedelem or Criteria object.

\*

\* @param mixed $criteria Criteria or StgBelfoldiOsszesJovedelem object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgBelfoldiOsszesJovedelemTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgBelfoldiOsszesJovedelem object

}

// Set the correct dbName

$query = StgBelfoldiOsszesJovedelemQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgBelfoldiOsszesJovedelemTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgBelfoldiOsszesJovedelemTableMap::buildTableMap();

### StgCegautokTableMap.php

<?php

namespace Map;

use \StgCegautok;

use \StgCegautokQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_cegautok' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgCegautokTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgCegautokTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_cegautok';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgCegautok';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgCegautok';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 4;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 4;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_cegautok.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_cegautok.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the SZEMELYGEPKOCSIK\_SZAMA field

\*/

const COL\_SZEMELYGEPKOCSIK\_SZAMA = 'stg\_cegautok.SZEMELYGEPKOCSIK\_SZAMA';

/\*\*

\* the column name for the TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA field

\*/

const COL\_TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA = 'stg\_cegautok.TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'SzemelygepkocsikSzama', 'TermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'szemelygepkocsikSzama', 'termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama', ),

self::TYPE\_COLNAME => array(StgCegautokTableMap::COL\_TELEPULES\_NEV, StgCegautokTableMap::COL\_TELEPULES\_KSHKOD, StgCegautokTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA, StgCegautokTableMap::COL\_TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_SZEMELYGEPKOCSIK\_SZAMA', 'COL\_TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'szemelygepkocsik\_szama', 'termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'SzemelygepkocsikSzama' => 2, 'TermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama' => 3, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'szemelygepkocsikSzama' => 2, 'termeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama' => 3, ),

self::TYPE\_COLNAME => array(StgCegautokTableMap::COL\_TELEPULES\_NEV => 0, StgCegautokTableMap::COL\_TELEPULES\_KSHKOD => 1, StgCegautokTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA => 2, StgCegautokTableMap::COL\_TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA => 3, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_SZEMELYGEPKOCSIK\_SZAMA' => 2, 'COL\_TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA' => 3, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'szemelygepkocsik\_szama' => 2, 'termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama' => 3, ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_cegautok');

$this->setPhpName('StgCegautok');

$this->setClassName('\\StgCegautok');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('SZEMELYGEPKOCSIK\_SZAMA', 'SzemelygepkocsikSzama', 'INTEGER', false, null, null);

$this->addColumn('TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA', 'TermeszetesSzemelyAltalUzemeltetettSzemelygepkocsikSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgCegautokTableMap::CLASS\_DEFAULT : StgCegautokTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgCegautok object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgCegautokTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgCegautokTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgCegautokTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgCegautokTableMap::OM\_CLASS;

/\*\* @var StgCegautok $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgCegautokTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgCegautokTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgCegautokTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgCegautok $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgCegautokTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgCegautokTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgCegautokTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgCegautokTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA);

$criteria->addSelectColumn(StgCegautokTableMap::COL\_TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.SZEMELYGEPKOCSIK\_SZAMA');

$criteria->addSelectColumn($alias . '.TERMESZETES\_SZEMELY\_ALTAL\_UZEMELTETETT\_SZEMELYGEPKOCSIK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgCegautokTableMap::DATABASE\_NAME)->getTable(StgCegautokTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgCegautokTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgCegautokTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgCegautokTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgCegautok or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgCegautok object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgCegautokTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgCegautok) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgCegautok object has no primary key');

}

$query = StgCegautokQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgCegautokTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgCegautokTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_cegautok table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgCegautokQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgCegautok or Criteria object.

\*

\* @param mixed $criteria Criteria or StgCegautok object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgCegautokTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgCegautok object

}

// Set the correct dbName

$query = StgCegautokQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgCegautokTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgCegautokTableMap::buildTableMap();

### StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap.php

<?php

namespace Map;

use \StgEgeszsegugyiSzocialisEllatottakElvandoroltak;

use \StgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgEgeszsegugyiSzocialisEllatottakElvandoroltak';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgEgeszsegugyiSzocialisEllatottakElvandoroltak';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 11;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 11;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA field

\*/

const COL\_HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA = 'stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak.HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA';

/\*\*

\* the column name for the HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA field

\*/

const COL\_HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA = 'stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak.HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA';

/\*\*

\* the column name for the ELBOCSATOTT\_BETEGEK\_SZAMA field

\*/

const COL\_ELBOCSATOTT\_BETEGEK\_SZAMA = 'stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak.ELBOCSATOTT\_BETEGEK\_SZAMA';

/\*\*

\* the column name for the MEGJELENESI\_ESETEK\_SZAMA field

\*/

const COL\_MEGJELENESI\_ESETEK\_SZAMA = 'stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak.MEGJELENESI\_ESETEK\_SZAMA';

/\*\*

\* the column name for the NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA field

\*/

const COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA = 'stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak.NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA';

/\*\*

\* the column name for the NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA field

\*/

const COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA = 'stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak.NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA';

/\*\*

\* the column name for the NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA field

\*/

const COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA = 'stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak.NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA';

/\*\*

\* the column name for the NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA field

\*/

const COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA = 'stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak.NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA';

/\*\*

\* the column name for the ODAVANDORLASOK\_SZAMA field

\*/

const COL\_ODAVANDORLASOK\_SZAMA = 'stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak.ODAVANDORLASOK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'HaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama', 'HaziOrvosiEllatasbanRendelesenMegjelentekSzama', 'ElbocsatottBetegekSzama', 'MegjelenesiEsetekSzama', 'NappaliEllatasbanReszesuloFogyatekosSzemelyekSzama', 'NappaliEllatasbanReszesuloIdoskoruakSzama', 'NappaliEllatasbanReszesuloPszichiatriaiBetegekSzama', 'NappaliEllatasbanReszesuloSzenvedelybetegekSzama', 'OdavandorlasokSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'haziGyermekorvosiEllatasbanRendelesenMegjelentekSzama', 'haziOrvosiEllatasbanRendelesenMegjelentekSzama', 'elbocsatottBetegekSzama', 'megjelenesiEsetekSzama', 'nappaliEllatasbanReszesuloFogyatekosSzemelyekSzama', 'nappaliEllatasbanReszesuloIdoskoruakSzama', 'nappaliEllatasbanReszesuloPszichiatriaiBetegekSzama', 'nappaliEllatasbanReszesuloSzenvedelybetegekSzama', 'odavandorlasokSzama', ),

self::TYPE\_COLNAME => array(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_TELEPULES\_NEV, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_TELEPULES\_KSHKOD, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ELBOCSATOTT\_BETEGEK\_SZAMA, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_MEGJELENESI\_ESETEK\_SZAMA, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ODAVANDORLASOK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA', 'COL\_HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA', 'COL\_ELBOCSATOTT\_BETEGEK\_SZAMA', 'COL\_MEGJELENESI\_ESETEK\_SZAMA', 'COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA', 'COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA', 'COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA', 'COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA', 'COL\_ODAVANDORLASOK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama', 'hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama', 'elbocsatott\_betegek\_szama', 'megjelenesi\_esetek\_szama', 'nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama', 'nappali\_ellatasban\_reszesulo\_idoskoruak\_szama', 'nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama', 'nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama', 'odavandorlasok\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'HaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama' => 2, 'HaziOrvosiEllatasbanRendelesenMegjelentekSzama' => 3, 'ElbocsatottBetegekSzama' => 4, 'MegjelenesiEsetekSzama' => 5, 'NappaliEllatasbanReszesuloFogyatekosSzemelyekSzama' => 6, 'NappaliEllatasbanReszesuloIdoskoruakSzama' => 7, 'NappaliEllatasbanReszesuloPszichiatriaiBetegekSzama' => 8, 'NappaliEllatasbanReszesuloSzenvedelybetegekSzama' => 9, 'OdavandorlasokSzama' => 10, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'haziGyermekorvosiEllatasbanRendelesenMegjelentekSzama' => 2, 'haziOrvosiEllatasbanRendelesenMegjelentekSzama' => 3, 'elbocsatottBetegekSzama' => 4, 'megjelenesiEsetekSzama' => 5, 'nappaliEllatasbanReszesuloFogyatekosSzemelyekSzama' => 6, 'nappaliEllatasbanReszesuloIdoskoruakSzama' => 7, 'nappaliEllatasbanReszesuloPszichiatriaiBetegekSzama' => 8, 'nappaliEllatasbanReszesuloSzenvedelybetegekSzama' => 9, 'odavandorlasokSzama' => 10, ),

self::TYPE\_COLNAME => array(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_TELEPULES\_NEV => 0, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_TELEPULES\_KSHKOD => 1, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA => 2, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA => 3, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ELBOCSATOTT\_BETEGEK\_SZAMA => 4, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_MEGJELENESI\_ESETEK\_SZAMA => 5, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA => 6, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA => 7, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA => 8, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA => 9, StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ODAVANDORLASOK\_SZAMA => 10, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA' => 2, 'COL\_HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA' => 3, 'COL\_ELBOCSATOTT\_BETEGEK\_SZAMA' => 4, 'COL\_MEGJELENESI\_ESETEK\_SZAMA' => 5, 'COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA' => 6, 'COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA' => 7, 'COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA' => 8, 'COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA' => 9, 'COL\_ODAVANDORLASOK\_SZAMA' => 10, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama' => 2, 'hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama' => 3, 'elbocsatott\_betegek\_szama' => 4, 'megjelenesi\_esetek\_szama' => 5, 'nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama' => 6, 'nappali\_ellatasban\_reszesulo\_idoskoruak\_szama' => 7, 'nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama' => 8, 'nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama' => 9, 'odavandorlasok\_szama' => 10, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak');

$this->setPhpName('StgEgeszsegugyiSzocialisEllatottakElvandoroltak');

$this->setClassName('\\StgEgeszsegugyiSzocialisEllatottakElvandoroltak');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA', 'HaziGyermekorvosiEllatasbanRendelesenMegjelentekSzama', 'INTEGER', false, null, null);

$this->addColumn('HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA', 'HaziOrvosiEllatasbanRendelesenMegjelentekSzama', 'INTEGER', false, null, null);

$this->addColumn('ELBOCSATOTT\_BETEGEK\_SZAMA', 'ElbocsatottBetegekSzama', 'INTEGER', false, null, null);

$this->addColumn('MEGJELENESI\_ESETEK\_SZAMA', 'MegjelenesiEsetekSzama', 'INTEGER', false, null, null);

$this->addColumn('NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA', 'NappaliEllatasbanReszesuloFogyatekosSzemelyekSzama', 'INTEGER', false, null, null);

$this->addColumn('NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA', 'NappaliEllatasbanReszesuloIdoskoruakSzama', 'INTEGER', false, null, null);

$this->addColumn('NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA', 'NappaliEllatasbanReszesuloPszichiatriaiBetegekSzama', 'INTEGER', false, null, null);

$this->addColumn('NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA', 'NappaliEllatasbanReszesuloSzenvedelybetegekSzama', 'INTEGER', false, null, null);

$this->addColumn('ODAVANDORLASOK\_SZAMA', 'OdavandorlasokSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::CLASS\_DEFAULT : StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgEgeszsegugyiSzocialisEllatottakElvandoroltak object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::OM\_CLASS;

/\*\* @var StgEgeszsegugyiSzocialisEllatottakElvandoroltak $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgEgeszsegugyiSzocialisEllatottakElvandoroltak $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA);

$criteria->addSelectColumn(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA);

$criteria->addSelectColumn(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ELBOCSATOTT\_BETEGEK\_SZAMA);

$criteria->addSelectColumn(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_MEGJELENESI\_ESETEK\_SZAMA);

$criteria->addSelectColumn(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA);

$criteria->addSelectColumn(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA);

$criteria->addSelectColumn(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA);

$criteria->addSelectColumn(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA);

$criteria->addSelectColumn(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::COL\_ODAVANDORLASOK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.HAZI\_GYERMEKORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA');

$criteria->addSelectColumn($alias . '.HAZI\_ORVOSI\_ELLATASBAN\_RENDELESEN\_MEGJELENTEK\_SZAMA');

$criteria->addSelectColumn($alias . '.ELBOCSATOTT\_BETEGEK\_SZAMA');

$criteria->addSelectColumn($alias . '.MEGJELENESI\_ESETEK\_SZAMA');

$criteria->addSelectColumn($alias . '.NAPPALI\_ELLATASBAN\_RESZESULO\_FOGYATEKOS\_SZEMELYEK\_SZAMA');

$criteria->addSelectColumn($alias . '.NAPPALI\_ELLATASBAN\_RESZESULO\_IDOSKORUAK\_SZAMA');

$criteria->addSelectColumn($alias . '.NAPPALI\_ELLATASBAN\_RESZESULO\_PSZICHIATRIAI\_BETEGEK\_SZAMA');

$criteria->addSelectColumn($alias . '.NAPPALI\_ELLATASBAN\_RESZESULO\_SZENVEDELYBETEGEK\_SZAMA');

$criteria->addSelectColumn($alias . '.ODAVANDORLASOK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::DATABASE\_NAME)->getTable(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgEgeszsegugyiSzocialisEllatottakElvandoroltak or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgEgeszsegugyiSzocialisEllatottakElvandoroltak object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgEgeszsegugyiSzocialisEllatottakElvandoroltak) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgEgeszsegugyiSzocialisEllatottakElvandoroltak object has no primary key');

}

$query = StgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgEgeszsegugyiSzocialisEllatottakElvandoroltak or Criteria object.

\*

\* @param mixed $criteria Criteria or StgEgeszsegugyiSzocialisEllatottakElvandoroltak object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgEgeszsegugyiSzocialisEllatottakElvandoroltak object

}

// Set the correct dbName

$query = StgEgeszsegugyiSzocialisEllatottakElvandoroltakQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgEgeszsegugyiSzocialisEllatottakElvandoroltakTableMap::buildTableMap();

### StgEgyebIntezmenyiEllatottsagSulyaTableMap.php

<?php

namespace Map;

use \StgEgyebIntezmenyiEllatottsagSulya;

use \StgEgyebIntezmenyiEllatottsagSulyaQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_egyeb\_intezmenyi\_ellatottsag\_sulya' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgEgyebIntezmenyiEllatottsagSulyaTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgEgyebIntezmenyiEllatottsagSulyaTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_egyeb\_intezmenyi\_ellatottsag\_sulya';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgEgyebIntezmenyiEllatottsagSulya';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgEgyebIntezmenyiEllatottsagSulya';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 13;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 13;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_egyeb\_intezmenyi\_ellatottsag\_sulya.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_egyeb\_intezmenyi\_ellatottsag\_sulya.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the BANKFIOK\_LETE field

\*/

const COL\_BANKFIOK\_LETE = 'stg\_egyeb\_intezmenyi\_ellatottsag\_sulya.BANKFIOK\_LETE';

/\*\*

\* the column name for the BENZINKUT\_LETE field

\*/

const COL\_BENZINKUT\_LETE = 'stg\_egyeb\_intezmenyi\_ellatottsag\_sulya.BENZINKUT\_LETE';

/\*\*

\* the column name for the KORJEGYZOSEG\_SZEKHELYE field

\*/

const COL\_KORJEGYZOSEG\_SZEKHELYE = 'stg\_egyeb\_intezmenyi\_ellatottsag\_sulya.KORJEGYZOSEG\_SZEKHELYE';

/\*\*

\* the column name for the POSTAHIVATAL\_LETE field

\*/

const COL\_POSTAHIVATAL\_LETE = 'stg\_egyeb\_intezmenyi\_ellatottsag\_sulya.POSTAHIVATAL\_LETE';

/\*\*

\* the column name for the BIROSAG\_UGYESZSEG\_LETE field

\*/

const COL\_BIROSAG\_UGYESZSEG\_LETE = 'stg\_egyeb\_intezmenyi\_ellatottsag\_sulya.BIROSAG\_UGYESZSEG\_LETE';

/\*\*

\* the column name for the OKMANYIRODA\_LETE field

\*/

const COL\_OKMANYIRODA\_LETE = 'stg\_egyeb\_intezmenyi\_ellatottsag\_sulya.OKMANYIRODA\_LETE';

/\*\*

\* the column name for the MEGYEI\_FOLDHIVATAL\_ILLETVE\_KIRENDELTSEG\_LETE field

\*/

const COL\_MEGYEI\_FOLDHIVATAL\_ILLETVE\_KIRENDELTSEG\_LETE = 'stg\_egyeb\_intezmenyi\_ellatottsag\_sulya.MEGYEI\_FOLDHIVATAL\_ILLETVE\_KIRENDELTSEG\_LETE';

/\*\*

\* the column name for the MUNKAUGYI\_KOZPONT\_ILLETVE\_KIRENDELTSEG\_LETE field

\*/

const COL\_MUNKAUGYI\_KOZPONT\_ILLETVE\_KIRENDELTSEG\_LETE = 'stg\_egyeb\_intezmenyi\_ellatottsag\_sulya.MUNKAUGYI\_KOZPONT\_ILLETVE\_KIRENDELTSEG\_LETE';

/\*\*

\* the column name for the VODAFONE\_UZLET\_LETE field

\*/

const COL\_VODAFONE\_UZLET\_LETE = 'stg\_egyeb\_intezmenyi\_ellatottsag\_sulya.VODAFONE\_UZLET\_LETE';

/\*\*

\* the column name for the T\_MOBILE\_UZLET\_LETE field

\*/

const COL\_T\_MOBILE\_UZLET\_LETE = 'stg\_egyeb\_intezmenyi\_ellatottsag\_sulya.T\_MOBILE\_UZLET\_LETE';

/\*\*

\* the column name for the TELENOR\_UZLET\_LETE field

\*/

const COL\_TELENOR\_UZLET\_LETE = 'stg\_egyeb\_intezmenyi\_ellatottsag\_sulya.TELENOR\_UZLET\_LETE';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'BankfiokLete', 'BenzinkutLete', 'KorjegyzosegSzekhelye', 'PostahivatalLete', 'BirosagUgyeszsegLete', 'OkmanyirodaLete', 'MegyeiFoldhivatalIlletveKirendeltsegLete', 'MunkaugyiKozpontIlletveKirendeltsegLete', 'VodafoneUzletLete', 'TMobileUzletLete', 'TelenorUzletLete', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'bankfiokLete', 'benzinkutLete', 'korjegyzosegSzekhelye', 'postahivatalLete', 'birosagUgyeszsegLete', 'okmanyirodaLete', 'megyeiFoldhivatalIlletveKirendeltsegLete', 'munkaugyiKozpontIlletveKirendeltsegLete', 'vodafoneUzletLete', 'tMobileUzletLete', 'telenorUzletLete', ),

self::TYPE\_COLNAME => array(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELEPULES\_NEV, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELEPULES\_KSHKOD, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BANKFIOK\_LETE, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BENZINKUT\_LETE, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_KORJEGYZOSEG\_SZEKHELYE, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_POSTAHIVATAL\_LETE, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BIROSAG\_UGYESZSEG\_LETE, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_OKMANYIRODA\_LETE, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_MEGYEI\_FOLDHIVATAL\_ILLETVE\_KIRENDELTSEG\_LETE, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_MUNKAUGYI\_KOZPONT\_ILLETVE\_KIRENDELTSEG\_LETE, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_VODAFONE\_UZLET\_LETE, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_T\_MOBILE\_UZLET\_LETE, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELENOR\_UZLET\_LETE, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_BANKFIOK\_LETE', 'COL\_BENZINKUT\_LETE', 'COL\_KORJEGYZOSEG\_SZEKHELYE', 'COL\_POSTAHIVATAL\_LETE', 'COL\_BIROSAG\_UGYESZSEG\_LETE', 'COL\_OKMANYIRODA\_LETE', 'COL\_MEGYEI\_FOLDHIVATAL\_ILLETVE\_KIRENDELTSEG\_LETE', 'COL\_MUNKAUGYI\_KOZPONT\_ILLETVE\_KIRENDELTSEG\_LETE', 'COL\_VODAFONE\_UZLET\_LETE', 'COL\_T\_MOBILE\_UZLET\_LETE', 'COL\_TELENOR\_UZLET\_LETE', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'bankfiok\_lete', 'benzinkut\_lete', 'korjegyzoseg\_szekhelye', 'postahivatal\_lete', 'birosag\_ugyeszseg\_lete', 'okmanyiroda\_lete', 'megyei\_foldhivatal\_illetve\_kirendeltseg\_lete', 'munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete', 'vodafone\_uzlet\_lete', 't\_mobile\_uzlet\_lete', 'telenor\_uzlet\_lete', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'BankfiokLete' => 2, 'BenzinkutLete' => 3, 'KorjegyzosegSzekhelye' => 4, 'PostahivatalLete' => 5, 'BirosagUgyeszsegLete' => 6, 'OkmanyirodaLete' => 7, 'MegyeiFoldhivatalIlletveKirendeltsegLete' => 8, 'MunkaugyiKozpontIlletveKirendeltsegLete' => 9, 'VodafoneUzletLete' => 10, 'TMobileUzletLete' => 11, 'TelenorUzletLete' => 12, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'bankfiokLete' => 2, 'benzinkutLete' => 3, 'korjegyzosegSzekhelye' => 4, 'postahivatalLete' => 5, 'birosagUgyeszsegLete' => 6, 'okmanyirodaLete' => 7, 'megyeiFoldhivatalIlletveKirendeltsegLete' => 8, 'munkaugyiKozpontIlletveKirendeltsegLete' => 9, 'vodafoneUzletLete' => 10, 'tMobileUzletLete' => 11, 'telenorUzletLete' => 12, ),

self::TYPE\_COLNAME => array(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELEPULES\_NEV => 0, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELEPULES\_KSHKOD => 1, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BANKFIOK\_LETE => 2, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BENZINKUT\_LETE => 3, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_KORJEGYZOSEG\_SZEKHELYE => 4, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_POSTAHIVATAL\_LETE => 5, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BIROSAG\_UGYESZSEG\_LETE => 6, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_OKMANYIRODA\_LETE => 7, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_MEGYEI\_FOLDHIVATAL\_ILLETVE\_KIRENDELTSEG\_LETE => 8, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_MUNKAUGYI\_KOZPONT\_ILLETVE\_KIRENDELTSEG\_LETE => 9, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_VODAFONE\_UZLET\_LETE => 10, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_T\_MOBILE\_UZLET\_LETE => 11, StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELENOR\_UZLET\_LETE => 12, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_BANKFIOK\_LETE' => 2, 'COL\_BENZINKUT\_LETE' => 3, 'COL\_KORJEGYZOSEG\_SZEKHELYE' => 4, 'COL\_POSTAHIVATAL\_LETE' => 5, 'COL\_BIROSAG\_UGYESZSEG\_LETE' => 6, 'COL\_OKMANYIRODA\_LETE' => 7, 'COL\_MEGYEI\_FOLDHIVATAL\_ILLETVE\_KIRENDELTSEG\_LETE' => 8, 'COL\_MUNKAUGYI\_KOZPONT\_ILLETVE\_KIRENDELTSEG\_LETE' => 9, 'COL\_VODAFONE\_UZLET\_LETE' => 10, 'COL\_T\_MOBILE\_UZLET\_LETE' => 11, 'COL\_TELENOR\_UZLET\_LETE' => 12, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'bankfiok\_lete' => 2, 'benzinkut\_lete' => 3, 'korjegyzoseg\_szekhelye' => 4, 'postahivatal\_lete' => 5, 'birosag\_ugyeszseg\_lete' => 6, 'okmanyiroda\_lete' => 7, 'megyei\_foldhivatal\_illetve\_kirendeltseg\_lete' => 8, 'munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete' => 9, 'vodafone\_uzlet\_lete' => 10, 't\_mobile\_uzlet\_lete' => 11, 'telenor\_uzlet\_lete' => 12, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_egyeb\_intezmenyi\_ellatottsag\_sulya');

$this->setPhpName('StgEgyebIntezmenyiEllatottsagSulya');

$this->setClassName('\\StgEgyebIntezmenyiEllatottsagSulya');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('BANKFIOK\_LETE', 'BankfiokLete', 'BOOLEAN', false, 1, null);

$this->addColumn('BENZINKUT\_LETE', 'BenzinkutLete', 'BOOLEAN', false, 1, null);

$this->addColumn('KORJEGYZOSEG\_SZEKHELYE', 'KorjegyzosegSzekhelye', 'BOOLEAN', false, 1, null);

$this->addColumn('POSTAHIVATAL\_LETE', 'PostahivatalLete', 'BOOLEAN', false, 1, null);

$this->addColumn('BIROSAG\_UGYESZSEG\_LETE', 'BirosagUgyeszsegLete', 'BOOLEAN', false, 1, null);

$this->addColumn('OKMANYIRODA\_LETE', 'OkmanyirodaLete', 'BOOLEAN', false, 1, null);

$this->addColumn('MEGYEI\_FOLDHIVATAL\_ILLETVE\_KIRENDELTSEG\_LETE', 'MegyeiFoldhivatalIlletveKirendeltsegLete', 'BOOLEAN', false, 1, null);

$this->addColumn('MUNKAUGYI\_KOZPONT\_ILLETVE\_KIRENDELTSEG\_LETE', 'MunkaugyiKozpontIlletveKirendeltsegLete', 'BOOLEAN', false, 1, null);

$this->addColumn('VODAFONE\_UZLET\_LETE', 'VodafoneUzletLete', 'BOOLEAN', false, 1, null);

$this->addColumn('T\_MOBILE\_UZLET\_LETE', 'TMobileUzletLete', 'BOOLEAN', false, 1, null);

$this->addColumn('TELENOR\_UZLET\_LETE', 'TelenorUzletLete', 'BOOLEAN', false, 1, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgEgyebIntezmenyiEllatottsagSulyaTableMap::CLASS\_DEFAULT : StgEgyebIntezmenyiEllatottsagSulyaTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgEgyebIntezmenyiEllatottsagSulya object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgEgyebIntezmenyiEllatottsagSulyaTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgEgyebIntezmenyiEllatottsagSulyaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgEgyebIntezmenyiEllatottsagSulyaTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgEgyebIntezmenyiEllatottsagSulyaTableMap::OM\_CLASS;

/\*\* @var StgEgyebIntezmenyiEllatottsagSulya $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgEgyebIntezmenyiEllatottsagSulyaTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgEgyebIntezmenyiEllatottsagSulyaTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgEgyebIntezmenyiEllatottsagSulyaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgEgyebIntezmenyiEllatottsagSulya $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgEgyebIntezmenyiEllatottsagSulyaTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BANKFIOK\_LETE);

$criteria->addSelectColumn(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BENZINKUT\_LETE);

$criteria->addSelectColumn(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_KORJEGYZOSEG\_SZEKHELYE);

$criteria->addSelectColumn(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_POSTAHIVATAL\_LETE);

$criteria->addSelectColumn(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_BIROSAG\_UGYESZSEG\_LETE);

$criteria->addSelectColumn(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_OKMANYIRODA\_LETE);

$criteria->addSelectColumn(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_MEGYEI\_FOLDHIVATAL\_ILLETVE\_KIRENDELTSEG\_LETE);

$criteria->addSelectColumn(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_MUNKAUGYI\_KOZPONT\_ILLETVE\_KIRENDELTSEG\_LETE);

$criteria->addSelectColumn(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_VODAFONE\_UZLET\_LETE);

$criteria->addSelectColumn(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_T\_MOBILE\_UZLET\_LETE);

$criteria->addSelectColumn(StgEgyebIntezmenyiEllatottsagSulyaTableMap::COL\_TELENOR\_UZLET\_LETE);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.BANKFIOK\_LETE');

$criteria->addSelectColumn($alias . '.BENZINKUT\_LETE');

$criteria->addSelectColumn($alias . '.KORJEGYZOSEG\_SZEKHELYE');

$criteria->addSelectColumn($alias . '.POSTAHIVATAL\_LETE');

$criteria->addSelectColumn($alias . '.BIROSAG\_UGYESZSEG\_LETE');

$criteria->addSelectColumn($alias . '.OKMANYIRODA\_LETE');

$criteria->addSelectColumn($alias . '.MEGYEI\_FOLDHIVATAL\_ILLETVE\_KIRENDELTSEG\_LETE');

$criteria->addSelectColumn($alias . '.MUNKAUGYI\_KOZPONT\_ILLETVE\_KIRENDELTSEG\_LETE');

$criteria->addSelectColumn($alias . '.VODAFONE\_UZLET\_LETE');

$criteria->addSelectColumn($alias . '.T\_MOBILE\_UZLET\_LETE');

$criteria->addSelectColumn($alias . '.TELENOR\_UZLET\_LETE');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgEgyebIntezmenyiEllatottsagSulyaTableMap::DATABASE\_NAME)->getTable(StgEgyebIntezmenyiEllatottsagSulyaTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgEgyebIntezmenyiEllatottsagSulyaTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgEgyebIntezmenyiEllatottsagSulyaTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgEgyebIntezmenyiEllatottsagSulyaTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgEgyebIntezmenyiEllatottsagSulya or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgEgyebIntezmenyiEllatottsagSulya object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgEgyebIntezmenyiEllatottsagSulyaTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgEgyebIntezmenyiEllatottsagSulya) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgEgyebIntezmenyiEllatottsagSulya object has no primary key');

}

$query = StgEgyebIntezmenyiEllatottsagSulyaQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgEgyebIntezmenyiEllatottsagSulyaTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgEgyebIntezmenyiEllatottsagSulyaTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_egyeb\_intezmenyi\_ellatottsag\_sulya table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgEgyebIntezmenyiEllatottsagSulyaQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgEgyebIntezmenyiEllatottsagSulya or Criteria object.

\*

\* @param mixed $criteria Criteria or StgEgyebIntezmenyiEllatottsagSulya object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgEgyebIntezmenyiEllatottsagSulyaTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgEgyebIntezmenyiEllatottsagSulya object

}

// Set the correct dbName

$query = StgEgyebIntezmenyiEllatottsagSulyaQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgEgyebIntezmenyiEllatottsagSulyaTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgEgyebIntezmenyiEllatottsagSulyaTableMap::buildTableMap();

### StgElelmiszeruzletekEsAruhazakTableMap.php

<?php

namespace Map;

use \StgElelmiszeruzletekEsAruhazak;

use \StgElelmiszeruzletekEsAruhazakQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_elelmiszeruzletek\_es\_aruhazak' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgElelmiszeruzletekEsAruhazakTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgElelmiszeruzletekEsAruhazakTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_elelmiszeruzletek\_es\_aruhazak';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgElelmiszeruzletekEsAruhazak';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgElelmiszeruzletekEsAruhazak';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 4;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 4;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_elelmiszeruzletek\_es\_aruhazak.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_elelmiszeruzletek\_es\_aruhazak.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the KISKERESKEDELMI\_UZLETEK\_SZAMA field

\*/

const COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA = 'stg\_elelmiszeruzletek\_es\_aruhazak.KISKERESKEDELMI\_UZLETEK\_SZAMA';

/\*\*

\* the column name for the ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA field

\*/

const COL\_ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA = 'stg\_elelmiszeruzletek\_es\_aruhazak.ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'KiskereskedelmiUzletekSzama', 'ElelmiszerVegyesuzletekEsAruhazakSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'kiskereskedelmiUzletekSzama', 'elelmiszerVegyesuzletekEsAruhazakSzama', ),

self::TYPE\_COLNAME => array(StgElelmiszeruzletekEsAruhazakTableMap::COL\_TELEPULES\_NEV, StgElelmiszeruzletekEsAruhazakTableMap::COL\_TELEPULES\_KSHKOD, StgElelmiszeruzletekEsAruhazakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA, StgElelmiszeruzletekEsAruhazakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA', 'COL\_ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'kiskereskedelmi\_uzletek\_szama', 'elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'KiskereskedelmiUzletekSzama' => 2, 'ElelmiszerVegyesuzletekEsAruhazakSzama' => 3, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'kiskereskedelmiUzletekSzama' => 2, 'elelmiszerVegyesuzletekEsAruhazakSzama' => 3, ),

self::TYPE\_COLNAME => array(StgElelmiszeruzletekEsAruhazakTableMap::COL\_TELEPULES\_NEV => 0, StgElelmiszeruzletekEsAruhazakTableMap::COL\_TELEPULES\_KSHKOD => 1, StgElelmiszeruzletekEsAruhazakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA => 2, StgElelmiszeruzletekEsAruhazakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA => 3, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA' => 2, 'COL\_ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA' => 3, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'kiskereskedelmi\_uzletek\_szama' => 2, 'elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama' => 3, ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_elelmiszeruzletek\_es\_aruhazak');

$this->setPhpName('StgElelmiszeruzletekEsAruhazak');

$this->setClassName('\\StgElelmiszeruzletekEsAruhazak');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('KISKERESKEDELMI\_UZLETEK\_SZAMA', 'KiskereskedelmiUzletekSzama', 'INTEGER', false, null, null);

$this->addColumn('ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA', 'ElelmiszerVegyesuzletekEsAruhazakSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgElelmiszeruzletekEsAruhazakTableMap::CLASS\_DEFAULT : StgElelmiszeruzletekEsAruhazakTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgElelmiszeruzletekEsAruhazak object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgElelmiszeruzletekEsAruhazakTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgElelmiszeruzletekEsAruhazakTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgElelmiszeruzletekEsAruhazakTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgElelmiszeruzletekEsAruhazakTableMap::OM\_CLASS;

/\*\* @var StgElelmiszeruzletekEsAruhazak $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgElelmiszeruzletekEsAruhazakTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgElelmiszeruzletekEsAruhazakTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgElelmiszeruzletekEsAruhazakTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgElelmiszeruzletekEsAruhazak $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgElelmiszeruzletekEsAruhazakTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgElelmiszeruzletekEsAruhazakTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgElelmiszeruzletekEsAruhazakTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgElelmiszeruzletekEsAruhazakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA);

$criteria->addSelectColumn(StgElelmiszeruzletekEsAruhazakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.KISKERESKEDELMI\_UZLETEK\_SZAMA');

$criteria->addSelectColumn($alias . '.ELELMISZER\_VEGYESUZLETEK\_ES\_ARUHAZAK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgElelmiszeruzletekEsAruhazakTableMap::DATABASE\_NAME)->getTable(StgElelmiszeruzletekEsAruhazakTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgElelmiszeruzletekEsAruhazakTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgElelmiszeruzletekEsAruhazakTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgElelmiszeruzletekEsAruhazakTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgElelmiszeruzletekEsAruhazak or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgElelmiszeruzletekEsAruhazak object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgElelmiszeruzletekEsAruhazakTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgElelmiszeruzletekEsAruhazak) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgElelmiszeruzletekEsAruhazak object has no primary key');

}

$query = StgElelmiszeruzletekEsAruhazakQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgElelmiszeruzletekEsAruhazakTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgElelmiszeruzletekEsAruhazakTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_elelmiszeruzletek\_es\_aruhazak table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgElelmiszeruzletekEsAruhazakQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgElelmiszeruzletekEsAruhazak or Criteria object.

\*

\* @param mixed $criteria Criteria or StgElelmiszeruzletekEsAruhazak object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgElelmiszeruzletekEsAruhazakTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgElelmiszeruzletekEsAruhazak object

}

// Set the correct dbName

$query = StgElelmiszeruzletekEsAruhazakQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgElelmiszeruzletekEsAruhazakTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgElelmiszeruzletekEsAruhazakTableMap::buildTableMap();

### StgErintettNepessegSzamaTableMap.php

<?php

namespace Map;

use \StgErintettNepessegSzama;

use \StgErintettNepessegSzamaQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_erintett\_nepesseg\_szama' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgErintettNepessegSzamaTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgErintettNepessegSzamaTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_erintett\_nepesseg\_szama';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgErintettNepessegSzama';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgErintettNepessegSzama';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 15;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 15;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_erintett\_nepesseg\_szama.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_erintett\_nepesseg\_szama.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the REGISZTRALT\_BUNELKOVETOK\_SZAMA field

\*/

const COL\_REGISZTRALT\_BUNELKOVETOK\_SZAMA = 'stg\_erintett\_nepesseg\_szama.REGISZTRALT\_BUNELKOVETOK\_SZAMA';

/\*\*

\* the column name for the HAZTARTASI\_GAZFOGYASZTOK\_SZAMA field

\*/

const COL\_HAZTARTASI\_GAZFOGYASZTOK\_SZAMA = 'stg\_erintett\_nepesseg\_szama.HAZTARTASI\_GAZFOGYASZTOK\_SZAMA';

/\*\*

\* the column name for the HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA field

\*/

const COL\_HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA = 'stg\_erintett\_nepesseg\_szama.HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA';

/\*\*

\* the column name for the ELVANDORLASOK\_SZAMA field

\*/

const COL\_ELVANDORLASOK\_SZAMA = 'stg\_erintett\_nepesseg\_szama.ELVANDORLASOK\_SZAMA';

/\*\*

\* the column name for the ALLANDO\_NEPESSEG\_SZAMA field

\*/

const COL\_ALLANDO\_NEPESSEG\_SZAMA = 'stg\_erintett\_nepesseg\_szama.ALLANDO\_NEPESSEG\_SZAMA';

/\*\*

\* the column name for the VALLALKOZASOK\_SZAMA field

\*/

const COL\_VALLALKOZASOK\_SZAMA = 'stg\_erintett\_nepesseg\_szama.VALLALKOZASOK\_SZAMA';

/\*\*

\* the column name for the NYILVANTARTOTT\_ALLASKERESOK\_SZAMA field

\*/

const COL\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA = 'stg\_erintett\_nepesseg\_szama.NYILVANTARTOTT\_ALLASKERESOK\_SZAMA';

/\*\*

\* the column name for the LAKASALLOMANY field

\*/

const COL\_LAKASALLOMANY = 'stg\_erintett\_nepesseg\_szama.LAKASALLOMANY';

/\*\*

\* the column name for the BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA field

\*/

const COL\_BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA = 'stg\_erintett\_nepesseg\_szama.BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA';

/\*\*

\* the column name for the ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA field

\*/

const COL\_ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA = 'stg\_erintett\_nepesseg\_szama.ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA';

/\*\*

\* the column name for the MOBILELOFIZETESEK\_SZAMA field

\*/

const COL\_MOBILELOFIZETESEK\_SZAMA = 'stg\_erintett\_nepesseg\_szama.MOBILELOFIZETESEK\_SZAMA';

/\*\*

\* the column name for the OSSZES\_ADOFIZETO field

\*/

const COL\_OSSZES\_ADOFIZETO = 'stg\_erintett\_nepesseg\_szama.OSSZES\_ADOFIZETO';

/\*\*

\* the column name for the OSSZES\_ADO field

\*/

const COL\_OSSZES\_ADO = 'stg\_erintett\_nepesseg\_szama.OSSZES\_ADO';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'RegisztraltBunelkovetokSzama', 'HaztartasiGazfogyasztokSzama', 'HaztartasiVillamosenergiaFogyasztokSzama', 'ElvandorlasokSzama', 'AllandoNepessegSzama', 'VallalkozasokSzama', 'NyilvantartottAllaskeresokSzama', 'Lakasallomany', 'BolcsodebeBeirtGyermekekSzama', 'AllandoNepessegbol1859EvesekSzama', 'MobilelofizetesekSzama', 'OsszesAdofizeto', 'OsszesAdo', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'regisztraltBunelkovetokSzama', 'haztartasiGazfogyasztokSzama', 'haztartasiVillamosenergiaFogyasztokSzama', 'elvandorlasokSzama', 'allandoNepessegSzama', 'vallalkozasokSzama', 'nyilvantartottAllaskeresokSzama', 'lakasallomany', 'bolcsodebeBeirtGyermekekSzama', 'allandoNepessegbol1859EvesekSzama', 'mobilelofizetesekSzama', 'osszesAdofizeto', 'osszesAdo', ),

self::TYPE\_COLNAME => array(StgErintettNepessegSzamaTableMap::COL\_TELEPULES\_NEV, StgErintettNepessegSzamaTableMap::COL\_TELEPULES\_KSHKOD, StgErintettNepessegSzamaTableMap::COL\_REGISZTRALT\_BUNELKOVETOK\_SZAMA, StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_GAZFOGYASZTOK\_SZAMA, StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA, StgErintettNepessegSzamaTableMap::COL\_ELVANDORLASOK\_SZAMA, StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA, StgErintettNepessegSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA, StgErintettNepessegSzamaTableMap::COL\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA, StgErintettNepessegSzamaTableMap::COL\_LAKASALLOMANY, StgErintettNepessegSzamaTableMap::COL\_BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA, StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA, StgErintettNepessegSzamaTableMap::COL\_MOBILELOFIZETESEK\_SZAMA, StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADOFIZETO, StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADO, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_REGISZTRALT\_BUNELKOVETOK\_SZAMA', 'COL\_HAZTARTASI\_GAZFOGYASZTOK\_SZAMA', 'COL\_HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA', 'COL\_ELVANDORLASOK\_SZAMA', 'COL\_ALLANDO\_NEPESSEG\_SZAMA', 'COL\_VALLALKOZASOK\_SZAMA', 'COL\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA', 'COL\_LAKASALLOMANY', 'COL\_BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA', 'COL\_ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA', 'COL\_MOBILELOFIZETESEK\_SZAMA', 'COL\_OSSZES\_ADOFIZETO', 'COL\_OSSZES\_ADO', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'regisztralt\_bunelkovetok\_szama', 'haztartasi\_gazfogyasztok\_szama', 'haztartasi\_villamosenergia\_fogyasztok\_szama', 'elvandorlasok\_szama', 'allando\_nepesseg\_szama', 'vallalkozasok\_szama', 'nyilvantartott\_allaskeresok\_szama', 'lakasallomany', 'bolcsodebe\_beirt\_gyermekek\_szama', 'allando\_nepessegbol\_18\_59\_evesek\_szama', 'mobilelofizetesek\_szama', 'osszes\_adofizeto', 'osszes\_ado', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'RegisztraltBunelkovetokSzama' => 2, 'HaztartasiGazfogyasztokSzama' => 3, 'HaztartasiVillamosenergiaFogyasztokSzama' => 4, 'ElvandorlasokSzama' => 5, 'AllandoNepessegSzama' => 6, 'VallalkozasokSzama' => 7, 'NyilvantartottAllaskeresokSzama' => 8, 'Lakasallomany' => 9, 'BolcsodebeBeirtGyermekekSzama' => 10, 'AllandoNepessegbol1859EvesekSzama' => 11, 'MobilelofizetesekSzama' => 12, 'OsszesAdofizeto' => 13, 'OsszesAdo' => 14, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'regisztraltBunelkovetokSzama' => 2, 'haztartasiGazfogyasztokSzama' => 3, 'haztartasiVillamosenergiaFogyasztokSzama' => 4, 'elvandorlasokSzama' => 5, 'allandoNepessegSzama' => 6, 'vallalkozasokSzama' => 7, 'nyilvantartottAllaskeresokSzama' => 8, 'lakasallomany' => 9, 'bolcsodebeBeirtGyermekekSzama' => 10, 'allandoNepessegbol1859EvesekSzama' => 11, 'mobilelofizetesekSzama' => 12, 'osszesAdofizeto' => 13, 'osszesAdo' => 14, ),

self::TYPE\_COLNAME => array(StgErintettNepessegSzamaTableMap::COL\_TELEPULES\_NEV => 0, StgErintettNepessegSzamaTableMap::COL\_TELEPULES\_KSHKOD => 1, StgErintettNepessegSzamaTableMap::COL\_REGISZTRALT\_BUNELKOVETOK\_SZAMA => 2, StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_GAZFOGYASZTOK\_SZAMA => 3, StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA => 4, StgErintettNepessegSzamaTableMap::COL\_ELVANDORLASOK\_SZAMA => 5, StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA => 6, StgErintettNepessegSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA => 7, StgErintettNepessegSzamaTableMap::COL\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA => 8, StgErintettNepessegSzamaTableMap::COL\_LAKASALLOMANY => 9, StgErintettNepessegSzamaTableMap::COL\_BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA => 10, StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA => 11, StgErintettNepessegSzamaTableMap::COL\_MOBILELOFIZETESEK\_SZAMA => 12, StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADOFIZETO => 13, StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADO => 14, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_REGISZTRALT\_BUNELKOVETOK\_SZAMA' => 2, 'COL\_HAZTARTASI\_GAZFOGYASZTOK\_SZAMA' => 3, 'COL\_HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA' => 4, 'COL\_ELVANDORLASOK\_SZAMA' => 5, 'COL\_ALLANDO\_NEPESSEG\_SZAMA' => 6, 'COL\_VALLALKOZASOK\_SZAMA' => 7, 'COL\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA' => 8, 'COL\_LAKASALLOMANY' => 9, 'COL\_BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA' => 10, 'COL\_ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA' => 11, 'COL\_MOBILELOFIZETESEK\_SZAMA' => 12, 'COL\_OSSZES\_ADOFIZETO' => 13, 'COL\_OSSZES\_ADO' => 14, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'regisztralt\_bunelkovetok\_szama' => 2, 'haztartasi\_gazfogyasztok\_szama' => 3, 'haztartasi\_villamosenergia\_fogyasztok\_szama' => 4, 'elvandorlasok\_szama' => 5, 'allando\_nepesseg\_szama' => 6, 'vallalkozasok\_szama' => 7, 'nyilvantartott\_allaskeresok\_szama' => 8, 'lakasallomany' => 9, 'bolcsodebe\_beirt\_gyermekek\_szama' => 10, 'allando\_nepessegbol\_18\_59\_evesek\_szama' => 11, 'mobilelofizetesek\_szama' => 12, 'osszes\_adofizeto' => 13, 'osszes\_ado' => 14, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_erintett\_nepesseg\_szama');

$this->setPhpName('StgErintettNepessegSzama');

$this->setClassName('\\StgErintettNepessegSzama');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('REGISZTRALT\_BUNELKOVETOK\_SZAMA', 'RegisztraltBunelkovetokSzama', 'INTEGER', false, null, null);

$this->addColumn('HAZTARTASI\_GAZFOGYASZTOK\_SZAMA', 'HaztartasiGazfogyasztokSzama', 'INTEGER', false, null, null);

$this->addColumn('HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA', 'HaztartasiVillamosenergiaFogyasztokSzama', 'INTEGER', false, null, null);

$this->addColumn('ELVANDORLASOK\_SZAMA', 'ElvandorlasokSzama', 'INTEGER', false, null, null);

$this->addColumn('ALLANDO\_NEPESSEG\_SZAMA', 'AllandoNepessegSzama', 'INTEGER', false, null, null);

$this->addColumn('VALLALKOZASOK\_SZAMA', 'VallalkozasokSzama', 'INTEGER', false, null, null);

$this->addColumn('NYILVANTARTOTT\_ALLASKERESOK\_SZAMA', 'NyilvantartottAllaskeresokSzama', 'INTEGER', false, null, null);

$this->addColumn('LAKASALLOMANY', 'Lakasallomany', 'INTEGER', false, null, null);

$this->addColumn('BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA', 'BolcsodebeBeirtGyermekekSzama', 'INTEGER', false, null, null);

$this->addColumn('ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA', 'AllandoNepessegbol1859EvesekSzama', 'INTEGER', false, null, null);

$this->addColumn('MOBILELOFIZETESEK\_SZAMA', 'MobilelofizetesekSzama', 'INTEGER', false, null, null);

$this->addColumn('OSSZES\_ADOFIZETO', 'OsszesAdofizeto', 'INTEGER', false, null, null);

$this->addColumn('OSSZES\_ADO', 'OsszesAdo', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgErintettNepessegSzamaTableMap::CLASS\_DEFAULT : StgErintettNepessegSzamaTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgErintettNepessegSzama object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgErintettNepessegSzamaTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgErintettNepessegSzamaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgErintettNepessegSzamaTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgErintettNepessegSzamaTableMap::OM\_CLASS;

/\*\* @var StgErintettNepessegSzama $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgErintettNepessegSzamaTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgErintettNepessegSzamaTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgErintettNepessegSzamaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgErintettNepessegSzama $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgErintettNepessegSzamaTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgErintettNepessegSzamaTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgErintettNepessegSzamaTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgErintettNepessegSzamaTableMap::COL\_REGISZTRALT\_BUNELKOVETOK\_SZAMA);

$criteria->addSelectColumn(StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_GAZFOGYASZTOK\_SZAMA);

$criteria->addSelectColumn(StgErintettNepessegSzamaTableMap::COL\_HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA);

$criteria->addSelectColumn(StgErintettNepessegSzamaTableMap::COL\_ELVANDORLASOK\_SZAMA);

$criteria->addSelectColumn(StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA);

$criteria->addSelectColumn(StgErintettNepessegSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA);

$criteria->addSelectColumn(StgErintettNepessegSzamaTableMap::COL\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA);

$criteria->addSelectColumn(StgErintettNepessegSzamaTableMap::COL\_LAKASALLOMANY);

$criteria->addSelectColumn(StgErintettNepessegSzamaTableMap::COL\_BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA);

$criteria->addSelectColumn(StgErintettNepessegSzamaTableMap::COL\_ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA);

$criteria->addSelectColumn(StgErintettNepessegSzamaTableMap::COL\_MOBILELOFIZETESEK\_SZAMA);

$criteria->addSelectColumn(StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADOFIZETO);

$criteria->addSelectColumn(StgErintettNepessegSzamaTableMap::COL\_OSSZES\_ADO);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.REGISZTRALT\_BUNELKOVETOK\_SZAMA');

$criteria->addSelectColumn($alias . '.HAZTARTASI\_GAZFOGYASZTOK\_SZAMA');

$criteria->addSelectColumn($alias . '.HAZTARTASI\_VILLAMOSENERGIA\_FOGYASZTOK\_SZAMA');

$criteria->addSelectColumn($alias . '.ELVANDORLASOK\_SZAMA');

$criteria->addSelectColumn($alias . '.ALLANDO\_NEPESSEG\_SZAMA');

$criteria->addSelectColumn($alias . '.VALLALKOZASOK\_SZAMA');

$criteria->addSelectColumn($alias . '.NYILVANTARTOTT\_ALLASKERESOK\_SZAMA');

$criteria->addSelectColumn($alias . '.LAKASALLOMANY');

$criteria->addSelectColumn($alias . '.BOLCSODEBE\_BEIRT\_GYERMEKEK\_SZAMA');

$criteria->addSelectColumn($alias . '.ALLANDO\_NEPESSEGBOL\_18\_59\_EVESEK\_SZAMA');

$criteria->addSelectColumn($alias . '.MOBILELOFIZETESEK\_SZAMA');

$criteria->addSelectColumn($alias . '.OSSZES\_ADOFIZETO');

$criteria->addSelectColumn($alias . '.OSSZES\_ADO');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgErintettNepessegSzamaTableMap::DATABASE\_NAME)->getTable(StgErintettNepessegSzamaTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgErintettNepessegSzamaTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgErintettNepessegSzamaTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgErintettNepessegSzamaTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgErintettNepessegSzama or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgErintettNepessegSzama object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgErintettNepessegSzamaTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgErintettNepessegSzama) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgErintettNepessegSzama object has no primary key');

}

$query = StgErintettNepessegSzamaQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgErintettNepessegSzamaTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgErintettNepessegSzamaTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_erintett\_nepesseg\_szama table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgErintettNepessegSzamaQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgErintettNepessegSzama or Criteria object.

\*

\* @param mixed $criteria Criteria or StgErintettNepessegSzama object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgErintettNepessegSzamaTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgErintettNepessegSzama object

}

// Set the correct dbName

$query = StgErintettNepessegSzamaQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgErintettNepessegSzamaTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgErintettNepessegSzamaTableMap::buildTableMap();

### StgErtekesitesNettoArbeveteleTableMap.php

<?php

namespace Map;

use \StgErtekesitesNettoArbevetele;

use \StgErtekesitesNettoArbeveteleQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_ertekesites\_netto\_arbevetele' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgErtekesitesNettoArbeveteleTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgErtekesitesNettoArbeveteleTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_ertekesites\_netto\_arbevetele';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgErtekesitesNettoArbevetele';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgErtekesitesNettoArbevetele';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 3;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 3;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_ertekesites\_netto\_arbevetele.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_ertekesites\_netto\_arbevetele.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the ERTEKESITES\_NETTO\_ARBEVETELE field

\*/

const COL\_ERTEKESITES\_NETTO\_ARBEVETELE = 'stg\_ertekesites\_netto\_arbevetele.ERTEKESITES\_NETTO\_ARBEVETELE';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'ErtekesitesNettoArbevetele', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'ertekesitesNettoArbevetele', ),

self::TYPE\_COLNAME => array(StgErtekesitesNettoArbeveteleTableMap::COL\_TELEPULES\_NEV, StgErtekesitesNettoArbeveteleTableMap::COL\_TELEPULES\_KSHKOD, StgErtekesitesNettoArbeveteleTableMap::COL\_ERTEKESITES\_NETTO\_ARBEVETELE, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_ERTEKESITES\_NETTO\_ARBEVETELE', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'ertekesites\_netto\_arbevetele', ),

self::TYPE\_NUM => array(0, 1, 2, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'ErtekesitesNettoArbevetele' => 2, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'ertekesitesNettoArbevetele' => 2, ),

self::TYPE\_COLNAME => array(StgErtekesitesNettoArbeveteleTableMap::COL\_TELEPULES\_NEV => 0, StgErtekesitesNettoArbeveteleTableMap::COL\_TELEPULES\_KSHKOD => 1, StgErtekesitesNettoArbeveteleTableMap::COL\_ERTEKESITES\_NETTO\_ARBEVETELE => 2, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_ERTEKESITES\_NETTO\_ARBEVETELE' => 2, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'ertekesites\_netto\_arbevetele' => 2, ),

self::TYPE\_NUM => array(0, 1, 2, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_ertekesites\_netto\_arbevetele');

$this->setPhpName('StgErtekesitesNettoArbevetele');

$this->setClassName('\\StgErtekesitesNettoArbevetele');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('ERTEKESITES\_NETTO\_ARBEVETELE', 'ErtekesitesNettoArbevetele', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgErtekesitesNettoArbeveteleTableMap::CLASS\_DEFAULT : StgErtekesitesNettoArbeveteleTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgErtekesitesNettoArbevetele object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgErtekesitesNettoArbeveteleTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgErtekesitesNettoArbeveteleTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgErtekesitesNettoArbeveteleTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgErtekesitesNettoArbeveteleTableMap::OM\_CLASS;

/\*\* @var StgErtekesitesNettoArbevetele $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgErtekesitesNettoArbeveteleTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgErtekesitesNettoArbeveteleTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgErtekesitesNettoArbeveteleTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgErtekesitesNettoArbevetele $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgErtekesitesNettoArbeveteleTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgErtekesitesNettoArbeveteleTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgErtekesitesNettoArbeveteleTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgErtekesitesNettoArbeveteleTableMap::COL\_ERTEKESITES\_NETTO\_ARBEVETELE);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.ERTEKESITES\_NETTO\_ARBEVETELE');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgErtekesitesNettoArbeveteleTableMap::DATABASE\_NAME)->getTable(StgErtekesitesNettoArbeveteleTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgErtekesitesNettoArbeveteleTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgErtekesitesNettoArbeveteleTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgErtekesitesNettoArbeveteleTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgErtekesitesNettoArbevetele or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgErtekesitesNettoArbevetele object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgErtekesitesNettoArbeveteleTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgErtekesitesNettoArbevetele) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgErtekesitesNettoArbevetele object has no primary key');

}

$query = StgErtekesitesNettoArbeveteleQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgErtekesitesNettoArbeveteleTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgErtekesitesNettoArbeveteleTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_ertekesites\_netto\_arbevetele table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgErtekesitesNettoArbeveteleQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgErtekesitesNettoArbevetele or Criteria object.

\*

\* @param mixed $criteria Criteria or StgErtekesitesNettoArbevetele object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgErtekesitesNettoArbeveteleTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgErtekesitesNettoArbevetele object

}

// Set the correct dbName

$query = StgErtekesitesNettoArbeveteleQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgErtekesitesNettoArbeveteleTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgErtekesitesNettoArbeveteleTableMap::buildTableMap();

### StgFoglalkoztatottakSzamaTableMap.php

<?php

namespace Map;

use \StgFoglalkoztatottakSzama;

use \StgFoglalkoztatottakSzamaQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_foglalkoztatottak\_szama' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgFoglalkoztatottakSzamaTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgFoglalkoztatottakSzamaTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_foglalkoztatottak\_szama';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgFoglalkoztatottakSzama';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgFoglalkoztatottakSzama';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 4;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 4;

/\*\*

\* the column name for the KISTERSEG\_NEV field

\*/

const COL\_KISTERSEG\_NEV = 'stg\_foglalkoztatottak\_szama.KISTERSEG\_NEV';

/\*\*

\* the column name for the KISTERSEG\_KSHKOD field

\*/

const COL\_KISTERSEG\_KSHKOD = 'stg\_foglalkoztatottak\_szama.KISTERSEG\_KSHKOD';

/\*\*

\* the column name for the FOGLALKOZTATOTTAK\_SZAMA field

\*/

const COL\_FOGLALKOZTATOTTAK\_SZAMA = 'stg\_foglalkoztatottak\_szama.FOGLALKOZTATOTTAK\_SZAMA';

/\*\*

\* the column name for the KISTERSEGI\_FOGLALKOZTATASI\_RATA field

\*/

const COL\_KISTERSEGI\_FOGLALKOZTATASI\_RATA = 'stg\_foglalkoztatottak\_szama.KISTERSEGI\_FOGLALKOZTATASI\_RATA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('KistersegNev', 'KistersegKshkod', 'FoglalkoztatottakSzama', 'KistersegiFoglalkoztatasiRata', ),

self::TYPE\_STUDLYPHPNAME => array('kistersegNev', 'kistersegKshkod', 'foglalkoztatottakSzama', 'kistersegiFoglalkoztatasiRata', ),

self::TYPE\_COLNAME => array(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEG\_NEV, StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEG\_KSHKOD, StgFoglalkoztatottakSzamaTableMap::COL\_FOGLALKOZTATOTTAK\_SZAMA, StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEGI\_FOGLALKOZTATASI\_RATA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_KISTERSEG\_NEV', 'COL\_KISTERSEG\_KSHKOD', 'COL\_FOGLALKOZTATOTTAK\_SZAMA', 'COL\_KISTERSEGI\_FOGLALKOZTATASI\_RATA', ),

self::TYPE\_FIELDNAME => array('kisterseg\_nev', 'kisterseg\_KSHKOD', 'foglalkoztatottak\_szama', 'kistersegi\_foglalkoztatasi\_rata', ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('KistersegNev' => 0, 'KistersegKshkod' => 1, 'FoglalkoztatottakSzama' => 2, 'KistersegiFoglalkoztatasiRata' => 3, ),

self::TYPE\_STUDLYPHPNAME => array('kistersegNev' => 0, 'kistersegKshkod' => 1, 'foglalkoztatottakSzama' => 2, 'kistersegiFoglalkoztatasiRata' => 3, ),

self::TYPE\_COLNAME => array(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEG\_NEV => 0, StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEG\_KSHKOD => 1, StgFoglalkoztatottakSzamaTableMap::COL\_FOGLALKOZTATOTTAK\_SZAMA => 2, StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEGI\_FOGLALKOZTATASI\_RATA => 3, ),

self::TYPE\_RAW\_COLNAME => array('COL\_KISTERSEG\_NEV' => 0, 'COL\_KISTERSEG\_KSHKOD' => 1, 'COL\_FOGLALKOZTATOTTAK\_SZAMA' => 2, 'COL\_KISTERSEGI\_FOGLALKOZTATASI\_RATA' => 3, ),

self::TYPE\_FIELDNAME => array('kisterseg\_nev' => 0, 'kisterseg\_KSHKOD' => 1, 'foglalkoztatottak\_szama' => 2, 'kistersegi\_foglalkoztatasi\_rata' => 3, ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_foglalkoztatottak\_szama');

$this->setPhpName('StgFoglalkoztatottakSzama');

$this->setClassName('\\StgFoglalkoztatottakSzama');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('KISTERSEG\_NEV', 'KistersegNev', 'VARCHAR', true, 45, null);

$this->addColumn('KISTERSEG\_KSHKOD', 'KistersegKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('FOGLALKOZTATOTTAK\_SZAMA', 'FoglalkoztatottakSzama', 'INTEGER', false, null, null);

$this->addColumn('KISTERSEGI\_FOGLALKOZTATASI\_RATA', 'KistersegiFoglalkoztatasiRata', 'DOUBLE', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgFoglalkoztatottakSzamaTableMap::CLASS\_DEFAULT : StgFoglalkoztatottakSzamaTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgFoglalkoztatottakSzama object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgFoglalkoztatottakSzamaTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgFoglalkoztatottakSzamaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgFoglalkoztatottakSzamaTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgFoglalkoztatottakSzamaTableMap::OM\_CLASS;

/\*\* @var StgFoglalkoztatottakSzama $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgFoglalkoztatottakSzamaTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgFoglalkoztatottakSzamaTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgFoglalkoztatottakSzamaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgFoglalkoztatottakSzama $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgFoglalkoztatottakSzamaTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEG\_NEV);

$criteria->addSelectColumn(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEG\_KSHKOD);

$criteria->addSelectColumn(StgFoglalkoztatottakSzamaTableMap::COL\_FOGLALKOZTATOTTAK\_SZAMA);

$criteria->addSelectColumn(StgFoglalkoztatottakSzamaTableMap::COL\_KISTERSEGI\_FOGLALKOZTATASI\_RATA);

} else {

$criteria->addSelectColumn($alias . '.KISTERSEG\_NEV');

$criteria->addSelectColumn($alias . '.KISTERSEG\_KSHKOD');

$criteria->addSelectColumn($alias . '.FOGLALKOZTATOTTAK\_SZAMA');

$criteria->addSelectColumn($alias . '.KISTERSEGI\_FOGLALKOZTATASI\_RATA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgFoglalkoztatottakSzamaTableMap::DATABASE\_NAME)->getTable(StgFoglalkoztatottakSzamaTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgFoglalkoztatottakSzamaTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgFoglalkoztatottakSzamaTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgFoglalkoztatottakSzamaTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgFoglalkoztatottakSzama or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgFoglalkoztatottakSzama object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgFoglalkoztatottakSzamaTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgFoglalkoztatottakSzama) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgFoglalkoztatottakSzama object has no primary key');

}

$query = StgFoglalkoztatottakSzamaQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgFoglalkoztatottakSzamaTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgFoglalkoztatottakSzamaTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_foglalkoztatottak\_szama table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgFoglalkoztatottakSzamaQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgFoglalkoztatottakSzama or Criteria object.

\*

\* @param mixed $criteria Criteria or StgFoglalkoztatottakSzama object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgFoglalkoztatottakSzamaTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgFoglalkoztatottakSzama object

}

// Set the correct dbName

$query = StgFoglalkoztatottakSzamaQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgFoglalkoztatottakSzamaTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgFoglalkoztatottakSzamaTableMap::buildTableMap();

### StgJovedelemEsAllaskeresokSzamaTableMap.php

<?php

namespace Map;

use \StgJovedelemEsAllaskeresokSzama;

use \StgJovedelemEsAllaskeresokSzamaQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_jovedelem\_es\_allaskeresok\_szama' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgJovedelemEsAllaskeresokSzamaTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgJovedelemEsAllaskeresokSzamaTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_jovedelem\_es\_allaskeresok\_szama';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgJovedelemEsAllaskeresokSzama';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgJovedelemEsAllaskeresokSzama';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 5;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 5;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_jovedelem\_es\_allaskeresok\_szama.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_jovedelem\_es\_allaskeresok\_szama.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the OSSZES\_JOVEDELEM\_FOALLASBOL field

\*/

const COL\_OSSZES\_JOVEDELEM\_FOALLASBOL = 'stg\_jovedelem\_es\_allaskeresok\_szama.OSSZES\_JOVEDELEM\_FOALLASBOL';

/\*\*

\* the column name for the ALLANDO\_NEPESSEG\_SZAMA field

\*/

const COL\_ALLANDO\_NEPESSEG\_SZAMA = 'stg\_jovedelem\_es\_allaskeresok\_szama.ALLANDO\_NEPESSEG\_SZAMA';

/\*\*

\* the column name for the P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA field

\*/

const COL\_P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA = 'stg\_jovedelem\_es\_allaskeresok\_szama.P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'OsszesJovedelemFoallasbol', 'AllandoNepessegSzama', 'P180NaponTuliNyilvantartottAllaskeresokSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'osszesJovedelemFoallasbol', 'allandoNepessegSzama', 'p180NaponTuliNyilvantartottAllaskeresokSzama', ),

self::TYPE\_COLNAME => array(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_TELEPULES\_NEV, StgJovedelemEsAllaskeresokSzamaTableMap::COL\_TELEPULES\_KSHKOD, StgJovedelemEsAllaskeresokSzamaTableMap::COL\_OSSZES\_JOVEDELEM\_FOALLASBOL, StgJovedelemEsAllaskeresokSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA, StgJovedelemEsAllaskeresokSzamaTableMap::COL\_P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_OSSZES\_JOVEDELEM\_FOALLASBOL', 'COL\_ALLANDO\_NEPESSEG\_SZAMA', 'COL\_P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'osszes\_jovedelem\_foallasbol', 'allando\_nepesseg\_szama', 'p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'OsszesJovedelemFoallasbol' => 2, 'AllandoNepessegSzama' => 3, 'P180NaponTuliNyilvantartottAllaskeresokSzama' => 4, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'osszesJovedelemFoallasbol' => 2, 'allandoNepessegSzama' => 3, 'p180NaponTuliNyilvantartottAllaskeresokSzama' => 4, ),

self::TYPE\_COLNAME => array(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_TELEPULES\_NEV => 0, StgJovedelemEsAllaskeresokSzamaTableMap::COL\_TELEPULES\_KSHKOD => 1, StgJovedelemEsAllaskeresokSzamaTableMap::COL\_OSSZES\_JOVEDELEM\_FOALLASBOL => 2, StgJovedelemEsAllaskeresokSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA => 3, StgJovedelemEsAllaskeresokSzamaTableMap::COL\_P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA => 4, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_OSSZES\_JOVEDELEM\_FOALLASBOL' => 2, 'COL\_ALLANDO\_NEPESSEG\_SZAMA' => 3, 'COL\_P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA' => 4, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'osszes\_jovedelem\_foallasbol' => 2, 'allando\_nepesseg\_szama' => 3, 'p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama' => 4, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_jovedelem\_es\_allaskeresok\_szama');

$this->setPhpName('StgJovedelemEsAllaskeresokSzama');

$this->setClassName('\\StgJovedelemEsAllaskeresokSzama');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('OSSZES\_JOVEDELEM\_FOALLASBOL', 'OsszesJovedelemFoallasbol', 'INTEGER', false, null, null);

$this->addColumn('ALLANDO\_NEPESSEG\_SZAMA', 'AllandoNepessegSzama', 'INTEGER', false, null, null);

$this->addColumn('P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA', 'P180NaponTuliNyilvantartottAllaskeresokSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgJovedelemEsAllaskeresokSzamaTableMap::CLASS\_DEFAULT : StgJovedelemEsAllaskeresokSzamaTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgJovedelemEsAllaskeresokSzama object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgJovedelemEsAllaskeresokSzamaTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgJovedelemEsAllaskeresokSzamaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgJovedelemEsAllaskeresokSzamaTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgJovedelemEsAllaskeresokSzamaTableMap::OM\_CLASS;

/\*\* @var StgJovedelemEsAllaskeresokSzama $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgJovedelemEsAllaskeresokSzamaTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgJovedelemEsAllaskeresokSzamaTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgJovedelemEsAllaskeresokSzamaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgJovedelemEsAllaskeresokSzama $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgJovedelemEsAllaskeresokSzamaTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_OSSZES\_JOVEDELEM\_FOALLASBOL);

$criteria->addSelectColumn(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_ALLANDO\_NEPESSEG\_SZAMA);

$criteria->addSelectColumn(StgJovedelemEsAllaskeresokSzamaTableMap::COL\_P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.OSSZES\_JOVEDELEM\_FOALLASBOL');

$criteria->addSelectColumn($alias . '.ALLANDO\_NEPESSEG\_SZAMA');

$criteria->addSelectColumn($alias . '.P180\_NAPON\_TULI\_NYILVANTARTOTT\_ALLASKERESOK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgJovedelemEsAllaskeresokSzamaTableMap::DATABASE\_NAME)->getTable(StgJovedelemEsAllaskeresokSzamaTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgJovedelemEsAllaskeresokSzamaTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgJovedelemEsAllaskeresokSzamaTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgJovedelemEsAllaskeresokSzamaTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgJovedelemEsAllaskeresokSzama or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgJovedelemEsAllaskeresokSzama object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgJovedelemEsAllaskeresokSzamaTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgJovedelemEsAllaskeresokSzama) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgJovedelemEsAllaskeresokSzama object has no primary key');

}

$query = StgJovedelemEsAllaskeresokSzamaQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgJovedelemEsAllaskeresokSzamaTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgJovedelemEsAllaskeresokSzamaTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_jovedelem\_es\_allaskeresok\_szama table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgJovedelemEsAllaskeresokSzamaQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgJovedelemEsAllaskeresokSzama or Criteria object.

\*

\* @param mixed $criteria Criteria or StgJovedelemEsAllaskeresokSzama object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgJovedelemEsAllaskeresokSzamaTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgJovedelemEsAllaskeresokSzama object

}

// Set the correct dbName

$query = StgJovedelemEsAllaskeresokSzamaQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgJovedelemEsAllaskeresokSzamaTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgJovedelemEsAllaskeresokSzamaTableMap::buildTableMap();

### StgJovedelmiHelyzetTableMap.php

<?php

namespace Map;

use \StgJovedelmiHelyzet;

use \StgJovedelmiHelyzetQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_jovedelmi\_helyzet' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgJovedelmiHelyzetTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgJovedelmiHelyzetTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_jovedelmi\_helyzet';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgJovedelmiHelyzet';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgJovedelmiHelyzet';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 4;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 4;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_jovedelmi\_helyzet.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_jovedelmi\_helyzet.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the OSSZES\_ADO field

\*/

const COL\_OSSZES\_ADO = 'stg\_jovedelmi\_helyzet.OSSZES\_ADO';

/\*\*

\* the column name for the ADOFIZETOK\_SZAMA field

\*/

const COL\_ADOFIZETOK\_SZAMA = 'stg\_jovedelmi\_helyzet.ADOFIZETOK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'OsszesAdo', 'AdofizetokSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'osszesAdo', 'adofizetokSzama', ),

self::TYPE\_COLNAME => array(StgJovedelmiHelyzetTableMap::COL\_TELEPULES\_NEV, StgJovedelmiHelyzetTableMap::COL\_TELEPULES\_KSHKOD, StgJovedelmiHelyzetTableMap::COL\_OSSZES\_ADO, StgJovedelmiHelyzetTableMap::COL\_ADOFIZETOK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_OSSZES\_ADO', 'COL\_ADOFIZETOK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'osszes\_ado', 'adofizetok\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'OsszesAdo' => 2, 'AdofizetokSzama' => 3, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'osszesAdo' => 2, 'adofizetokSzama' => 3, ),

self::TYPE\_COLNAME => array(StgJovedelmiHelyzetTableMap::COL\_TELEPULES\_NEV => 0, StgJovedelmiHelyzetTableMap::COL\_TELEPULES\_KSHKOD => 1, StgJovedelmiHelyzetTableMap::COL\_OSSZES\_ADO => 2, StgJovedelmiHelyzetTableMap::COL\_ADOFIZETOK\_SZAMA => 3, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_OSSZES\_ADO' => 2, 'COL\_ADOFIZETOK\_SZAMA' => 3, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'osszes\_ado' => 2, 'adofizetok\_szama' => 3, ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_jovedelmi\_helyzet');

$this->setPhpName('StgJovedelmiHelyzet');

$this->setClassName('\\StgJovedelmiHelyzet');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('OSSZES\_ADO', 'OsszesAdo', 'INTEGER', false, null, null);

$this->addColumn('ADOFIZETOK\_SZAMA', 'AdofizetokSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgJovedelmiHelyzetTableMap::CLASS\_DEFAULT : StgJovedelmiHelyzetTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgJovedelmiHelyzet object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgJovedelmiHelyzetTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgJovedelmiHelyzetTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgJovedelmiHelyzetTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgJovedelmiHelyzetTableMap::OM\_CLASS;

/\*\* @var StgJovedelmiHelyzet $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgJovedelmiHelyzetTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgJovedelmiHelyzetTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgJovedelmiHelyzetTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgJovedelmiHelyzet $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgJovedelmiHelyzetTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgJovedelmiHelyzetTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgJovedelmiHelyzetTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgJovedelmiHelyzetTableMap::COL\_OSSZES\_ADO);

$criteria->addSelectColumn(StgJovedelmiHelyzetTableMap::COL\_ADOFIZETOK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.OSSZES\_ADO');

$criteria->addSelectColumn($alias . '.ADOFIZETOK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgJovedelmiHelyzetTableMap::DATABASE\_NAME)->getTable(StgJovedelmiHelyzetTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgJovedelmiHelyzetTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgJovedelmiHelyzetTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgJovedelmiHelyzetTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgJovedelmiHelyzet or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgJovedelmiHelyzet object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgJovedelmiHelyzetTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgJovedelmiHelyzet) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgJovedelmiHelyzet object has no primary key');

}

$query = StgJovedelmiHelyzetQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgJovedelmiHelyzetTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgJovedelmiHelyzetTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_jovedelmi\_helyzet table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgJovedelmiHelyzetQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgJovedelmiHelyzet or Criteria object.

\*

\* @param mixed $criteria Criteria or StgJovedelmiHelyzet object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgJovedelmiHelyzetTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgJovedelmiHelyzet object

}

// Set the correct dbName

$query = StgJovedelmiHelyzetQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgJovedelmiHelyzetTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgJovedelmiHelyzetTableMap::buildTableMap();

### StgKeresztmetszetiForgalomTableMap.php

<?php

namespace Map;

use \StgKeresztmetszetiForgalom;

use \StgKeresztmetszetiForgalomQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_keresztmetszeti\_forgalom' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgKeresztmetszetiForgalomTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgKeresztmetszetiForgalomTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_keresztmetszeti\_forgalom';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgKeresztmetszetiForgalom';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgKeresztmetszetiForgalom';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 44;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 44;

/\*\*

\* the column name for the SHAPENR field

\*/

const COL\_SHAPENR = 'stg\_keresztmetszeti\_forgalom.SHAPENR';

/\*\*

\* the column name for the PARTNR field

\*/

const COL\_PARTNR = 'stg\_keresztmetszeti\_forgalom.PARTNR';

/\*\*

\* the column name for the NRPARTS field

\*/

const COL\_NRPARTS = 'stg\_keresztmetszeti\_forgalom.NRPARTS';

/\*\*

\* the column name for the POINTNR field

\*/

const COL\_POINTNR = 'stg\_keresztmetszeti\_forgalom.POINTNR';

/\*\*

\* the column name for the NRPOINTS field

\*/

const COL\_NRPOINTS = 'stg\_keresztmetszeti\_forgalom.NRPOINTS';

/\*\*

\* the column name for the X field

\*/

const COL\_X = 'stg\_keresztmetszeti\_forgalom.X';

/\*\*

\* the column name for the Y field

\*/

const COL\_Y = 'stg\_keresztmetszeti\_forgalom.Y';

/\*\*

\* the column name for the MEASURE field

\*/

const COL\_MEASURE = 'stg\_keresztmetszeti\_forgalom.MEASURE';

/\*\*

\* the column name for the KSZAM field

\*/

const COL\_KSZAM = 'stg\_keresztmetszeti\_forgalom.KSZAM';

/\*\*

\* the column name for the PKOD field

\*/

const COL\_PKOD = 'stg\_keresztmetszeti\_forgalom.PKOD';

/\*\*

\* the column name for the KKOD field

\*/

const COL\_KKOD = 'stg\_keresztmetszeti\_forgalom.KKOD';

/\*\*

\* the column name for the VVKOD field

\*/

const COL\_VVKOD = 'stg\_keresztmetszeti\_forgalom.VVKOD';

/\*\*

\* the column name for the KSZELV field

\*/

const COL\_KSZELV = 'stg\_keresztmetszeti\_forgalom.KSZELV';

/\*\*

\* the column name for the VSZELV field

\*/

const COL\_VSZELV = 'stg\_keresztmetszeti\_forgalom.VSZELV';

/\*\*

\* the column name for the RSHOSSZ field

\*/

const COL\_RSHOSSZ = 'stg\_keresztmetszeti\_forgalom.RSHOSSZ';

/\*\*

\* the column name for the ANF field

\*/

const COL\_ANF = 'stg\_keresztmetszeti\_forgalom.ANF';

/\*\*

\* the column name for the ANET field

\*/

const COL\_ANET = 'stg\_keresztmetszeti\_forgalom.ANET';

/\*\*

\* the column name for the MOF field

\*/

const COL\_MOF = 'stg\_keresztmetszeti\_forgalom.MOF';

/\*\*

\* the column name for the ONGJ field

\*/

const COL\_ONGJ = 'stg\_keresztmetszeti\_forgalom.ONGJ';

/\*\*

\* the column name for the OJ field

\*/

const COL\_OJ = 'stg\_keresztmetszeti\_forgalom.OJ';

/\*\*

\* the column name for the OMOT field

\*/

const COL\_OMOT = 'stg\_keresztmetszeti\_forgalom.OMOT';

/\*\*

\* the column name for the EV field

\*/

const COL\_EV = 'stg\_keresztmetszeti\_forgalom.EV';

/\*\*

\* the column name for the ASZ field

\*/

const COL\_ASZ = 'stg\_keresztmetszeti\_forgalom.ASZ';

/\*\*

\* the column name for the BUSZCS field

\*/

const COL\_BUSZCS = 'stg\_keresztmetszeti\_forgalom.BUSZCS';

/\*\*

\* the column name for the BUSZE field

\*/

const COL\_BUSZE = 'stg\_keresztmetszeti\_forgalom.BUSZE';

/\*\*

\* the column name for the OBUSZ field

\*/

const COL\_OBUSZ = 'stg\_keresztmetszeti\_forgalom.OBUSZ';

/\*\*

\* the column name for the NYSZER field

\*/

const COL\_NYSZER = 'stg\_keresztmetszeti\_forgalom.NYSZER';

/\*\*

\* the column name for the POTKTGK field

\*/

const COL\_POTKTGK = 'stg\_keresztmetszeti\_forgalom.POTKTGK';

/\*\*

\* the column name for the KTGK field

\*/

const COL\_KTGK = 'stg\_keresztmetszeti\_forgalom.KTGK';

/\*\*

\* the column name for the NTGK field

\*/

const COL\_NTGK = 'stg\_keresztmetszeti\_forgalom.NTGK';

/\*\*

\* the column name for the KNTGK field

\*/

const COL\_KNTGK = 'stg\_keresztmetszeti\_forgalom.KNTGK';

/\*\*

\* the column name for the OTGK field

\*/

const COL\_OTGK = 'stg\_keresztmetszeti\_forgalom.OTGK';

/\*\*

\* the column name for the SZGK field

\*/

const COL\_SZGK = 'stg\_keresztmetszeti\_forgalom.SZGK';

/\*\*

\* the column name for the OSZGK field

\*/

const COL\_OSZGK = 'stg\_keresztmetszeti\_forgalom.OSZGK';

/\*\*

\* the column name for the MKP field

\*/

const COL\_MKP = 'stg\_keresztmetszeti\_forgalom.MKP';

/\*\*

\* the column name for the KPF field

\*/

const COL\_KPF = 'stg\_keresztmetszeti\_forgalom.KPF';

/\*\*

\* the column name for the LASSU field

\*/

const COL\_LASSU = 'stg\_keresztmetszeti\_forgalom.LASSU';

/\*\*

\* the column name for the SPEC field

\*/

const COL\_SPEC = 'stg\_keresztmetszeti\_forgalom.SPEC';

/\*\*

\* the column name for the FMEGB field

\*/

const COL\_FMEGB = 'stg\_keresztmetszeti\_forgalom.FMEGB';

/\*\*

\* the column name for the ADATFORR field

\*/

const COL\_ADATFORR = 'stg\_keresztmetszeti\_forgalom.ADATFORR';

/\*\*

\* the column name for the SZAMLNAP field

\*/

const COL\_SZAMLNAP = 'stg\_keresztmetszeti\_forgalom.SZAMLNAP';

/\*\*

\* the column name for the JELLEG1 field

\*/

const COL\_JELLEG1 = 'stg\_keresztmetszeti\_forgalom.JELLEG1';

/\*\*

\* the column name for the JELLEG2 field

\*/

const COL\_JELLEG2 = 'stg\_keresztmetszeti\_forgalom.JELLEG2';

/\*\*

\* the column name for the FMEGJ field

\*/

const COL\_FMEGJ = 'stg\_keresztmetszeti\_forgalom.FMEGJ';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('Shapenr', 'Partnr', 'Nrparts', 'Pointnr', 'Nrpoints', 'X', 'Y', 'Measure', 'Kszam', 'Pkod', 'Kkod', 'Vvkod', 'Kszelv', 'Vszelv', 'Rshossz', 'Anf', 'Anet', 'Mof', 'Ongj', 'Oj', 'Omot', 'Ev', 'Asz', 'Buszcs', 'Busze', 'Obusz', 'Nyszer', 'Potktgk', 'Ktgk', 'Ntgk', 'Kntgk', 'Otgk', 'Szgk', 'Oszgk', 'Mkp', 'Kpf', 'Lassu', 'Spec', 'Fmegb', 'Adatforr', 'Szamlnap', 'Jelleg1', 'Jelleg2', 'Fmegj', ),

self::TYPE\_STUDLYPHPNAME => array('shapenr', 'partnr', 'nrparts', 'pointnr', 'nrpoints', 'x', 'y', 'measure', 'kszam', 'pkod', 'kkod', 'vvkod', 'kszelv', 'vszelv', 'rshossz', 'anf', 'anet', 'mof', 'ongj', 'oj', 'omot', 'ev', 'asz', 'buszcs', 'busze', 'obusz', 'nyszer', 'potktgk', 'ktgk', 'ntgk', 'kntgk', 'otgk', 'szgk', 'oszgk', 'mkp', 'kpf', 'lassu', 'spec', 'fmegb', 'adatforr', 'szamlnap', 'jelleg1', 'jelleg2', 'fmegj', ),

self::TYPE\_COLNAME => array(StgKeresztmetszetiForgalomTableMap::COL\_SHAPENR, StgKeresztmetszetiForgalomTableMap::COL\_PARTNR, StgKeresztmetszetiForgalomTableMap::COL\_NRPARTS, StgKeresztmetszetiForgalomTableMap::COL\_POINTNR, StgKeresztmetszetiForgalomTableMap::COL\_NRPOINTS, StgKeresztmetszetiForgalomTableMap::COL\_X, StgKeresztmetszetiForgalomTableMap::COL\_Y, StgKeresztmetszetiForgalomTableMap::COL\_MEASURE, StgKeresztmetszetiForgalomTableMap::COL\_KSZAM, StgKeresztmetszetiForgalomTableMap::COL\_PKOD, StgKeresztmetszetiForgalomTableMap::COL\_KKOD, StgKeresztmetszetiForgalomTableMap::COL\_VVKOD, StgKeresztmetszetiForgalomTableMap::COL\_KSZELV, StgKeresztmetszetiForgalomTableMap::COL\_VSZELV, StgKeresztmetszetiForgalomTableMap::COL\_RSHOSSZ, StgKeresztmetszetiForgalomTableMap::COL\_ANF, StgKeresztmetszetiForgalomTableMap::COL\_ANET, StgKeresztmetszetiForgalomTableMap::COL\_MOF, StgKeresztmetszetiForgalomTableMap::COL\_ONGJ, StgKeresztmetszetiForgalomTableMap::COL\_OJ, StgKeresztmetszetiForgalomTableMap::COL\_OMOT, StgKeresztmetszetiForgalomTableMap::COL\_EV, StgKeresztmetszetiForgalomTableMap::COL\_ASZ, StgKeresztmetszetiForgalomTableMap::COL\_BUSZCS, StgKeresztmetszetiForgalomTableMap::COL\_BUSZE, StgKeresztmetszetiForgalomTableMap::COL\_OBUSZ, StgKeresztmetszetiForgalomTableMap::COL\_NYSZER, StgKeresztmetszetiForgalomTableMap::COL\_POTKTGK, StgKeresztmetszetiForgalomTableMap::COL\_KTGK, StgKeresztmetszetiForgalomTableMap::COL\_NTGK, StgKeresztmetszetiForgalomTableMap::COL\_KNTGK, StgKeresztmetszetiForgalomTableMap::COL\_OTGK, StgKeresztmetszetiForgalomTableMap::COL\_SZGK, StgKeresztmetszetiForgalomTableMap::COL\_OSZGK, StgKeresztmetszetiForgalomTableMap::COL\_MKP, StgKeresztmetszetiForgalomTableMap::COL\_KPF, StgKeresztmetszetiForgalomTableMap::COL\_LASSU, StgKeresztmetszetiForgalomTableMap::COL\_SPEC, StgKeresztmetszetiForgalomTableMap::COL\_FMEGB, StgKeresztmetszetiForgalomTableMap::COL\_ADATFORR, StgKeresztmetszetiForgalomTableMap::COL\_SZAMLNAP, StgKeresztmetszetiForgalomTableMap::COL\_JELLEG1, StgKeresztmetszetiForgalomTableMap::COL\_JELLEG2, StgKeresztmetszetiForgalomTableMap::COL\_FMEGJ, ),

self::TYPE\_RAW\_COLNAME => array('COL\_SHAPENR', 'COL\_PARTNR', 'COL\_NRPARTS', 'COL\_POINTNR', 'COL\_NRPOINTS', 'COL\_X', 'COL\_Y', 'COL\_MEASURE', 'COL\_KSZAM', 'COL\_PKOD', 'COL\_KKOD', 'COL\_VVKOD', 'COL\_KSZELV', 'COL\_VSZELV', 'COL\_RSHOSSZ', 'COL\_ANF', 'COL\_ANET', 'COL\_MOF', 'COL\_ONGJ', 'COL\_OJ', 'COL\_OMOT', 'COL\_EV', 'COL\_ASZ', 'COL\_BUSZCS', 'COL\_BUSZE', 'COL\_OBUSZ', 'COL\_NYSZER', 'COL\_POTKTGK', 'COL\_KTGK', 'COL\_NTGK', 'COL\_KNTGK', 'COL\_OTGK', 'COL\_SZGK', 'COL\_OSZGK', 'COL\_MKP', 'COL\_KPF', 'COL\_LASSU', 'COL\_SPEC', 'COL\_FMEGB', 'COL\_ADATFORR', 'COL\_SZAMLNAP', 'COL\_JELLEG1', 'COL\_JELLEG2', 'COL\_FMEGJ', ),

self::TYPE\_FIELDNAME => array('shapenr', 'partnr', 'nrparts', 'pointnr', 'nrpointS', 'x', 'y', 'measure', 'kszam', 'pkod', 'kkod', 'vvkod', 'kszelv', 'vszelv', 'RSHOSSZ', 'ANF', 'ANET', 'MOF', 'ONGJ', 'OJ', 'OMOT', 'EV', 'ASZ', 'BUSZCS', 'BUSZE', 'OBUSZ', 'NYSZER', 'POTKTGK', 'KTGK', 'NTGK', 'KNTGK', 'OTGK', 'SZGK', 'OSZGK', 'MKP', 'KPF', 'LASSU', 'SPEC', 'FMEGB', 'ADATFORR', 'SZAMLNAP', 'JELLEG1', 'JELLEG2', 'FMEGJ', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('Shapenr' => 0, 'Partnr' => 1, 'Nrparts' => 2, 'Pointnr' => 3, 'Nrpoints' => 4, 'X' => 5, 'Y' => 6, 'Measure' => 7, 'Kszam' => 8, 'Pkod' => 9, 'Kkod' => 10, 'Vvkod' => 11, 'Kszelv' => 12, 'Vszelv' => 13, 'Rshossz' => 14, 'Anf' => 15, 'Anet' => 16, 'Mof' => 17, 'Ongj' => 18, 'Oj' => 19, 'Omot' => 20, 'Ev' => 21, 'Asz' => 22, 'Buszcs' => 23, 'Busze' => 24, 'Obusz' => 25, 'Nyszer' => 26, 'Potktgk' => 27, 'Ktgk' => 28, 'Ntgk' => 29, 'Kntgk' => 30, 'Otgk' => 31, 'Szgk' => 32, 'Oszgk' => 33, 'Mkp' => 34, 'Kpf' => 35, 'Lassu' => 36, 'Spec' => 37, 'Fmegb' => 38, 'Adatforr' => 39, 'Szamlnap' => 40, 'Jelleg1' => 41, 'Jelleg2' => 42, 'Fmegj' => 43, ),

self::TYPE\_STUDLYPHPNAME => array('shapenr' => 0, 'partnr' => 1, 'nrparts' => 2, 'pointnr' => 3, 'nrpoints' => 4, 'x' => 5, 'y' => 6, 'measure' => 7, 'kszam' => 8, 'pkod' => 9, 'kkod' => 10, 'vvkod' => 11, 'kszelv' => 12, 'vszelv' => 13, 'rshossz' => 14, 'anf' => 15, 'anet' => 16, 'mof' => 17, 'ongj' => 18, 'oj' => 19, 'omot' => 20, 'ev' => 21, 'asz' => 22, 'buszcs' => 23, 'busze' => 24, 'obusz' => 25, 'nyszer' => 26, 'potktgk' => 27, 'ktgk' => 28, 'ntgk' => 29, 'kntgk' => 30, 'otgk' => 31, 'szgk' => 32, 'oszgk' => 33, 'mkp' => 34, 'kpf' => 35, 'lassu' => 36, 'spec' => 37, 'fmegb' => 38, 'adatforr' => 39, 'szamlnap' => 40, 'jelleg1' => 41, 'jelleg2' => 42, 'fmegj' => 43, ),

self::TYPE\_COLNAME => array(StgKeresztmetszetiForgalomTableMap::COL\_SHAPENR => 0, StgKeresztmetszetiForgalomTableMap::COL\_PARTNR => 1, StgKeresztmetszetiForgalomTableMap::COL\_NRPARTS => 2, StgKeresztmetszetiForgalomTableMap::COL\_POINTNR => 3, StgKeresztmetszetiForgalomTableMap::COL\_NRPOINTS => 4, StgKeresztmetszetiForgalomTableMap::COL\_X => 5, StgKeresztmetszetiForgalomTableMap::COL\_Y => 6, StgKeresztmetszetiForgalomTableMap::COL\_MEASURE => 7, StgKeresztmetszetiForgalomTableMap::COL\_KSZAM => 8, StgKeresztmetszetiForgalomTableMap::COL\_PKOD => 9, StgKeresztmetszetiForgalomTableMap::COL\_KKOD => 10, StgKeresztmetszetiForgalomTableMap::COL\_VVKOD => 11, StgKeresztmetszetiForgalomTableMap::COL\_KSZELV => 12, StgKeresztmetszetiForgalomTableMap::COL\_VSZELV => 13, StgKeresztmetszetiForgalomTableMap::COL\_RSHOSSZ => 14, StgKeresztmetszetiForgalomTableMap::COL\_ANF => 15, StgKeresztmetszetiForgalomTableMap::COL\_ANET => 16, StgKeresztmetszetiForgalomTableMap::COL\_MOF => 17, StgKeresztmetszetiForgalomTableMap::COL\_ONGJ => 18, StgKeresztmetszetiForgalomTableMap::COL\_OJ => 19, StgKeresztmetszetiForgalomTableMap::COL\_OMOT => 20, StgKeresztmetszetiForgalomTableMap::COL\_EV => 21, StgKeresztmetszetiForgalomTableMap::COL\_ASZ => 22, StgKeresztmetszetiForgalomTableMap::COL\_BUSZCS => 23, StgKeresztmetszetiForgalomTableMap::COL\_BUSZE => 24, StgKeresztmetszetiForgalomTableMap::COL\_OBUSZ => 25, StgKeresztmetszetiForgalomTableMap::COL\_NYSZER => 26, StgKeresztmetszetiForgalomTableMap::COL\_POTKTGK => 27, StgKeresztmetszetiForgalomTableMap::COL\_KTGK => 28, StgKeresztmetszetiForgalomTableMap::COL\_NTGK => 29, StgKeresztmetszetiForgalomTableMap::COL\_KNTGK => 30, StgKeresztmetszetiForgalomTableMap::COL\_OTGK => 31, StgKeresztmetszetiForgalomTableMap::COL\_SZGK => 32, StgKeresztmetszetiForgalomTableMap::COL\_OSZGK => 33, StgKeresztmetszetiForgalomTableMap::COL\_MKP => 34, StgKeresztmetszetiForgalomTableMap::COL\_KPF => 35, StgKeresztmetszetiForgalomTableMap::COL\_LASSU => 36, StgKeresztmetszetiForgalomTableMap::COL\_SPEC => 37, StgKeresztmetszetiForgalomTableMap::COL\_FMEGB => 38, StgKeresztmetszetiForgalomTableMap::COL\_ADATFORR => 39, StgKeresztmetszetiForgalomTableMap::COL\_SZAMLNAP => 40, StgKeresztmetszetiForgalomTableMap::COL\_JELLEG1 => 41, StgKeresztmetszetiForgalomTableMap::COL\_JELLEG2 => 42, StgKeresztmetszetiForgalomTableMap::COL\_FMEGJ => 43, ),

self::TYPE\_RAW\_COLNAME => array('COL\_SHAPENR' => 0, 'COL\_PARTNR' => 1, 'COL\_NRPARTS' => 2, 'COL\_POINTNR' => 3, 'COL\_NRPOINTS' => 4, 'COL\_X' => 5, 'COL\_Y' => 6, 'COL\_MEASURE' => 7, 'COL\_KSZAM' => 8, 'COL\_PKOD' => 9, 'COL\_KKOD' => 10, 'COL\_VVKOD' => 11, 'COL\_KSZELV' => 12, 'COL\_VSZELV' => 13, 'COL\_RSHOSSZ' => 14, 'COL\_ANF' => 15, 'COL\_ANET' => 16, 'COL\_MOF' => 17, 'COL\_ONGJ' => 18, 'COL\_OJ' => 19, 'COL\_OMOT' => 20, 'COL\_EV' => 21, 'COL\_ASZ' => 22, 'COL\_BUSZCS' => 23, 'COL\_BUSZE' => 24, 'COL\_OBUSZ' => 25, 'COL\_NYSZER' => 26, 'COL\_POTKTGK' => 27, 'COL\_KTGK' => 28, 'COL\_NTGK' => 29, 'COL\_KNTGK' => 30, 'COL\_OTGK' => 31, 'COL\_SZGK' => 32, 'COL\_OSZGK' => 33, 'COL\_MKP' => 34, 'COL\_KPF' => 35, 'COL\_LASSU' => 36, 'COL\_SPEC' => 37, 'COL\_FMEGB' => 38, 'COL\_ADATFORR' => 39, 'COL\_SZAMLNAP' => 40, 'COL\_JELLEG1' => 41, 'COL\_JELLEG2' => 42, 'COL\_FMEGJ' => 43, ),

self::TYPE\_FIELDNAME => array('shapenr' => 0, 'partnr' => 1, 'nrparts' => 2, 'pointnr' => 3, 'nrpointS' => 4, 'x' => 5, 'y' => 6, 'measure' => 7, 'kszam' => 8, 'pkod' => 9, 'kkod' => 10, 'vvkod' => 11, 'kszelv' => 12, 'vszelv' => 13, 'RSHOSSZ' => 14, 'ANF' => 15, 'ANET' => 16, 'MOF' => 17, 'ONGJ' => 18, 'OJ' => 19, 'OMOT' => 20, 'EV' => 21, 'ASZ' => 22, 'BUSZCS' => 23, 'BUSZE' => 24, 'OBUSZ' => 25, 'NYSZER' => 26, 'POTKTGK' => 27, 'KTGK' => 28, 'NTGK' => 29, 'KNTGK' => 30, 'OTGK' => 31, 'SZGK' => 32, 'OSZGK' => 33, 'MKP' => 34, 'KPF' => 35, 'LASSU' => 36, 'SPEC' => 37, 'FMEGB' => 38, 'ADATFORR' => 39, 'SZAMLNAP' => 40, 'JELLEG1' => 41, 'JELLEG2' => 42, 'FMEGJ' => 43, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_keresztmetszeti\_forgalom');

$this->setPhpName('StgKeresztmetszetiForgalom');

$this->setClassName('\\StgKeresztmetszetiForgalom');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('SHAPENR', 'Shapenr', 'INTEGER', false, null, null);

$this->addColumn('PARTNR', 'Partnr', 'INTEGER', false, null, null);

$this->addColumn('NRPARTS', 'Nrparts', 'INTEGER', false, null, null);

$this->addColumn('POINTNR', 'Pointnr', 'INTEGER', false, null, null);

$this->addColumn('NRPOINTS', 'Nrpoints', 'INTEGER', false, null, null);

$this->addColumn('X', 'X', 'DOUBLE', false, null, null);

$this->addColumn('Y', 'Y', 'DOUBLE', false, null, null);

$this->addColumn('MEASURE', 'Measure', 'DOUBLE', false, null, null);

$this->addColumn('KSZAM', 'Kszam', 'VARCHAR', false, 6, null);

$this->addColumn('PKOD', 'Pkod', 'DOUBLE', false, null, null);

$this->addColumn('KKOD', 'Kkod', 'VARCHAR', false, 15, null);

$this->addColumn('VVKOD', 'Vvkod', 'VARCHAR', false, 15, null);

$this->addColumn('KSZELV', 'Kszelv', 'VARCHAR', false, 15, null);

$this->addColumn('VSZELV', 'Vszelv', 'VARCHAR', false, 15, null);

$this->addColumn('RSHOSSZ', 'Rshossz', 'DOUBLE', false, null, null);

$this->addColumn('ANF', 'Anf', 'SMALLINT', false, 9, null);

$this->addColumn('ANET', 'Anet', 'SMALLINT', false, null, null);

$this->addColumn('MOF', 'Mof', 'SMALLINT', false, null, null);

$this->addColumn('ONGJ', 'Ongj', 'SMALLINT', false, null, null);

$this->addColumn('OJ', 'Oj', 'SMALLINT', false, 9, null);

$this->addColumn('OMOT', 'Omot', 'SMALLINT', false, 9, null);

$this->addColumn('EV', 'Ev', 'SMALLINT', false, null, null);

$this->addColumn('ASZ', 'Asz', 'INTEGER', false, null, null);

$this->addColumn('BUSZCS', 'Buszcs', 'INTEGER', false, null, null);

$this->addColumn('BUSZE', 'Busze', 'SMALLINT', false, null, null);

$this->addColumn('OBUSZ', 'Obusz', 'VARCHAR', false, 45, null);

$this->addColumn('NYSZER', 'Nyszer', 'VARCHAR', false, 45, null);

$this->addColumn('POTKTGK', 'Potktgk', 'VARCHAR', false, 45, null);

$this->addColumn('KTGK', 'Ktgk', 'VARCHAR', false, 45, null);

$this->addColumn('NTGK', 'Ntgk', 'VARCHAR', false, 45, null);

$this->addColumn('KNTGK', 'Kntgk', 'VARCHAR', false, 45, null);

$this->addColumn('OTGK', 'Otgk', 'VARCHAR', false, 45, null);

$this->addColumn('SZGK', 'Szgk', 'VARCHAR', false, 45, null);

$this->addColumn('OSZGK', 'Oszgk', 'VARCHAR', false, 45, null);

$this->addColumn('MKP', 'Mkp', 'VARCHAR', false, 45, null);

$this->addColumn('KPF', 'Kpf', 'VARCHAR', false, 45, null);

$this->addColumn('LASSU', 'Lassu', 'VARCHAR', false, 45, null);

$this->addColumn('SPEC', 'Spec', 'VARCHAR', false, 45, null);

$this->addColumn('FMEGB', 'Fmegb', 'VARCHAR', false, 45, null);

$this->addColumn('ADATFORR', 'Adatforr', 'VARCHAR', false, 45, null);

$this->addColumn('SZAMLNAP', 'Szamlnap', 'VARCHAR', false, 45, null);

$this->addColumn('JELLEG1', 'Jelleg1', 'VARCHAR', false, 45, null);

$this->addColumn('JELLEG2', 'Jelleg2', 'VARCHAR', false, 45, null);

$this->addColumn('FMEGJ', 'Fmegj', 'VARCHAR', false, 45, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgKeresztmetszetiForgalomTableMap::CLASS\_DEFAULT : StgKeresztmetszetiForgalomTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgKeresztmetszetiForgalom object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgKeresztmetszetiForgalomTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgKeresztmetszetiForgalomTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgKeresztmetszetiForgalomTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgKeresztmetszetiForgalomTableMap::OM\_CLASS;

/\*\* @var StgKeresztmetszetiForgalom $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgKeresztmetszetiForgalomTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgKeresztmetszetiForgalomTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgKeresztmetszetiForgalomTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgKeresztmetszetiForgalom $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgKeresztmetszetiForgalomTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_SHAPENR);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_PARTNR);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_NRPARTS);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_POINTNR);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_NRPOINTS);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_X);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_Y);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_MEASURE);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_KSZAM);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_PKOD);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_KKOD);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_VVKOD);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_KSZELV);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_VSZELV);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_RSHOSSZ);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_ANF);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_ANET);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_MOF);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_ONGJ);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_OJ);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_OMOT);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_EV);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_ASZ);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_BUSZCS);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_BUSZE);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_OBUSZ);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_NYSZER);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_POTKTGK);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_KTGK);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_NTGK);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_KNTGK);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_OTGK);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_SZGK);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_OSZGK);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_MKP);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_KPF);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_LASSU);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_SPEC);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_FMEGB);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_ADATFORR);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_SZAMLNAP);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_JELLEG1);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_JELLEG2);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomTableMap::COL\_FMEGJ);

} else {

$criteria->addSelectColumn($alias . '.SHAPENR');

$criteria->addSelectColumn($alias . '.PARTNR');

$criteria->addSelectColumn($alias . '.NRPARTS');

$criteria->addSelectColumn($alias . '.POINTNR');

$criteria->addSelectColumn($alias . '.NRPOINTS');

$criteria->addSelectColumn($alias . '.X');

$criteria->addSelectColumn($alias . '.Y');

$criteria->addSelectColumn($alias . '.MEASURE');

$criteria->addSelectColumn($alias . '.KSZAM');

$criteria->addSelectColumn($alias . '.PKOD');

$criteria->addSelectColumn($alias . '.KKOD');

$criteria->addSelectColumn($alias . '.VVKOD');

$criteria->addSelectColumn($alias . '.KSZELV');

$criteria->addSelectColumn($alias . '.VSZELV');

$criteria->addSelectColumn($alias . '.RSHOSSZ');

$criteria->addSelectColumn($alias . '.ANF');

$criteria->addSelectColumn($alias . '.ANET');

$criteria->addSelectColumn($alias . '.MOF');

$criteria->addSelectColumn($alias . '.ONGJ');

$criteria->addSelectColumn($alias . '.OJ');

$criteria->addSelectColumn($alias . '.OMOT');

$criteria->addSelectColumn($alias . '.EV');

$criteria->addSelectColumn($alias . '.ASZ');

$criteria->addSelectColumn($alias . '.BUSZCS');

$criteria->addSelectColumn($alias . '.BUSZE');

$criteria->addSelectColumn($alias . '.OBUSZ');

$criteria->addSelectColumn($alias . '.NYSZER');

$criteria->addSelectColumn($alias . '.POTKTGK');

$criteria->addSelectColumn($alias . '.KTGK');

$criteria->addSelectColumn($alias . '.NTGK');

$criteria->addSelectColumn($alias . '.KNTGK');

$criteria->addSelectColumn($alias . '.OTGK');

$criteria->addSelectColumn($alias . '.SZGK');

$criteria->addSelectColumn($alias . '.OSZGK');

$criteria->addSelectColumn($alias . '.MKP');

$criteria->addSelectColumn($alias . '.KPF');

$criteria->addSelectColumn($alias . '.LASSU');

$criteria->addSelectColumn($alias . '.SPEC');

$criteria->addSelectColumn($alias . '.FMEGB');

$criteria->addSelectColumn($alias . '.ADATFORR');

$criteria->addSelectColumn($alias . '.SZAMLNAP');

$criteria->addSelectColumn($alias . '.JELLEG1');

$criteria->addSelectColumn($alias . '.JELLEG2');

$criteria->addSelectColumn($alias . '.FMEGJ');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgKeresztmetszetiForgalomTableMap::DATABASE\_NAME)->getTable(StgKeresztmetszetiForgalomTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgKeresztmetszetiForgalomTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgKeresztmetszetiForgalomTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgKeresztmetszetiForgalomTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgKeresztmetszetiForgalom or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgKeresztmetszetiForgalom object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKeresztmetszetiForgalomTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgKeresztmetszetiForgalom) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgKeresztmetszetiForgalom object has no primary key');

}

$query = StgKeresztmetszetiForgalomQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgKeresztmetszetiForgalomTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgKeresztmetszetiForgalomTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_keresztmetszeti\_forgalom table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgKeresztmetszetiForgalomQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgKeresztmetszetiForgalom or Criteria object.

\*

\* @param mixed $criteria Criteria or StgKeresztmetszetiForgalom object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKeresztmetszetiForgalomTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgKeresztmetszetiForgalom object

}

// Set the correct dbName

$query = StgKeresztmetszetiForgalomQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgKeresztmetszetiForgalomTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgKeresztmetszetiForgalomTableMap::buildTableMap();

### StgKeresztmetszetiForgalomV1TableMap.php

<?php

namespace Map;

use \StgKeresztmetszetiForgalomV1;

use \StgKeresztmetszetiForgalomV1Query;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_keresztmetszeti\_forgalom\_v1' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgKeresztmetszetiForgalomV1TableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgKeresztmetszetiForgalomV1TableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_keresztmetszeti\_forgalom\_v1';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgKeresztmetszetiForgalomV1';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgKeresztmetszetiForgalomV1';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 11;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 11;

/\*\*

\* the column name for the PARTNR field

\*/

const COL\_PARTNR = 'stg\_keresztmetszeti\_forgalom\_v1.PARTNR';

/\*\*

\* the column name for the NRPARTS field

\*/

const COL\_NRPARTS = 'stg\_keresztmetszeti\_forgalom\_v1.NRPARTS';

/\*\*

\* the column name for the POINTNR field

\*/

const COL\_POINTNR = 'stg\_keresztmetszeti\_forgalom\_v1.POINTNR';

/\*\*

\* the column name for the NRPOINTS field

\*/

const COL\_NRPOINTS = 'stg\_keresztmetszeti\_forgalom\_v1.NRPOINTS';

/\*\*

\* the column name for the X field

\*/

const COL\_X = 'stg\_keresztmetszeti\_forgalom\_v1.X';

/\*\*

\* the column name for the Y field

\*/

const COL\_Y = 'stg\_keresztmetszeti\_forgalom\_v1.Y';

/\*\*

\* the column name for the MEASURE field

\*/

const COL\_MEASURE = 'stg\_keresztmetszeti\_forgalom\_v1.MEASURE';

/\*\*

\* the column name for the KEZD field

\*/

const COL\_KEZD = 'stg\_keresztmetszeti\_forgalom\_v1.KEZD';

/\*\*

\* the column name for the VEGE field

\*/

const COL\_VEGE = 'stg\_keresztmetszeti\_forgalom\_v1.VEGE';

/\*\*

\* the column name for the KSZAM field

\*/

const COL\_KSZAM = 'stg\_keresztmetszeti\_forgalom\_v1.KSZAM';

/\*\*

\* the column name for the PKOD field

\*/

const COL\_PKOD = 'stg\_keresztmetszeti\_forgalom\_v1.PKOD';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('Partnr', 'Nrparts', 'Pointnr', 'Nrpoints', 'X', 'Y', 'Measure', 'Kezd', 'Vege', 'Kszam', 'Pkod', ),

self::TYPE\_STUDLYPHPNAME => array('partnr', 'nrparts', 'pointnr', 'nrpoints', 'x', 'y', 'measure', 'kezd', 'vege', 'kszam', 'pkod', ),

self::TYPE\_COLNAME => array(StgKeresztmetszetiForgalomV1TableMap::COL\_PARTNR, StgKeresztmetszetiForgalomV1TableMap::COL\_NRPARTS, StgKeresztmetszetiForgalomV1TableMap::COL\_POINTNR, StgKeresztmetszetiForgalomV1TableMap::COL\_NRPOINTS, StgKeresztmetszetiForgalomV1TableMap::COL\_X, StgKeresztmetszetiForgalomV1TableMap::COL\_Y, StgKeresztmetszetiForgalomV1TableMap::COL\_MEASURE, StgKeresztmetszetiForgalomV1TableMap::COL\_KEZD, StgKeresztmetszetiForgalomV1TableMap::COL\_VEGE, StgKeresztmetszetiForgalomV1TableMap::COL\_KSZAM, StgKeresztmetszetiForgalomV1TableMap::COL\_PKOD, ),

self::TYPE\_RAW\_COLNAME => array('COL\_PARTNR', 'COL\_NRPARTS', 'COL\_POINTNR', 'COL\_NRPOINTS', 'COL\_X', 'COL\_Y', 'COL\_MEASURE', 'COL\_KEZD', 'COL\_VEGE', 'COL\_KSZAM', 'COL\_PKOD', ),

self::TYPE\_FIELDNAME => array('partnr', 'nrparts', 'pointnr', 'nrpointS', 'x', 'y', 'measure', 'kezd', 'vege', 'kszam', 'pkod', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('Partnr' => 0, 'Nrparts' => 1, 'Pointnr' => 2, 'Nrpoints' => 3, 'X' => 4, 'Y' => 5, 'Measure' => 6, 'Kezd' => 7, 'Vege' => 8, 'Kszam' => 9, 'Pkod' => 10, ),

self::TYPE\_STUDLYPHPNAME => array('partnr' => 0, 'nrparts' => 1, 'pointnr' => 2, 'nrpoints' => 3, 'x' => 4, 'y' => 5, 'measure' => 6, 'kezd' => 7, 'vege' => 8, 'kszam' => 9, 'pkod' => 10, ),

self::TYPE\_COLNAME => array(StgKeresztmetszetiForgalomV1TableMap::COL\_PARTNR => 0, StgKeresztmetszetiForgalomV1TableMap::COL\_NRPARTS => 1, StgKeresztmetszetiForgalomV1TableMap::COL\_POINTNR => 2, StgKeresztmetszetiForgalomV1TableMap::COL\_NRPOINTS => 3, StgKeresztmetszetiForgalomV1TableMap::COL\_X => 4, StgKeresztmetszetiForgalomV1TableMap::COL\_Y => 5, StgKeresztmetszetiForgalomV1TableMap::COL\_MEASURE => 6, StgKeresztmetszetiForgalomV1TableMap::COL\_KEZD => 7, StgKeresztmetszetiForgalomV1TableMap::COL\_VEGE => 8, StgKeresztmetszetiForgalomV1TableMap::COL\_KSZAM => 9, StgKeresztmetszetiForgalomV1TableMap::COL\_PKOD => 10, ),

self::TYPE\_RAW\_COLNAME => array('COL\_PARTNR' => 0, 'COL\_NRPARTS' => 1, 'COL\_POINTNR' => 2, 'COL\_NRPOINTS' => 3, 'COL\_X' => 4, 'COL\_Y' => 5, 'COL\_MEASURE' => 6, 'COL\_KEZD' => 7, 'COL\_VEGE' => 8, 'COL\_KSZAM' => 9, 'COL\_PKOD' => 10, ),

self::TYPE\_FIELDNAME => array('partnr' => 0, 'nrparts' => 1, 'pointnr' => 2, 'nrpointS' => 3, 'x' => 4, 'y' => 5, 'measure' => 6, 'kezd' => 7, 'vege' => 8, 'kszam' => 9, 'pkod' => 10, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_keresztmetszeti\_forgalom\_v1');

$this->setPhpName('StgKeresztmetszetiForgalomV1');

$this->setClassName('\\StgKeresztmetszetiForgalomV1');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addPrimaryKey('PARTNR', 'Partnr', 'INTEGER', true, null, null);

$this->addColumn('NRPARTS', 'Nrparts', 'VARCHAR', false, 45, null);

$this->addColumn('POINTNR', 'Pointnr', 'VARCHAR', false, 45, null);

$this->addColumn('NRPOINTS', 'Nrpoints', 'VARCHAR', false, 45, null);

$this->addColumn('X', 'X', 'VARCHAR', false, 45, null);

$this->addColumn('Y', 'Y', 'VARCHAR', false, 45, null);

$this->addColumn('MEASURE', 'Measure', 'VARCHAR', false, 45, null);

$this->addColumn('KEZD', 'Kezd', 'VARCHAR', false, 45, null);

$this->addColumn('VEGE', 'Vege', 'VARCHAR', false, 45, null);

$this->addColumn('KSZAM', 'Kszam', 'VARCHAR', false, 45, null);

$this->addColumn('PKOD', 'Pkod', 'VARCHAR', false, 45, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

// If the PK cannot be derived from the row, return NULL.

if ($row[TableMap::TYPE\_NUM == $indexType ? 0 + $offset : static::translateFieldName('Partnr', TableMap::TYPE\_PHPNAME, $indexType)] === null) {

return null;

}

return (string) $row[TableMap::TYPE\_NUM == $indexType ? 0 + $offset : static::translateFieldName('Partnr', TableMap::TYPE\_PHPNAME, $indexType)];

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return (int) $row[

$indexType == TableMap::TYPE\_NUM

? 0 + $offset

: self::translateFieldName('Partnr', TableMap::TYPE\_PHPNAME, $indexType)

];

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgKeresztmetszetiForgalomV1TableMap::CLASS\_DEFAULT : StgKeresztmetszetiForgalomV1TableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgKeresztmetszetiForgalomV1 object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgKeresztmetszetiForgalomV1TableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgKeresztmetszetiForgalomV1TableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgKeresztmetszetiForgalomV1TableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgKeresztmetszetiForgalomV1TableMap::OM\_CLASS;

/\*\* @var StgKeresztmetszetiForgalomV1 $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgKeresztmetszetiForgalomV1TableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgKeresztmetszetiForgalomV1TableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgKeresztmetszetiForgalomV1TableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgKeresztmetszetiForgalomV1 $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgKeresztmetszetiForgalomV1TableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgKeresztmetszetiForgalomV1TableMap::COL\_PARTNR);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomV1TableMap::COL\_NRPARTS);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomV1TableMap::COL\_POINTNR);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomV1TableMap::COL\_NRPOINTS);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomV1TableMap::COL\_X);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomV1TableMap::COL\_Y);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomV1TableMap::COL\_MEASURE);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomV1TableMap::COL\_KEZD);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomV1TableMap::COL\_VEGE);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomV1TableMap::COL\_KSZAM);

$criteria->addSelectColumn(StgKeresztmetszetiForgalomV1TableMap::COL\_PKOD);

} else {

$criteria->addSelectColumn($alias . '.PARTNR');

$criteria->addSelectColumn($alias . '.NRPARTS');

$criteria->addSelectColumn($alias . '.POINTNR');

$criteria->addSelectColumn($alias . '.NRPOINTS');

$criteria->addSelectColumn($alias . '.X');

$criteria->addSelectColumn($alias . '.Y');

$criteria->addSelectColumn($alias . '.MEASURE');

$criteria->addSelectColumn($alias . '.KEZD');

$criteria->addSelectColumn($alias . '.VEGE');

$criteria->addSelectColumn($alias . '.KSZAM');

$criteria->addSelectColumn($alias . '.PKOD');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgKeresztmetszetiForgalomV1TableMap::DATABASE\_NAME)->getTable(StgKeresztmetszetiForgalomV1TableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgKeresztmetszetiForgalomV1TableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgKeresztmetszetiForgalomV1TableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgKeresztmetszetiForgalomV1TableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgKeresztmetszetiForgalomV1 or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgKeresztmetszetiForgalomV1 object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKeresztmetszetiForgalomV1TableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgKeresztmetszetiForgalomV1) { // it's a model object

// create criteria based on pk values

$criteria = $values->buildPkeyCriteria();

} else { // it's a primary key, or an array of pks

$criteria = new Criteria(StgKeresztmetszetiForgalomV1TableMap::DATABASE\_NAME);

$criteria->add(StgKeresztmetszetiForgalomV1TableMap::COL\_PARTNR, (array) $values, Criteria::IN);

}

$query = StgKeresztmetszetiForgalomV1Query::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgKeresztmetszetiForgalomV1TableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgKeresztmetszetiForgalomV1TableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_keresztmetszeti\_forgalom\_v1 table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgKeresztmetszetiForgalomV1Query::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgKeresztmetszetiForgalomV1 or Criteria object.

\*

\* @param mixed $criteria Criteria or StgKeresztmetszetiForgalomV1 object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKeresztmetszetiForgalomV1TableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgKeresztmetszetiForgalomV1 object

}

// Set the correct dbName

$query = StgKeresztmetszetiForgalomV1Query::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgKeresztmetszetiForgalomV1TableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgKeresztmetszetiForgalomV1TableMap::buildTableMap();

### StgKollegiumTableMap.php

<?php

namespace Map;

use \StgKollegium;

use \StgKollegiumQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_kollegium' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgKollegiumTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgKollegiumTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_kollegium';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgKollegium';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgKollegium';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 6;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 6;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_kollegium.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_kollegium.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA field

\*/

const COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA = 'stg\_kollegium.KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA';

/\*\*

\* the column name for the KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA field

\*/

const COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA = 'stg\_kollegium.KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA';

/\*\*

\* the column name for the KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA field

\*/

const COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA = 'stg\_kollegium.KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA';

/\*\*

\* the column name for the KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA field

\*/

const COL\_KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA = 'stg\_kollegium.KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'KollegiumbanLakoNappaliAltTanulokSzama', 'KollegiumbanLakoNappaliKozepiskolaiTanulokSzama', 'KollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama', 'KollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'kollegiumbanLakoNappaliAltTanulokSzama', 'kollegiumbanLakoNappaliKozepiskolaiTanulokSzama', 'kollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama', 'kollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama', ),

self::TYPE\_COLNAME => array(StgKollegiumTableMap::COL\_TELEPULES\_NEV, StgKollegiumTableMap::COL\_TELEPULES\_KSHKOD, StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA, StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA, StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA', 'COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA', 'COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA', 'COL\_KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'kollegiumban\_lako\_nappali\_alt\_tanulok\_szama', 'kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama', 'kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama', 'kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'KollegiumbanLakoNappaliAltTanulokSzama' => 2, 'KollegiumbanLakoNappaliKozepiskolaiTanulokSzama' => 3, 'KollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama' => 4, 'KollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama' => 5, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'kollegiumbanLakoNappaliAltTanulokSzama' => 2, 'kollegiumbanLakoNappaliKozepiskolaiTanulokSzama' => 3, 'kollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama' => 4, 'kollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama' => 5, ),

self::TYPE\_COLNAME => array(StgKollegiumTableMap::COL\_TELEPULES\_NEV => 0, StgKollegiumTableMap::COL\_TELEPULES\_KSHKOD => 1, StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA => 2, StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA => 3, StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA => 4, StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA => 5, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA' => 2, 'COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA' => 3, 'COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA' => 4, 'COL\_KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA' => 5, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'kollegiumban\_lako\_nappali\_alt\_tanulok\_szama' => 2, 'kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama' => 3, 'kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama' => 4, 'kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama' => 5, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_kollegium');

$this->setPhpName('StgKollegium');

$this->setClassName('\\StgKollegium');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA', 'KollegiumbanLakoNappaliAltTanulokSzama', 'INTEGER', false, null, null);

$this->addColumn('KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA', 'KollegiumbanLakoNappaliKozepiskolaiTanulokSzama', 'INTEGER', false, null, null);

$this->addColumn('KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA', 'KollegiumbanLakoNappaliSzakiskolaiSpecTanulokSzama', 'INTEGER', false, null, null);

$this->addColumn('KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA', 'KollegiumbanLakoFelsofokuAlapMesterkepzesbenTanulokSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgKollegiumTableMap::CLASS\_DEFAULT : StgKollegiumTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgKollegium object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgKollegiumTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgKollegiumTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgKollegiumTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgKollegiumTableMap::OM\_CLASS;

/\*\* @var StgKollegium $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgKollegiumTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgKollegiumTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgKollegiumTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgKollegium $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgKollegiumTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgKollegiumTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgKollegiumTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA);

$criteria->addSelectColumn(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA);

$criteria->addSelectColumn(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA);

$criteria->addSelectColumn(StgKollegiumTableMap::COL\_KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.KOLLEGIUMBAN\_LAKO\_NAPPALI\_ALT\_TANULOK\_SZAMA');

$criteria->addSelectColumn($alias . '.KOLLEGIUMBAN\_LAKO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA');

$criteria->addSelectColumn($alias . '.KOLLEGIUMBAN\_LAKO\_NAPPALI\_SZAKISKOLAI\_SPEC\_TANULOK\_SZAMA');

$criteria->addSelectColumn($alias . '.KOLLEGIUMBAN\_LAKO\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_TANULOK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgKollegiumTableMap::DATABASE\_NAME)->getTable(StgKollegiumTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgKollegiumTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgKollegiumTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgKollegiumTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgKollegium or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgKollegium object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKollegiumTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgKollegium) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgKollegium object has no primary key');

}

$query = StgKollegiumQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgKollegiumTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgKollegiumTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_kollegium table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgKollegiumQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgKollegium or Criteria object.

\*

\* @param mixed $criteria Criteria or StgKollegium object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKollegiumTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgKollegium object

}

// Set the correct dbName

$query = StgKollegiumQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgKollegiumTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgKollegiumTableMap::buildTableMap();

### StgKulturaEsSportTableMap.php

<?php

namespace Map;

use \StgKulturaEsSport;

use \StgKulturaEsSportQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_kultura\_es\_sport' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgKulturaEsSportTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgKulturaEsSportTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_kultura\_es\_sport';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgKulturaEsSport';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgKulturaEsSport';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 10;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 10;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_kultura\_es\_sport.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_kultura\_es\_sport.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the FILMSZINHAZ field

\*/

const COL\_FILMSZINHAZ = 'stg\_kultura\_es\_sport.FILMSZINHAZ';

/\*\*

\* the column name for the KOZMUVELODESI\_INTEZMENY field

\*/

const COL\_KOZMUVELODESI\_INTEZMENY = 'stg\_kultura\_es\_sport.KOZMUVELODESI\_INTEZMENY';

/\*\*

\* the column name for the MUZEUM field

\*/

const COL\_MUZEUM = 'stg\_kultura\_es\_sport.MUZEUM';

/\*\*

\* the column name for the PIAC field

\*/

const COL\_PIAC = 'stg\_kultura\_es\_sport.PIAC';

/\*\*

\* the column name for the SPORTCSARNOK\_SPORTPALYA field

\*/

const COL\_SPORTCSARNOK\_SPORTPALYA = 'stg\_kultura\_es\_sport.SPORTCSARNOK\_SPORTPALYA';

/\*\*

\* the column name for the STRAND field

\*/

const COL\_STRAND = 'stg\_kultura\_es\_sport.STRAND';

/\*\*

\* the column name for the TELEPULESI\_KONYVTAR field

\*/

const COL\_TELEPULESI\_KONYVTAR = 'stg\_kultura\_es\_sport.TELEPULESI\_KONYVTAR';

/\*\*

\* the column name for the USZODA\_FURDO\_GYOGYFURDO field

\*/

const COL\_USZODA\_FURDO\_GYOGYFURDO = 'stg\_kultura\_es\_sport.USZODA\_FURDO\_GYOGYFURDO';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'Filmszinhaz', 'KozmuvelodesiIntezmeny', 'Muzeum', 'Piac', 'SportcsarnokSportpalya', 'Strand', 'TelepulesiKonyvtar', 'UszodaFurdoGyogyfurdo', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'filmszinhaz', 'kozmuvelodesiIntezmeny', 'muzeum', 'piac', 'sportcsarnokSportpalya', 'strand', 'telepulesiKonyvtar', 'uszodaFurdoGyogyfurdo', ),

self::TYPE\_COLNAME => array(StgKulturaEsSportTableMap::COL\_TELEPULES\_NEV, StgKulturaEsSportTableMap::COL\_TELEPULES\_KSHKOD, StgKulturaEsSportTableMap::COL\_FILMSZINHAZ, StgKulturaEsSportTableMap::COL\_KOZMUVELODESI\_INTEZMENY, StgKulturaEsSportTableMap::COL\_MUZEUM, StgKulturaEsSportTableMap::COL\_PIAC, StgKulturaEsSportTableMap::COL\_SPORTCSARNOK\_SPORTPALYA, StgKulturaEsSportTableMap::COL\_STRAND, StgKulturaEsSportTableMap::COL\_TELEPULESI\_KONYVTAR, StgKulturaEsSportTableMap::COL\_USZODA\_FURDO\_GYOGYFURDO, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_FILMSZINHAZ', 'COL\_KOZMUVELODESI\_INTEZMENY', 'COL\_MUZEUM', 'COL\_PIAC', 'COL\_SPORTCSARNOK\_SPORTPALYA', 'COL\_STRAND', 'COL\_TELEPULESI\_KONYVTAR', 'COL\_USZODA\_FURDO\_GYOGYFURDO', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'filmszinhaz', 'kozmuvelodesi\_intezmeny', 'muzeum', 'piac', 'sportcsarnok\_sportpalya', 'strand', 'telepulesi\_konyvtar', 'uszoda\_furdo\_gyogyfurdo', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'Filmszinhaz' => 2, 'KozmuvelodesiIntezmeny' => 3, 'Muzeum' => 4, 'Piac' => 5, 'SportcsarnokSportpalya' => 6, 'Strand' => 7, 'TelepulesiKonyvtar' => 8, 'UszodaFurdoGyogyfurdo' => 9, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'filmszinhaz' => 2, 'kozmuvelodesiIntezmeny' => 3, 'muzeum' => 4, 'piac' => 5, 'sportcsarnokSportpalya' => 6, 'strand' => 7, 'telepulesiKonyvtar' => 8, 'uszodaFurdoGyogyfurdo' => 9, ),

self::TYPE\_COLNAME => array(StgKulturaEsSportTableMap::COL\_TELEPULES\_NEV => 0, StgKulturaEsSportTableMap::COL\_TELEPULES\_KSHKOD => 1, StgKulturaEsSportTableMap::COL\_FILMSZINHAZ => 2, StgKulturaEsSportTableMap::COL\_KOZMUVELODESI\_INTEZMENY => 3, StgKulturaEsSportTableMap::COL\_MUZEUM => 4, StgKulturaEsSportTableMap::COL\_PIAC => 5, StgKulturaEsSportTableMap::COL\_SPORTCSARNOK\_SPORTPALYA => 6, StgKulturaEsSportTableMap::COL\_STRAND => 7, StgKulturaEsSportTableMap::COL\_TELEPULESI\_KONYVTAR => 8, StgKulturaEsSportTableMap::COL\_USZODA\_FURDO\_GYOGYFURDO => 9, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_FILMSZINHAZ' => 2, 'COL\_KOZMUVELODESI\_INTEZMENY' => 3, 'COL\_MUZEUM' => 4, 'COL\_PIAC' => 5, 'COL\_SPORTCSARNOK\_SPORTPALYA' => 6, 'COL\_STRAND' => 7, 'COL\_TELEPULESI\_KONYVTAR' => 8, 'COL\_USZODA\_FURDO\_GYOGYFURDO' => 9, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'filmszinhaz' => 2, 'kozmuvelodesi\_intezmeny' => 3, 'muzeum' => 4, 'piac' => 5, 'sportcsarnok\_sportpalya' => 6, 'strand' => 7, 'telepulesi\_konyvtar' => 8, 'uszoda\_furdo\_gyogyfurdo' => 9, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_kultura\_es\_sport');

$this->setPhpName('StgKulturaEsSport');

$this->setClassName('\\StgKulturaEsSport');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('FILMSZINHAZ', 'Filmszinhaz', 'BOOLEAN', false, 1, null);

$this->addColumn('KOZMUVELODESI\_INTEZMENY', 'KozmuvelodesiIntezmeny', 'BOOLEAN', false, 1, null);

$this->addColumn('MUZEUM', 'Muzeum', 'BOOLEAN', false, 1, null);

$this->addColumn('PIAC', 'Piac', 'BOOLEAN', false, 1, null);

$this->addColumn('SPORTCSARNOK\_SPORTPALYA', 'SportcsarnokSportpalya', 'BOOLEAN', false, 1, null);

$this->addColumn('STRAND', 'Strand', 'BOOLEAN', false, 1, null);

$this->addColumn('TELEPULESI\_KONYVTAR', 'TelepulesiKonyvtar', 'BOOLEAN', false, 1, null);

$this->addColumn('USZODA\_FURDO\_GYOGYFURDO', 'UszodaFurdoGyogyfurdo', 'BOOLEAN', false, 1, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgKulturaEsSportTableMap::CLASS\_DEFAULT : StgKulturaEsSportTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgKulturaEsSport object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgKulturaEsSportTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgKulturaEsSportTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgKulturaEsSportTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgKulturaEsSportTableMap::OM\_CLASS;

/\*\* @var StgKulturaEsSport $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgKulturaEsSportTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgKulturaEsSportTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgKulturaEsSportTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgKulturaEsSport $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgKulturaEsSportTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgKulturaEsSportTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgKulturaEsSportTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgKulturaEsSportTableMap::COL\_FILMSZINHAZ);

$criteria->addSelectColumn(StgKulturaEsSportTableMap::COL\_KOZMUVELODESI\_INTEZMENY);

$criteria->addSelectColumn(StgKulturaEsSportTableMap::COL\_MUZEUM);

$criteria->addSelectColumn(StgKulturaEsSportTableMap::COL\_PIAC);

$criteria->addSelectColumn(StgKulturaEsSportTableMap::COL\_SPORTCSARNOK\_SPORTPALYA);

$criteria->addSelectColumn(StgKulturaEsSportTableMap::COL\_STRAND);

$criteria->addSelectColumn(StgKulturaEsSportTableMap::COL\_TELEPULESI\_KONYVTAR);

$criteria->addSelectColumn(StgKulturaEsSportTableMap::COL\_USZODA\_FURDO\_GYOGYFURDO);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.FILMSZINHAZ');

$criteria->addSelectColumn($alias . '.KOZMUVELODESI\_INTEZMENY');

$criteria->addSelectColumn($alias . '.MUZEUM');

$criteria->addSelectColumn($alias . '.PIAC');

$criteria->addSelectColumn($alias . '.SPORTCSARNOK\_SPORTPALYA');

$criteria->addSelectColumn($alias . '.STRAND');

$criteria->addSelectColumn($alias . '.TELEPULESI\_KONYVTAR');

$criteria->addSelectColumn($alias . '.USZODA\_FURDO\_GYOGYFURDO');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgKulturaEsSportTableMap::DATABASE\_NAME)->getTable(StgKulturaEsSportTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgKulturaEsSportTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgKulturaEsSportTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgKulturaEsSportTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgKulturaEsSport or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgKulturaEsSport object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKulturaEsSportTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgKulturaEsSport) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgKulturaEsSport object has no primary key');

}

$query = StgKulturaEsSportQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgKulturaEsSportTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgKulturaEsSportTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_kultura\_es\_sport table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgKulturaEsSportQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgKulturaEsSport or Criteria object.

\*

\* @param mixed $criteria Criteria or StgKulturaEsSport object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKulturaEsSportTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgKulturaEsSport object

}

// Set the correct dbName

$query = StgKulturaEsSportQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgKulturaEsSportTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgKulturaEsSportTableMap::buildTableMap();

### StgKulturalisFunkciokAranyaTableMap.php

<?php

namespace Map;

use \StgKulturalisFunkciokAranya;

use \StgKulturalisFunkciokAranyaQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_kulturalis\_funkciok\_aranya' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgKulturalisFunkciokAranyaTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgKulturalisFunkciokAranyaTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_kulturalis\_funkciok\_aranya';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgKulturalisFunkciokAranya';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgKulturalisFunkciokAranya';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 5;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 5;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_kulturalis\_funkciok\_aranya.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_kulturalis\_funkciok\_aranya.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA field

\*/

const COL\_KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA = 'stg\_kulturalis\_funkciok\_aranya.KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA';

/\*\*

\* the column name for the MOZILATOGATASOK\_SZAMA field

\*/

const COL\_MOZILATOGATASOK\_SZAMA = 'stg\_kulturalis\_funkciok\_aranya.MOZILATOGATASOK\_SZAMA';

/\*\*

\* the column name for the MUZEUMI\_LATOGATOK\_SZAMA field

\*/

const COL\_MUZEUMI\_LATOGATOK\_SZAMA = 'stg\_kulturalis\_funkciok\_aranya.MUZEUMI\_LATOGATOK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'KulturalisRendezvenyekenResztvevokSzama', 'MozilatogatasokSzama', 'MuzeumiLatogatokSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'kulturalisRendezvenyekenResztvevokSzama', 'mozilatogatasokSzama', 'muzeumiLatogatokSzama', ),

self::TYPE\_COLNAME => array(StgKulturalisFunkciokAranyaTableMap::COL\_TELEPULES\_NEV, StgKulturalisFunkciokAranyaTableMap::COL\_TELEPULES\_KSHKOD, StgKulturalisFunkciokAranyaTableMap::COL\_KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA, StgKulturalisFunkciokAranyaTableMap::COL\_MOZILATOGATASOK\_SZAMA, StgKulturalisFunkciokAranyaTableMap::COL\_MUZEUMI\_LATOGATOK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA', 'COL\_MOZILATOGATASOK\_SZAMA', 'COL\_MUZEUMI\_LATOGATOK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'kulturalis\_rendezvenyeken\_resztvevok\_szama', 'mozilatogatasok\_szama', 'muzeumi\_latogatok\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'KulturalisRendezvenyekenResztvevokSzama' => 2, 'MozilatogatasokSzama' => 3, 'MuzeumiLatogatokSzama' => 4, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'kulturalisRendezvenyekenResztvevokSzama' => 2, 'mozilatogatasokSzama' => 3, 'muzeumiLatogatokSzama' => 4, ),

self::TYPE\_COLNAME => array(StgKulturalisFunkciokAranyaTableMap::COL\_TELEPULES\_NEV => 0, StgKulturalisFunkciokAranyaTableMap::COL\_TELEPULES\_KSHKOD => 1, StgKulturalisFunkciokAranyaTableMap::COL\_KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA => 2, StgKulturalisFunkciokAranyaTableMap::COL\_MOZILATOGATASOK\_SZAMA => 3, StgKulturalisFunkciokAranyaTableMap::COL\_MUZEUMI\_LATOGATOK\_SZAMA => 4, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA' => 2, 'COL\_MOZILATOGATASOK\_SZAMA' => 3, 'COL\_MUZEUMI\_LATOGATOK\_SZAMA' => 4, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'kulturalis\_rendezvenyeken\_resztvevok\_szama' => 2, 'mozilatogatasok\_szama' => 3, 'muzeumi\_latogatok\_szama' => 4, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_kulturalis\_funkciok\_aranya');

$this->setPhpName('StgKulturalisFunkciokAranya');

$this->setClassName('\\StgKulturalisFunkciokAranya');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA', 'KulturalisRendezvenyekenResztvevokSzama', 'INTEGER', false, null, null);

$this->addColumn('MOZILATOGATASOK\_SZAMA', 'MozilatogatasokSzama', 'INTEGER', false, null, null);

$this->addColumn('MUZEUMI\_LATOGATOK\_SZAMA', 'MuzeumiLatogatokSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgKulturalisFunkciokAranyaTableMap::CLASS\_DEFAULT : StgKulturalisFunkciokAranyaTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgKulturalisFunkciokAranya object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgKulturalisFunkciokAranyaTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgKulturalisFunkciokAranyaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgKulturalisFunkciokAranyaTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgKulturalisFunkciokAranyaTableMap::OM\_CLASS;

/\*\* @var StgKulturalisFunkciokAranya $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgKulturalisFunkciokAranyaTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgKulturalisFunkciokAranyaTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgKulturalisFunkciokAranyaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgKulturalisFunkciokAranya $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgKulturalisFunkciokAranyaTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgKulturalisFunkciokAranyaTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgKulturalisFunkciokAranyaTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgKulturalisFunkciokAranyaTableMap::COL\_KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA);

$criteria->addSelectColumn(StgKulturalisFunkciokAranyaTableMap::COL\_MOZILATOGATASOK\_SZAMA);

$criteria->addSelectColumn(StgKulturalisFunkciokAranyaTableMap::COL\_MUZEUMI\_LATOGATOK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.KULTURALIS\_RENDEZVENYEKEN\_RESZTVEVOK\_SZAMA');

$criteria->addSelectColumn($alias . '.MOZILATOGATASOK\_SZAMA');

$criteria->addSelectColumn($alias . '.MUZEUMI\_LATOGATOK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgKulturalisFunkciokAranyaTableMap::DATABASE\_NAME)->getTable(StgKulturalisFunkciokAranyaTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgKulturalisFunkciokAranyaTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgKulturalisFunkciokAranyaTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgKulturalisFunkciokAranyaTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgKulturalisFunkciokAranya or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgKulturalisFunkciokAranya object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKulturalisFunkciokAranyaTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgKulturalisFunkciokAranya) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgKulturalisFunkciokAranya object has no primary key');

}

$query = StgKulturalisFunkciokAranyaQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgKulturalisFunkciokAranyaTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgKulturalisFunkciokAranyaTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_kulturalis\_funkciok\_aranya table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgKulturalisFunkciokAranyaQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgKulturalisFunkciokAranya or Criteria object.

\*

\* @param mixed $criteria Criteria or StgKulturalisFunkciokAranya object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgKulturalisFunkciokAranyaTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgKulturalisFunkciokAranya object

}

// Set the correct dbName

$query = StgKulturalisFunkciokAranyaQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgKulturalisFunkciokAranyaTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgKulturalisFunkciokAranyaTableMap::buildTableMap();

### StgLakonepessegTableMap.php

<?php

namespace Map;

use \StgLakonepesseg;

use \StgLakonepessegQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_lakonepesseg' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgLakonepessegTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgLakonepessegTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_lakonepesseg';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgLakonepesseg';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgLakonepesseg';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 3;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 3;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_lakonepesseg.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_lakonepesseg.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the NEPESSEG field

\*/

const COL\_NEPESSEG = 'stg\_lakonepesseg.NEPESSEG';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'Nepesseg', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'nepesseg', ),

self::TYPE\_COLNAME => array(StgLakonepessegTableMap::COL\_TELEPULES\_NEV, StgLakonepessegTableMap::COL\_TELEPULES\_KSHKOD, StgLakonepessegTableMap::COL\_NEPESSEG, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_NEPESSEG', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'nepesseg', ),

self::TYPE\_NUM => array(0, 1, 2, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'Nepesseg' => 2, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'nepesseg' => 2, ),

self::TYPE\_COLNAME => array(StgLakonepessegTableMap::COL\_TELEPULES\_NEV => 0, StgLakonepessegTableMap::COL\_TELEPULES\_KSHKOD => 1, StgLakonepessegTableMap::COL\_NEPESSEG => 2, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_NEPESSEG' => 2, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'nepesseg' => 2, ),

self::TYPE\_NUM => array(0, 1, 2, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_lakonepesseg');

$this->setPhpName('StgLakonepesseg');

$this->setClassName('\\StgLakonepesseg');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('NEPESSEG', 'Nepesseg', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgLakonepessegTableMap::CLASS\_DEFAULT : StgLakonepessegTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgLakonepesseg object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgLakonepessegTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgLakonepessegTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgLakonepessegTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgLakonepessegTableMap::OM\_CLASS;

/\*\* @var StgLakonepesseg $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgLakonepessegTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgLakonepessegTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgLakonepessegTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgLakonepesseg $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgLakonepessegTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgLakonepessegTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgLakonepessegTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgLakonepessegTableMap::COL\_NEPESSEG);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.NEPESSEG');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgLakonepessegTableMap::DATABASE\_NAME)->getTable(StgLakonepessegTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgLakonepessegTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgLakonepessegTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgLakonepessegTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgLakonepesseg or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgLakonepesseg object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgLakonepessegTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgLakonepesseg) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgLakonepesseg object has no primary key');

}

$query = StgLakonepessegQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgLakonepessegTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgLakonepessegTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_lakonepesseg table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgLakonepessegQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgLakonepesseg or Criteria object.

\*

\* @param mixed $criteria Criteria or StgLakonepesseg object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgLakonepessegTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgLakonepesseg object

}

// Set the correct dbName

$query = StgLakonepessegQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgLakonepessegTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgLakonepessegTableMap::buildTableMap();

### StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap.php

<?php

namespace Map;

use \StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok;

use \StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_mas\_telepulesrol\_bejaro\_altalanos\_es\_kozepiskolasok' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_mas\_telepulesrol\_bejaro\_altalanos\_es\_kozepiskolasok';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 4;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 4;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_mas\_telepulesrol\_bejaro\_altalanos\_es\_kozepiskolasok.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_mas\_telepulesrol\_bejaro\_altalanos\_es\_kozepiskolasok.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA field

\*/

const COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA = 'stg\_mas\_telepulesrol\_bejaro\_altalanos\_es\_kozepiskolasok.MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA';

/\*\*

\* the column name for the MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA field

\*/

const COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA = 'stg\_mas\_telepulesrol\_bejaro\_altalanos\_es\_kozepiskolasok.MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'MasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama', 'MasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama', 'masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama', ),

self::TYPE\_COLNAME => array(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_TELEPULES\_NEV, StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_TELEPULES\_KSHKOD, StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA, StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA', 'COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama', 'mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'MasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama' => 2, 'MasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama' => 3, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama' => 2, 'masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama' => 3, ),

self::TYPE\_COLNAME => array(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_TELEPULES\_NEV => 0, StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_TELEPULES\_KSHKOD => 1, StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA => 2, StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA => 3, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA' => 2, 'COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA' => 3, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama' => 2, 'mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama' => 3, ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_mas\_telepulesrol\_bejaro\_altalanos\_es\_kozepiskolasok');

$this->setPhpName('StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok');

$this->setClassName('\\StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA', 'MasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama', 'INTEGER', false, null, null);

$this->addColumn('MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA', 'MasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::CLASS\_DEFAULT : StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::OM\_CLASS;

/\*\* @var StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA);

$criteria->addSelectColumn(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA');

$criteria->addSelectColumn($alias . '.MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::DATABASE\_NAME)->getTable(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok object has no primary key');

}

$query = StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_mas\_telepulesrol\_bejaro\_altalanos\_es\_kozepiskolasok table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok or Criteria object.

\*

\* @param mixed $criteria Criteria or StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgMasTelepulesrolBejaroAltalanosEsKozepiskolasok object

}

// Set the correct dbName

$query = StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgMasTelepulesrolBejaroAltalanosEsKozepiskolasokTableMap::buildTableMap();

### StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap.php

<?php

namespace Map;

use \StgMukodoVallalkozasokSzamaAKiemeltIparagakban;

use \StgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgMukodoVallalkozasokSzamaAKiemeltIparagakban';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgMukodoVallalkozasokSzamaAKiemeltIparagakban';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 7;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 7;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA field

\*/

const COL\_FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA = 'stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban.FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA';

/\*\*

\* the column name for the KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA field

\*/

const COL\_KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA = 'stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban.KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA';

/\*\*

\* the column name for the SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA field

\*/

const COL\_SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA = 'stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban.SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA';

/\*\*

\* the column name for the ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA field

\*/

const COL\_ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA = 'stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban.ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA';

/\*\*

\* the column name for the EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA field

\*/

const COL\_EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA = 'stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban.EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'FeldolgozoiparbanMukodoVallalkozasokSzama', 'KereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama', 'SzallitasbanRaktarozasbanMukodoVallalkozasokSzama', 'AdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama', 'EgyebSzolgaltatasbanMukodoVallalkozasokSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'feldolgozoiparbanMukodoVallalkozasokSzama', 'kereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama', 'szallitasbanRaktarozasbanMukodoVallalkozasokSzama', 'adminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama', 'egyebSzolgaltatasbanMukodoVallalkozasokSzama', ),

self::TYPE\_COLNAME => array(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_TELEPULES\_NEV, StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_TELEPULES\_KSHKOD, StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA', 'COL\_KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA', 'COL\_SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA', 'COL\_ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA', 'COL\_EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'feldolgozoiparban\_mukodo\_vallalkozasok\_szama', 'kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama', 'szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama', 'adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama', 'egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'FeldolgozoiparbanMukodoVallalkozasokSzama' => 2, 'KereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama' => 3, 'SzallitasbanRaktarozasbanMukodoVallalkozasokSzama' => 4, 'AdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama' => 5, 'EgyebSzolgaltatasbanMukodoVallalkozasokSzama' => 6, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'feldolgozoiparbanMukodoVallalkozasokSzama' => 2, 'kereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama' => 3, 'szallitasbanRaktarozasbanMukodoVallalkozasokSzama' => 4, 'adminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama' => 5, 'egyebSzolgaltatasbanMukodoVallalkozasokSzama' => 6, ),

self::TYPE\_COLNAME => array(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_TELEPULES\_NEV => 0, StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_TELEPULES\_KSHKOD => 1, StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA => 2, StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA => 3, StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA => 4, StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA => 5, StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA => 6, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA' => 2, 'COL\_KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA' => 3, 'COL\_SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA' => 4, 'COL\_ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA' => 5, 'COL\_EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA' => 6, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'feldolgozoiparban\_mukodo\_vallalkozasok\_szama' => 2, 'kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama' => 3, 'szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama' => 4, 'adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama' => 5, 'egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama' => 6, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban');

$this->setPhpName('StgMukodoVallalkozasokSzamaAKiemeltIparagakban');

$this->setClassName('\\StgMukodoVallalkozasokSzamaAKiemeltIparagakban');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA', 'FeldolgozoiparbanMukodoVallalkozasokSzama', 'INTEGER', false, null, null);

$this->addColumn('KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA', 'KereskedelembenGepjarmujavitasbanMukodoVallalkozasokSzama', 'INTEGER', false, null, null);

$this->addColumn('SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA', 'SzallitasbanRaktarozasbanMukodoVallalkozasokSzama', 'INTEGER', false, null, null);

$this->addColumn('ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA', 'AdminisztrativEsSzolgaltatasbanMukodoVallalkozasokSzama', 'INTEGER', false, null, null);

$this->addColumn('EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA', 'EgyebSzolgaltatasbanMukodoVallalkozasokSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::CLASS\_DEFAULT : StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgMukodoVallalkozasokSzamaAKiemeltIparagakban object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::OM\_CLASS;

/\*\* @var StgMukodoVallalkozasokSzamaAKiemeltIparagakban $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgMukodoVallalkozasokSzamaAKiemeltIparagakban $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA);

$criteria->addSelectColumn(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA);

$criteria->addSelectColumn(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA);

$criteria->addSelectColumn(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA);

$criteria->addSelectColumn(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::COL\_EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.FELDOLGOZOIPARBAN\_MUKODO\_VALLALKOZASOK\_SZAMA');

$criteria->addSelectColumn($alias . '.KERESKEDELEMBEN\_GEPJARMUJAVITASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA');

$criteria->addSelectColumn($alias . '.SZALLITASBAN\_RAKTAROZASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA');

$criteria->addSelectColumn($alias . '.ADMINISZTRATIV\_ES\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA');

$criteria->addSelectColumn($alias . '.EGYEB\_SZOLGALTATASBAN\_MUKODO\_VALLALKOZASOK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::DATABASE\_NAME)->getTable(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgMukodoVallalkozasokSzamaAKiemeltIparagakban or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgMukodoVallalkozasokSzamaAKiemeltIparagakban object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgMukodoVallalkozasokSzamaAKiemeltIparagakban) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgMukodoVallalkozasokSzamaAKiemeltIparagakban object has no primary key');

}

$query = StgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgMukodoVallalkozasokSzamaAKiemeltIparagakban or Criteria object.

\*

\* @param mixed $criteria Criteria or StgMukodoVallalkozasokSzamaAKiemeltIparagakban object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgMukodoVallalkozasokSzamaAKiemeltIparagakban object

}

// Set the correct dbName

$query = StgMukodoVallalkozasokSzamaAKiemeltIparagakbanQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgMukodoVallalkozasokSzamaAKiemeltIparagakbanTableMap::buildTableMap();

### StgMukodoVallalkozasokSzamaTableMap.php

<?php

namespace Map;

use \StgMukodoVallalkozasokSzama;

use \StgMukodoVallalkozasokSzamaQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_mukodo\_vallalkozasok\_szama' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgMukodoVallalkozasokSzamaTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgMukodoVallalkozasokSzamaTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_mukodo\_vallalkozasok\_szama';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgMukodoVallalkozasokSzama';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgMukodoVallalkozasokSzama';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 3;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 3;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_mukodo\_vallalkozasok\_szama.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_mukodo\_vallalkozasok\_szama.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the VALLALKOZASOK\_SZAMA field

\*/

const COL\_VALLALKOZASOK\_SZAMA = 'stg\_mukodo\_vallalkozasok\_szama.VALLALKOZASOK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'VallalkozasokSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'vallalkozasokSzama', ),

self::TYPE\_COLNAME => array(StgMukodoVallalkozasokSzamaTableMap::COL\_TELEPULES\_NEV, StgMukodoVallalkozasokSzamaTableMap::COL\_TELEPULES\_KSHKOD, StgMukodoVallalkozasokSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_VALLALKOZASOK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'vallalkozasok\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'VallalkozasokSzama' => 2, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'vallalkozasokSzama' => 2, ),

self::TYPE\_COLNAME => array(StgMukodoVallalkozasokSzamaTableMap::COL\_TELEPULES\_NEV => 0, StgMukodoVallalkozasokSzamaTableMap::COL\_TELEPULES\_KSHKOD => 1, StgMukodoVallalkozasokSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA => 2, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_VALLALKOZASOK\_SZAMA' => 2, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'vallalkozasok\_szama' => 2, ),

self::TYPE\_NUM => array(0, 1, 2, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_mukodo\_vallalkozasok\_szama');

$this->setPhpName('StgMukodoVallalkozasokSzama');

$this->setClassName('\\StgMukodoVallalkozasokSzama');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('VALLALKOZASOK\_SZAMA', 'VallalkozasokSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgMukodoVallalkozasokSzamaTableMap::CLASS\_DEFAULT : StgMukodoVallalkozasokSzamaTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgMukodoVallalkozasokSzama object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgMukodoVallalkozasokSzamaTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgMukodoVallalkozasokSzamaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgMukodoVallalkozasokSzamaTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgMukodoVallalkozasokSzamaTableMap::OM\_CLASS;

/\*\* @var StgMukodoVallalkozasokSzama $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgMukodoVallalkozasokSzamaTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgMukodoVallalkozasokSzamaTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgMukodoVallalkozasokSzamaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgMukodoVallalkozasokSzama $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgMukodoVallalkozasokSzamaTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgMukodoVallalkozasokSzamaTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgMukodoVallalkozasokSzamaTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgMukodoVallalkozasokSzamaTableMap::COL\_VALLALKOZASOK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.VALLALKOZASOK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgMukodoVallalkozasokSzamaTableMap::DATABASE\_NAME)->getTable(StgMukodoVallalkozasokSzamaTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgMukodoVallalkozasokSzamaTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgMukodoVallalkozasokSzamaTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgMukodoVallalkozasokSzamaTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgMukodoVallalkozasokSzama or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgMukodoVallalkozasokSzama object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMukodoVallalkozasokSzamaTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgMukodoVallalkozasokSzama) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgMukodoVallalkozasokSzama object has no primary key');

}

$query = StgMukodoVallalkozasokSzamaQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgMukodoVallalkozasokSzamaTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgMukodoVallalkozasokSzamaTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_mukodo\_vallalkozasok\_szama table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgMukodoVallalkozasokSzamaQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgMukodoVallalkozasokSzama or Criteria object.

\*

\* @param mixed $criteria Criteria or StgMukodoVallalkozasokSzama object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMukodoVallalkozasokSzamaTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgMukodoVallalkozasokSzama object

}

// Set the correct dbName

$query = StgMukodoVallalkozasokSzamaQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgMukodoVallalkozasokSzamaTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgMukodoVallalkozasokSzamaTableMap::buildTableMap();

### StgMunkakepesKoruNepessegTableMap.php

<?php

namespace Map;

use \StgMunkakepesKoruNepesseg;

use \StgMunkakepesKoruNepessegQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_munkakepes\_koru\_nepesseg' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgMunkakepesKoruNepessegTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgMunkakepesKoruNepessegTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_munkakepes\_koru\_nepesseg';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgMunkakepesKoruNepesseg';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgMunkakepesKoruNepesseg';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 3;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 3;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_munkakepes\_koru\_nepesseg.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_munkakepes\_koru\_nepesseg.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the NEPESSEG field

\*/

const COL\_NEPESSEG = 'stg\_munkakepes\_koru\_nepesseg.NEPESSEG';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'Nepesseg', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'nepesseg', ),

self::TYPE\_COLNAME => array(StgMunkakepesKoruNepessegTableMap::COL\_TELEPULES\_NEV, StgMunkakepesKoruNepessegTableMap::COL\_TELEPULES\_KSHKOD, StgMunkakepesKoruNepessegTableMap::COL\_NEPESSEG, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_NEPESSEG', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'nepesseg', ),

self::TYPE\_NUM => array(0, 1, 2, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'Nepesseg' => 2, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'nepesseg' => 2, ),

self::TYPE\_COLNAME => array(StgMunkakepesKoruNepessegTableMap::COL\_TELEPULES\_NEV => 0, StgMunkakepesKoruNepessegTableMap::COL\_TELEPULES\_KSHKOD => 1, StgMunkakepesKoruNepessegTableMap::COL\_NEPESSEG => 2, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_NEPESSEG' => 2, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'nepesseg' => 2, ),

self::TYPE\_NUM => array(0, 1, 2, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_munkakepes\_koru\_nepesseg');

$this->setPhpName('StgMunkakepesKoruNepesseg');

$this->setClassName('\\StgMunkakepesKoruNepesseg');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('NEPESSEG', 'Nepesseg', 'VARCHAR', false, 45, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgMunkakepesKoruNepessegTableMap::CLASS\_DEFAULT : StgMunkakepesKoruNepessegTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgMunkakepesKoruNepesseg object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgMunkakepesKoruNepessegTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgMunkakepesKoruNepessegTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgMunkakepesKoruNepessegTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgMunkakepesKoruNepessegTableMap::OM\_CLASS;

/\*\* @var StgMunkakepesKoruNepesseg $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgMunkakepesKoruNepessegTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgMunkakepesKoruNepessegTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgMunkakepesKoruNepessegTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgMunkakepesKoruNepesseg $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgMunkakepesKoruNepessegTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgMunkakepesKoruNepessegTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgMunkakepesKoruNepessegTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgMunkakepesKoruNepessegTableMap::COL\_NEPESSEG);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.NEPESSEG');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgMunkakepesKoruNepessegTableMap::DATABASE\_NAME)->getTable(StgMunkakepesKoruNepessegTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgMunkakepesKoruNepessegTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgMunkakepesKoruNepessegTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgMunkakepesKoruNepessegTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgMunkakepesKoruNepesseg or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgMunkakepesKoruNepesseg object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMunkakepesKoruNepessegTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgMunkakepesKoruNepesseg) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgMunkakepesKoruNepesseg object has no primary key');

}

$query = StgMunkakepesKoruNepessegQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgMunkakepesKoruNepessegTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgMunkakepesKoruNepessegTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_munkakepes\_koru\_nepesseg table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgMunkakepesKoruNepessegQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgMunkakepesKoruNepesseg or Criteria object.

\*

\* @param mixed $criteria Criteria or StgMunkakepesKoruNepesseg object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgMunkakepesKoruNepessegTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgMunkakepesKoruNepesseg object

}

// Set the correct dbName

$query = StgMunkakepesKoruNepessegQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgMunkakepesKoruNepessegTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgMunkakepesKoruNepessegTableMap::buildTableMap();

### StgNemzetkoziTranzitforgalomTableMap.php

<?php

namespace Map;

use \StgNemzetkoziTranzitforgalom;

use \StgNemzetkoziTranzitforgalomQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_nemzetkozi\_tranzitforgalom' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgNemzetkoziTranzitforgalomTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgNemzetkoziTranzitforgalomTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_nemzetkozi\_tranzitforgalom';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgNemzetkoziTranzitforgalom';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgNemzetkoziTranzitforgalom';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 3;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 3;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_nemzetkozi\_tranzitforgalom.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_nemzetkozi\_tranzitforgalom.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the HATARATLEPOK\_SZAMA field

\*/

const COL\_HATARATLEPOK\_SZAMA = 'stg\_nemzetkozi\_tranzitforgalom.HATARATLEPOK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'HataratlepokSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'hataratlepokSzama', ),

self::TYPE\_COLNAME => array(StgNemzetkoziTranzitforgalomTableMap::COL\_TELEPULES\_NEV, StgNemzetkoziTranzitforgalomTableMap::COL\_TELEPULES\_KSHKOD, StgNemzetkoziTranzitforgalomTableMap::COL\_HATARATLEPOK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_HATARATLEPOK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'hataratlepok\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'HataratlepokSzama' => 2, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'hataratlepokSzama' => 2, ),

self::TYPE\_COLNAME => array(StgNemzetkoziTranzitforgalomTableMap::COL\_TELEPULES\_NEV => 0, StgNemzetkoziTranzitforgalomTableMap::COL\_TELEPULES\_KSHKOD => 1, StgNemzetkoziTranzitforgalomTableMap::COL\_HATARATLEPOK\_SZAMA => 2, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_HATARATLEPOK\_SZAMA' => 2, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'hataratlepok\_szama' => 2, ),

self::TYPE\_NUM => array(0, 1, 2, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_nemzetkozi\_tranzitforgalom');

$this->setPhpName('StgNemzetkoziTranzitforgalom');

$this->setClassName('\\StgNemzetkoziTranzitforgalom');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('HATARATLEPOK\_SZAMA', 'HataratlepokSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgNemzetkoziTranzitforgalomTableMap::CLASS\_DEFAULT : StgNemzetkoziTranzitforgalomTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgNemzetkoziTranzitforgalom object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgNemzetkoziTranzitforgalomTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgNemzetkoziTranzitforgalomTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgNemzetkoziTranzitforgalomTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgNemzetkoziTranzitforgalomTableMap::OM\_CLASS;

/\*\* @var StgNemzetkoziTranzitforgalom $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgNemzetkoziTranzitforgalomTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgNemzetkoziTranzitforgalomTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgNemzetkoziTranzitforgalomTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgNemzetkoziTranzitforgalom $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgNemzetkoziTranzitforgalomTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgNemzetkoziTranzitforgalomTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgNemzetkoziTranzitforgalomTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgNemzetkoziTranzitforgalomTableMap::COL\_HATARATLEPOK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.HATARATLEPOK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgNemzetkoziTranzitforgalomTableMap::DATABASE\_NAME)->getTable(StgNemzetkoziTranzitforgalomTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgNemzetkoziTranzitforgalomTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgNemzetkoziTranzitforgalomTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgNemzetkoziTranzitforgalomTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgNemzetkoziTranzitforgalom or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgNemzetkoziTranzitforgalom object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNemzetkoziTranzitforgalomTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgNemzetkoziTranzitforgalom) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgNemzetkoziTranzitforgalom object has no primary key');

}

$query = StgNemzetkoziTranzitforgalomQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgNemzetkoziTranzitforgalomTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgNemzetkoziTranzitforgalomTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_nemzetkozi\_tranzitforgalom table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgNemzetkoziTranzitforgalomQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgNemzetkoziTranzitforgalom or Criteria object.

\*

\* @param mixed $criteria Criteria or StgNemzetkoziTranzitforgalom object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNemzetkoziTranzitforgalomTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgNemzetkoziTranzitforgalom object

}

// Set the correct dbName

$query = StgNemzetkoziTranzitforgalomQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgNemzetkoziTranzitforgalomTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgNemzetkoziTranzitforgalomTableMap::buildTableMap();

### StgNodeAttribsTableMap.php

<?php

namespace Map;

use \StgNodeAttribs;

use \StgNodeAttribsQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_node\_attribs' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgNodeAttribsTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgNodeAttribsTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_node\_attribs';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgNodeAttribs';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgNodeAttribs';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 37;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 37;

/\*\*

\* the column name for the SHAPE\_ID field

\*/

const COL\_SHAPE\_ID = 'stg\_node\_attribs.SHAPE\_ID';

/\*\*

\* the column name for the KSZAM field

\*/

const COL\_KSZAM = 'stg\_node\_attribs.KSZAM';

/\*\*

\* the column name for the PKOD field

\*/

const COL\_PKOD = 'stg\_node\_attribs.PKOD';

/\*\*

\* the column name for the KKOD field

\*/

const COL\_KKOD = 'stg\_node\_attribs.KKOD';

/\*\*

\* the column name for the VVKOD field

\*/

const COL\_VVKOD = 'stg\_node\_attribs.VVKOD';

/\*\*

\* the column name for the KSZELV field

\*/

const COL\_KSZELV = 'stg\_node\_attribs.KSZELV';

/\*\*

\* the column name for the VSZELV field

\*/

const COL\_VSZELV = 'stg\_node\_attribs.VSZELV';

/\*\*

\* the column name for the RSHOSSZ field

\*/

const COL\_RSHOSSZ = 'stg\_node\_attribs.RSHOSSZ';

/\*\*

\* the column name for the ANF field

\*/

const COL\_ANF = 'stg\_node\_attribs.ANF';

/\*\*

\* the column name for the ANET field

\*/

const COL\_ANET = 'stg\_node\_attribs.ANET';

/\*\*

\* the column name for the MOF field

\*/

const COL\_MOF = 'stg\_node\_attribs.MOF';

/\*\*

\* the column name for the ONGJ field

\*/

const COL\_ONGJ = 'stg\_node\_attribs.ONGJ';

/\*\*

\* the column name for the OJ field

\*/

const COL\_OJ = 'stg\_node\_attribs.OJ';

/\*\*

\* the column name for the OMOT field

\*/

const COL\_OMOT = 'stg\_node\_attribs.OMOT';

/\*\*

\* the column name for the EV field

\*/

const COL\_EV = 'stg\_node\_attribs.EV';

/\*\*

\* the column name for the ASZ field

\*/

const COL\_ASZ = 'stg\_node\_attribs.ASZ';

/\*\*

\* the column name for the BUSZCS field

\*/

const COL\_BUSZCS = 'stg\_node\_attribs.BUSZCS';

/\*\*

\* the column name for the BUSZE field

\*/

const COL\_BUSZE = 'stg\_node\_attribs.BUSZE';

/\*\*

\* the column name for the OBUSZ field

\*/

const COL\_OBUSZ = 'stg\_node\_attribs.OBUSZ';

/\*\*

\* the column name for the NYSZER field

\*/

const COL\_NYSZER = 'stg\_node\_attribs.NYSZER';

/\*\*

\* the column name for the POTKTGK field

\*/

const COL\_POTKTGK = 'stg\_node\_attribs.POTKTGK';

/\*\*

\* the column name for the KTGK field

\*/

const COL\_KTGK = 'stg\_node\_attribs.KTGK';

/\*\*

\* the column name for the NTGK field

\*/

const COL\_NTGK = 'stg\_node\_attribs.NTGK';

/\*\*

\* the column name for the KNTGK field

\*/

const COL\_KNTGK = 'stg\_node\_attribs.KNTGK';

/\*\*

\* the column name for the OTGK field

\*/

const COL\_OTGK = 'stg\_node\_attribs.OTGK';

/\*\*

\* the column name for the SZGK field

\*/

const COL\_SZGK = 'stg\_node\_attribs.SZGK';

/\*\*

\* the column name for the OSZGK field

\*/

const COL\_OSZGK = 'stg\_node\_attribs.OSZGK';

/\*\*

\* the column name for the MKP field

\*/

const COL\_MKP = 'stg\_node\_attribs.MKP';

/\*\*

\* the column name for the KPF field

\*/

const COL\_KPF = 'stg\_node\_attribs.KPF';

/\*\*

\* the column name for the LASSU field

\*/

const COL\_LASSU = 'stg\_node\_attribs.LASSU';

/\*\*

\* the column name for the SPEC field

\*/

const COL\_SPEC = 'stg\_node\_attribs.SPEC';

/\*\*

\* the column name for the FMEGB field

\*/

const COL\_FMEGB = 'stg\_node\_attribs.FMEGB';

/\*\*

\* the column name for the ADATFORR field

\*/

const COL\_ADATFORR = 'stg\_node\_attribs.ADATFORR';

/\*\*

\* the column name for the SZAMLNAP field

\*/

const COL\_SZAMLNAP = 'stg\_node\_attribs.SZAMLNAP';

/\*\*

\* the column name for the JELLEG1 field

\*/

const COL\_JELLEG1 = 'stg\_node\_attribs.JELLEG1';

/\*\*

\* the column name for the JELLEG2 field

\*/

const COL\_JELLEG2 = 'stg\_node\_attribs.JELLEG2';

/\*\*

\* the column name for the FMEGJ field

\*/

const COL\_FMEGJ = 'stg\_node\_attribs.FMEGJ';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('ShapeId', 'Kszam', 'Pkod', 'Kkod', 'Vvkod', 'Kszelv', 'Vszelv', 'Rshossz', 'Anf', 'Anet', 'Mof', 'Ongj', 'Oj', 'Omot', 'Ev', 'Asz', 'Buszcs', 'Busze', 'Obusz', 'Nyszer', 'Potktgk', 'Ktgk', 'Ntgk', 'Kntgk', 'Otgk', 'Szgk', 'Oszgk', 'Mkp', 'Kpf', 'Lassu', 'Spec', 'Fmegb', 'Adatforr', 'Szamlnap', 'Jelleg1', 'Jelleg2', 'Fmegj', ),

self::TYPE\_STUDLYPHPNAME => array('shapeId', 'kszam', 'pkod', 'kkod', 'vvkod', 'kszelv', 'vszelv', 'rshossz', 'anf', 'anet', 'mof', 'ongj', 'oj', 'omot', 'ev', 'asz', 'buszcs', 'busze', 'obusz', 'nyszer', 'potktgk', 'ktgk', 'ntgk', 'kntgk', 'otgk', 'szgk', 'oszgk', 'mkp', 'kpf', 'lassu', 'spec', 'fmegb', 'adatforr', 'szamlnap', 'jelleg1', 'jelleg2', 'fmegj', ),

self::TYPE\_COLNAME => array(StgNodeAttribsTableMap::COL\_SHAPE\_ID, StgNodeAttribsTableMap::COL\_KSZAM, StgNodeAttribsTableMap::COL\_PKOD, StgNodeAttribsTableMap::COL\_KKOD, StgNodeAttribsTableMap::COL\_VVKOD, StgNodeAttribsTableMap::COL\_KSZELV, StgNodeAttribsTableMap::COL\_VSZELV, StgNodeAttribsTableMap::COL\_RSHOSSZ, StgNodeAttribsTableMap::COL\_ANF, StgNodeAttribsTableMap::COL\_ANET, StgNodeAttribsTableMap::COL\_MOF, StgNodeAttribsTableMap::COL\_ONGJ, StgNodeAttribsTableMap::COL\_OJ, StgNodeAttribsTableMap::COL\_OMOT, StgNodeAttribsTableMap::COL\_EV, StgNodeAttribsTableMap::COL\_ASZ, StgNodeAttribsTableMap::COL\_BUSZCS, StgNodeAttribsTableMap::COL\_BUSZE, StgNodeAttribsTableMap::COL\_OBUSZ, StgNodeAttribsTableMap::COL\_NYSZER, StgNodeAttribsTableMap::COL\_POTKTGK, StgNodeAttribsTableMap::COL\_KTGK, StgNodeAttribsTableMap::COL\_NTGK, StgNodeAttribsTableMap::COL\_KNTGK, StgNodeAttribsTableMap::COL\_OTGK, StgNodeAttribsTableMap::COL\_SZGK, StgNodeAttribsTableMap::COL\_OSZGK, StgNodeAttribsTableMap::COL\_MKP, StgNodeAttribsTableMap::COL\_KPF, StgNodeAttribsTableMap::COL\_LASSU, StgNodeAttribsTableMap::COL\_SPEC, StgNodeAttribsTableMap::COL\_FMEGB, StgNodeAttribsTableMap::COL\_ADATFORR, StgNodeAttribsTableMap::COL\_SZAMLNAP, StgNodeAttribsTableMap::COL\_JELLEG1, StgNodeAttribsTableMap::COL\_JELLEG2, StgNodeAttribsTableMap::COL\_FMEGJ, ),

self::TYPE\_RAW\_COLNAME => array('COL\_SHAPE\_ID', 'COL\_KSZAM', 'COL\_PKOD', 'COL\_KKOD', 'COL\_VVKOD', 'COL\_KSZELV', 'COL\_VSZELV', 'COL\_RSHOSSZ', 'COL\_ANF', 'COL\_ANET', 'COL\_MOF', 'COL\_ONGJ', 'COL\_OJ', 'COL\_OMOT', 'COL\_EV', 'COL\_ASZ', 'COL\_BUSZCS', 'COL\_BUSZE', 'COL\_OBUSZ', 'COL\_NYSZER', 'COL\_POTKTGK', 'COL\_KTGK', 'COL\_NTGK', 'COL\_KNTGK', 'COL\_OTGK', 'COL\_SZGK', 'COL\_OSZGK', 'COL\_MKP', 'COL\_KPF', 'COL\_LASSU', 'COL\_SPEC', 'COL\_FMEGB', 'COL\_ADATFORR', 'COL\_SZAMLNAP', 'COL\_JELLEG1', 'COL\_JELLEG2', 'COL\_FMEGJ', ),

self::TYPE\_FIELDNAME => array('shape\_id', 'kszam', 'pkod', 'kkod', 'vvkod', 'kszelv', 'vszelv', 'RSHOSSZ', 'ANF', 'ANET', 'MOF', 'ONGJ', 'OJ', 'OMOT', 'EV', 'ASZ', 'BUSZCS', 'BUSZE', 'OBUSZ', 'NYSZER', 'POTKTGK', 'KTGK', 'NTGK', 'KNTGK', 'OTGK', 'SZGK', 'OSZGK', 'MKP', 'KPF', 'LASSU', 'SPEC', 'FMEGB', 'ADATFORR', 'SZAMLNAP', 'JELLEG1', 'JELLEG2', 'FMEGJ', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('ShapeId' => 0, 'Kszam' => 1, 'Pkod' => 2, 'Kkod' => 3, 'Vvkod' => 4, 'Kszelv' => 5, 'Vszelv' => 6, 'Rshossz' => 7, 'Anf' => 8, 'Anet' => 9, 'Mof' => 10, 'Ongj' => 11, 'Oj' => 12, 'Omot' => 13, 'Ev' => 14, 'Asz' => 15, 'Buszcs' => 16, 'Busze' => 17, 'Obusz' => 18, 'Nyszer' => 19, 'Potktgk' => 20, 'Ktgk' => 21, 'Ntgk' => 22, 'Kntgk' => 23, 'Otgk' => 24, 'Szgk' => 25, 'Oszgk' => 26, 'Mkp' => 27, 'Kpf' => 28, 'Lassu' => 29, 'Spec' => 30, 'Fmegb' => 31, 'Adatforr' => 32, 'Szamlnap' => 33, 'Jelleg1' => 34, 'Jelleg2' => 35, 'Fmegj' => 36, ),

self::TYPE\_STUDLYPHPNAME => array('shapeId' => 0, 'kszam' => 1, 'pkod' => 2, 'kkod' => 3, 'vvkod' => 4, 'kszelv' => 5, 'vszelv' => 6, 'rshossz' => 7, 'anf' => 8, 'anet' => 9, 'mof' => 10, 'ongj' => 11, 'oj' => 12, 'omot' => 13, 'ev' => 14, 'asz' => 15, 'buszcs' => 16, 'busze' => 17, 'obusz' => 18, 'nyszer' => 19, 'potktgk' => 20, 'ktgk' => 21, 'ntgk' => 22, 'kntgk' => 23, 'otgk' => 24, 'szgk' => 25, 'oszgk' => 26, 'mkp' => 27, 'kpf' => 28, 'lassu' => 29, 'spec' => 30, 'fmegb' => 31, 'adatforr' => 32, 'szamlnap' => 33, 'jelleg1' => 34, 'jelleg2' => 35, 'fmegj' => 36, ),

self::TYPE\_COLNAME => array(StgNodeAttribsTableMap::COL\_SHAPE\_ID => 0, StgNodeAttribsTableMap::COL\_KSZAM => 1, StgNodeAttribsTableMap::COL\_PKOD => 2, StgNodeAttribsTableMap::COL\_KKOD => 3, StgNodeAttribsTableMap::COL\_VVKOD => 4, StgNodeAttribsTableMap::COL\_KSZELV => 5, StgNodeAttribsTableMap::COL\_VSZELV => 6, StgNodeAttribsTableMap::COL\_RSHOSSZ => 7, StgNodeAttribsTableMap::COL\_ANF => 8, StgNodeAttribsTableMap::COL\_ANET => 9, StgNodeAttribsTableMap::COL\_MOF => 10, StgNodeAttribsTableMap::COL\_ONGJ => 11, StgNodeAttribsTableMap::COL\_OJ => 12, StgNodeAttribsTableMap::COL\_OMOT => 13, StgNodeAttribsTableMap::COL\_EV => 14, StgNodeAttribsTableMap::COL\_ASZ => 15, StgNodeAttribsTableMap::COL\_BUSZCS => 16, StgNodeAttribsTableMap::COL\_BUSZE => 17, StgNodeAttribsTableMap::COL\_OBUSZ => 18, StgNodeAttribsTableMap::COL\_NYSZER => 19, StgNodeAttribsTableMap::COL\_POTKTGK => 20, StgNodeAttribsTableMap::COL\_KTGK => 21, StgNodeAttribsTableMap::COL\_NTGK => 22, StgNodeAttribsTableMap::COL\_KNTGK => 23, StgNodeAttribsTableMap::COL\_OTGK => 24, StgNodeAttribsTableMap::COL\_SZGK => 25, StgNodeAttribsTableMap::COL\_OSZGK => 26, StgNodeAttribsTableMap::COL\_MKP => 27, StgNodeAttribsTableMap::COL\_KPF => 28, StgNodeAttribsTableMap::COL\_LASSU => 29, StgNodeAttribsTableMap::COL\_SPEC => 30, StgNodeAttribsTableMap::COL\_FMEGB => 31, StgNodeAttribsTableMap::COL\_ADATFORR => 32, StgNodeAttribsTableMap::COL\_SZAMLNAP => 33, StgNodeAttribsTableMap::COL\_JELLEG1 => 34, StgNodeAttribsTableMap::COL\_JELLEG2 => 35, StgNodeAttribsTableMap::COL\_FMEGJ => 36, ),

self::TYPE\_RAW\_COLNAME => array('COL\_SHAPE\_ID' => 0, 'COL\_KSZAM' => 1, 'COL\_PKOD' => 2, 'COL\_KKOD' => 3, 'COL\_VVKOD' => 4, 'COL\_KSZELV' => 5, 'COL\_VSZELV' => 6, 'COL\_RSHOSSZ' => 7, 'COL\_ANF' => 8, 'COL\_ANET' => 9, 'COL\_MOF' => 10, 'COL\_ONGJ' => 11, 'COL\_OJ' => 12, 'COL\_OMOT' => 13, 'COL\_EV' => 14, 'COL\_ASZ' => 15, 'COL\_BUSZCS' => 16, 'COL\_BUSZE' => 17, 'COL\_OBUSZ' => 18, 'COL\_NYSZER' => 19, 'COL\_POTKTGK' => 20, 'COL\_KTGK' => 21, 'COL\_NTGK' => 22, 'COL\_KNTGK' => 23, 'COL\_OTGK' => 24, 'COL\_SZGK' => 25, 'COL\_OSZGK' => 26, 'COL\_MKP' => 27, 'COL\_KPF' => 28, 'COL\_LASSU' => 29, 'COL\_SPEC' => 30, 'COL\_FMEGB' => 31, 'COL\_ADATFORR' => 32, 'COL\_SZAMLNAP' => 33, 'COL\_JELLEG1' => 34, 'COL\_JELLEG2' => 35, 'COL\_FMEGJ' => 36, ),

self::TYPE\_FIELDNAME => array('shape\_id' => 0, 'kszam' => 1, 'pkod' => 2, 'kkod' => 3, 'vvkod' => 4, 'kszelv' => 5, 'vszelv' => 6, 'RSHOSSZ' => 7, 'ANF' => 8, 'ANET' => 9, 'MOF' => 10, 'ONGJ' => 11, 'OJ' => 12, 'OMOT' => 13, 'EV' => 14, 'ASZ' => 15, 'BUSZCS' => 16, 'BUSZE' => 17, 'OBUSZ' => 18, 'NYSZER' => 19, 'POTKTGK' => 20, 'KTGK' => 21, 'NTGK' => 22, 'KNTGK' => 23, 'OTGK' => 24, 'SZGK' => 25, 'OSZGK' => 26, 'MKP' => 27, 'KPF' => 28, 'LASSU' => 29, 'SPEC' => 30, 'FMEGB' => 31, 'ADATFORR' => 32, 'SZAMLNAP' => 33, 'JELLEG1' => 34, 'JELLEG2' => 35, 'FMEGJ' => 36, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_node\_attribs');

$this->setPhpName('StgNodeAttribs');

$this->setClassName('\\StgNodeAttribs');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('SHAPE\_ID', 'ShapeId', 'INTEGER', false, null, null);

$this->addColumn('KSZAM', 'Kszam', 'VARCHAR', false, 6, null);

$this->addColumn('PKOD', 'Pkod', 'INTEGER', false, null, null);

$this->addColumn('KKOD', 'Kkod', 'VARCHAR', false, 15, null);

$this->addColumn('VVKOD', 'Vvkod', 'VARCHAR', false, 15, null);

$this->addColumn('KSZELV', 'Kszelv', 'VARCHAR', false, 15, null);

$this->addColumn('VSZELV', 'Vszelv', 'VARCHAR', false, 15, null);

$this->addColumn('RSHOSSZ', 'Rshossz', 'DOUBLE', false, null, null);

$this->addColumn('ANF', 'Anf', 'SMALLINT', false, 9, null);

$this->addColumn('ANET', 'Anet', 'SMALLINT', false, null, null);

$this->addColumn('MOF', 'Mof', 'SMALLINT', false, null, null);

$this->addColumn('ONGJ', 'Ongj', 'SMALLINT', false, null, null);

$this->addColumn('OJ', 'Oj', 'SMALLINT', false, 9, null);

$this->addColumn('OMOT', 'Omot', 'SMALLINT', false, 9, null);

$this->addColumn('EV', 'Ev', 'SMALLINT', false, null, null);

$this->addColumn('ASZ', 'Asz', 'INTEGER', false, null, null);

$this->addColumn('BUSZCS', 'Buszcs', 'INTEGER', false, null, null);

$this->addColumn('BUSZE', 'Busze', 'SMALLINT', false, null, null);

$this->addColumn('OBUSZ', 'Obusz', 'SMALLINT', false, null, null);

$this->addColumn('NYSZER', 'Nyszer', 'SMALLINT', false, null, null);

$this->addColumn('POTKTGK', 'Potktgk', 'SMALLINT', false, 9, null);

$this->addColumn('KTGK', 'Ktgk', 'SMALLINT', false, 9, null);

$this->addColumn('NTGK', 'Ntgk', 'SMALLINT', false, 9, null);

$this->addColumn('KNTGK', 'Kntgk', 'SMALLINT', false, 9, null);

$this->addColumn('OTGK', 'Otgk', 'SMALLINT', false, 9, null);

$this->addColumn('SZGK', 'Szgk', 'SMALLINT', false, 9, null);

$this->addColumn('OSZGK', 'Oszgk', 'SMALLINT', false, 9, null);

$this->addColumn('MKP', 'Mkp', 'SMALLINT', false, 9, null);

$this->addColumn('KPF', 'Kpf', 'SMALLINT', false, 9, null);

$this->addColumn('LASSU', 'Lassu', 'SMALLINT', false, 9, null);

$this->addColumn('SPEC', 'Spec', 'SMALLINT', false, 9, null);

$this->addColumn('FMEGB', 'Fmegb', 'VARCHAR', false, 45, null);

$this->addColumn('ADATFORR', 'Adatforr', 'VARCHAR', false, 45, null);

$this->addColumn('SZAMLNAP', 'Szamlnap', 'SMALLINT', false, null, null);

$this->addColumn('JELLEG1', 'Jelleg1', 'VARCHAR', false, 45, null);

$this->addColumn('JELLEG2', 'Jelleg2', 'VARCHAR', false, 45, null);

$this->addColumn('FMEGJ', 'Fmegj', 'VARCHAR', false, 45, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgNodeAttribsTableMap::CLASS\_DEFAULT : StgNodeAttribsTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgNodeAttribs object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgNodeAttribsTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgNodeAttribsTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgNodeAttribsTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgNodeAttribsTableMap::OM\_CLASS;

/\*\* @var StgNodeAttribs $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgNodeAttribsTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgNodeAttribsTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgNodeAttribsTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgNodeAttribs $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgNodeAttribsTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_SHAPE\_ID);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_KSZAM);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_PKOD);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_KKOD);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_VVKOD);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_KSZELV);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_VSZELV);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_RSHOSSZ);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_ANF);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_ANET);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_MOF);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_ONGJ);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_OJ);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_OMOT);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_EV);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_ASZ);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_BUSZCS);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_BUSZE);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_OBUSZ);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_NYSZER);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_POTKTGK);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_KTGK);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_NTGK);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_KNTGK);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_OTGK);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_SZGK);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_OSZGK);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_MKP);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_KPF);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_LASSU);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_SPEC);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_FMEGB);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_ADATFORR);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_SZAMLNAP);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_JELLEG1);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_JELLEG2);

$criteria->addSelectColumn(StgNodeAttribsTableMap::COL\_FMEGJ);

} else {

$criteria->addSelectColumn($alias . '.SHAPE\_ID');

$criteria->addSelectColumn($alias . '.KSZAM');

$criteria->addSelectColumn($alias . '.PKOD');

$criteria->addSelectColumn($alias . '.KKOD');

$criteria->addSelectColumn($alias . '.VVKOD');

$criteria->addSelectColumn($alias . '.KSZELV');

$criteria->addSelectColumn($alias . '.VSZELV');

$criteria->addSelectColumn($alias . '.RSHOSSZ');

$criteria->addSelectColumn($alias . '.ANF');

$criteria->addSelectColumn($alias . '.ANET');

$criteria->addSelectColumn($alias . '.MOF');

$criteria->addSelectColumn($alias . '.ONGJ');

$criteria->addSelectColumn($alias . '.OJ');

$criteria->addSelectColumn($alias . '.OMOT');

$criteria->addSelectColumn($alias . '.EV');

$criteria->addSelectColumn($alias . '.ASZ');

$criteria->addSelectColumn($alias . '.BUSZCS');

$criteria->addSelectColumn($alias . '.BUSZE');

$criteria->addSelectColumn($alias . '.OBUSZ');

$criteria->addSelectColumn($alias . '.NYSZER');

$criteria->addSelectColumn($alias . '.POTKTGK');

$criteria->addSelectColumn($alias . '.KTGK');

$criteria->addSelectColumn($alias . '.NTGK');

$criteria->addSelectColumn($alias . '.KNTGK');

$criteria->addSelectColumn($alias . '.OTGK');

$criteria->addSelectColumn($alias . '.SZGK');

$criteria->addSelectColumn($alias . '.OSZGK');

$criteria->addSelectColumn($alias . '.MKP');

$criteria->addSelectColumn($alias . '.KPF');

$criteria->addSelectColumn($alias . '.LASSU');

$criteria->addSelectColumn($alias . '.SPEC');

$criteria->addSelectColumn($alias . '.FMEGB');

$criteria->addSelectColumn($alias . '.ADATFORR');

$criteria->addSelectColumn($alias . '.SZAMLNAP');

$criteria->addSelectColumn($alias . '.JELLEG1');

$criteria->addSelectColumn($alias . '.JELLEG2');

$criteria->addSelectColumn($alias . '.FMEGJ');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgNodeAttribsTableMap::DATABASE\_NAME)->getTable(StgNodeAttribsTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgNodeAttribsTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgNodeAttribsTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgNodeAttribsTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgNodeAttribs or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgNodeAttribs object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNodeAttribsTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgNodeAttribs) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgNodeAttribs object has no primary key');

}

$query = StgNodeAttribsQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgNodeAttribsTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgNodeAttribsTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_node\_attribs table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgNodeAttribsQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgNodeAttribs or Criteria object.

\*

\* @param mixed $criteria Criteria or StgNodeAttribs object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNodeAttribsTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgNodeAttribs object

}

// Set the correct dbName

$query = StgNodeAttribsQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgNodeAttribsTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgNodeAttribsTableMap::buildTableMap();

### StgNodesTableMap.php

<?php

namespace Map;

use \StgNodes;

use \StgNodesQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_nodes' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgNodesTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgNodesTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_nodes';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgNodes';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgNodes';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 4;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 4;

/\*\*

\* the column name for the SHAPE\_ID field

\*/

const COL\_SHAPE\_ID = 'stg\_nodes.SHAPE\_ID';

/\*\*

\* the column name for the X field

\*/

const COL\_X = 'stg\_nodes.X';

/\*\*

\* the column name for the Y field

\*/

const COL\_Y = 'stg\_nodes.Y';

/\*\*

\* the column name for the NUMBER field

\*/

const COL\_NUMBER = 'stg\_nodes.NUMBER';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('ShapeId', 'X', 'Y', 'Number', ),

self::TYPE\_STUDLYPHPNAME => array('shapeId', 'x', 'y', 'number', ),

self::TYPE\_COLNAME => array(StgNodesTableMap::COL\_SHAPE\_ID, StgNodesTableMap::COL\_X, StgNodesTableMap::COL\_Y, StgNodesTableMap::COL\_NUMBER, ),

self::TYPE\_RAW\_COLNAME => array('COL\_SHAPE\_ID', 'COL\_X', 'COL\_Y', 'COL\_NUMBER', ),

self::TYPE\_FIELDNAME => array('shape\_id', 'x', 'y', 'number', ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('ShapeId' => 0, 'X' => 1, 'Y' => 2, 'Number' => 3, ),

self::TYPE\_STUDLYPHPNAME => array('shapeId' => 0, 'x' => 1, 'y' => 2, 'number' => 3, ),

self::TYPE\_COLNAME => array(StgNodesTableMap::COL\_SHAPE\_ID => 0, StgNodesTableMap::COL\_X => 1, StgNodesTableMap::COL\_Y => 2, StgNodesTableMap::COL\_NUMBER => 3, ),

self::TYPE\_RAW\_COLNAME => array('COL\_SHAPE\_ID' => 0, 'COL\_X' => 1, 'COL\_Y' => 2, 'COL\_NUMBER' => 3, ),

self::TYPE\_FIELDNAME => array('shape\_id' => 0, 'x' => 1, 'y' => 2, 'number' => 3, ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_nodes');

$this->setPhpName('StgNodes');

$this->setClassName('\\StgNodes');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('SHAPE\_ID', 'ShapeId', 'INTEGER', true, null, null);

$this->addColumn('X', 'X', 'DOUBLE', true, null, null);

$this->addColumn('Y', 'Y', 'DOUBLE', true, null, null);

$this->addColumn('NUMBER', 'Number', 'BIGINT', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgNodesTableMap::CLASS\_DEFAULT : StgNodesTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgNodes object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgNodesTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgNodesTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgNodesTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgNodesTableMap::OM\_CLASS;

/\*\* @var StgNodes $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgNodesTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgNodesTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgNodesTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgNodes $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgNodesTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgNodesTableMap::COL\_SHAPE\_ID);

$criteria->addSelectColumn(StgNodesTableMap::COL\_X);

$criteria->addSelectColumn(StgNodesTableMap::COL\_Y);

$criteria->addSelectColumn(StgNodesTableMap::COL\_NUMBER);

} else {

$criteria->addSelectColumn($alias . '.SHAPE\_ID');

$criteria->addSelectColumn($alias . '.X');

$criteria->addSelectColumn($alias . '.Y');

$criteria->addSelectColumn($alias . '.NUMBER');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgNodesTableMap::DATABASE\_NAME)->getTable(StgNodesTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgNodesTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgNodesTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgNodesTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgNodes or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgNodes object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNodesTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgNodes) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgNodes object has no primary key');

}

$query = StgNodesQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgNodesTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgNodesTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_nodes table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgNodesQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgNodes or Criteria object.

\*

\* @param mixed $criteria Criteria or StgNodes object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgNodesTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgNodes object

}

// Set the correct dbName

$query = StgNodesQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgNodesTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgNodesTableMap::buildTableMap();

### StgOktatasiFerohelyekSzamaTableMap.php

<?php

namespace Map;

use \StgOktatasiFerohelyekSzama;

use \StgOktatasiFerohelyekSzamaQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_oktatasi\_ferohelyek\_szama' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgOktatasiFerohelyekSzamaTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgOktatasiFerohelyekSzamaTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_oktatasi\_ferohelyek\_szama';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgOktatasiFerohelyekSzama';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgOktatasiFerohelyekSzama';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 7;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 7;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_oktatasi\_ferohelyek\_szama.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_oktatasi\_ferohelyek\_szama.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA field

\*/

const COL\_NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA = 'stg\_oktatasi\_ferohelyek\_szama.NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA';

/\*\*

\* the column name for the NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA field

\*/

const COL\_NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA = 'stg\_oktatasi\_ferohelyek\_szama.NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA';

/\*\*

\* the column name for the NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA field

\*/

const COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA = 'stg\_oktatasi\_ferohelyek\_szama.NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA';

/\*\*

\* the column name for the NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN field

\*/

const COL\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN = 'stg\_oktatasi\_ferohelyek\_szama.NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN';

/\*\*

\* the column name for the NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN field

\*/

const COL\_NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN = 'stg\_oktatasi\_ferohelyek\_szama.NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'NappaliAltIskolaiOsztalyokSzama', 'NappaliKozepiskolaiOsztalyokSzama', 'NappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama', 'NappaliHallgatokSzamaFelsofokuAlapMesterkepzesben', 'NemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'nappaliAltIskolaiOsztalyokSzama', 'nappaliKozepiskolaiOsztalyokSzama', 'nappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama', 'nappaliHallgatokSzamaFelsofokuAlapMesterkepzesben', 'nemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben', ),

self::TYPE\_COLNAME => array(StgOktatasiFerohelyekSzamaTableMap::COL\_TELEPULES\_NEV, StgOktatasiFerohelyekSzamaTableMap::COL\_TELEPULES\_KSHKOD, StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA, StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA, StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA, StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN, StgOktatasiFerohelyekSzamaTableMap::COL\_NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA', 'COL\_NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA', 'COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA', 'COL\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN', 'COL\_NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKod', 'nappali\_alt\_iskolai\_osztalyok\_szama', 'nappali\_kozepiskolai\_osztalyok\_szama', 'nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama', 'nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben', 'nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'NappaliAltIskolaiOsztalyokSzama' => 2, 'NappaliKozepiskolaiOsztalyokSzama' => 3, 'NappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama' => 4, 'NappaliHallgatokSzamaFelsofokuAlapMesterkepzesben' => 5, 'NemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben' => 6, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'nappaliAltIskolaiOsztalyokSzama' => 2, 'nappaliKozepiskolaiOsztalyokSzama' => 3, 'nappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama' => 4, 'nappaliHallgatokSzamaFelsofokuAlapMesterkepzesben' => 5, 'nemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben' => 6, ),

self::TYPE\_COLNAME => array(StgOktatasiFerohelyekSzamaTableMap::COL\_TELEPULES\_NEV => 0, StgOktatasiFerohelyekSzamaTableMap::COL\_TELEPULES\_KSHKOD => 1, StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA => 2, StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA => 3, StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA => 4, StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN => 5, StgOktatasiFerohelyekSzamaTableMap::COL\_NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN => 6, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA' => 2, 'COL\_NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA' => 3, 'COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA' => 4, 'COL\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN' => 5, 'COL\_NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN' => 6, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKod' => 1, 'nappali\_alt\_iskolai\_osztalyok\_szama' => 2, 'nappali\_kozepiskolai\_osztalyok\_szama' => 3, 'nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama' => 4, 'nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben' => 5, 'nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben' => 6, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_oktatasi\_ferohelyek\_szama');

$this->setPhpName('StgOktatasiFerohelyekSzama');

$this->setClassName('\\StgOktatasiFerohelyekSzama');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA', 'NappaliAltIskolaiOsztalyokSzama', 'INTEGER', false, null, null);

$this->addColumn('NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA', 'NappaliKozepiskolaiOsztalyokSzama', 'INTEGER', false, null, null);

$this->addColumn('NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA', 'NappaliSzakiskolaiSpecSzakiskolaiOsztalyokSzama', 'INTEGER', false, null, null);

$this->addColumn('NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN', 'NappaliHallgatokSzamaFelsofokuAlapMesterkepzesben', 'INTEGER', false, null, null);

$this->addColumn('NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN', 'NemNappaliHallgatokSzamaFelsofokuAlapMesterkepzesben', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgOktatasiFerohelyekSzamaTableMap::CLASS\_DEFAULT : StgOktatasiFerohelyekSzamaTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgOktatasiFerohelyekSzama object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgOktatasiFerohelyekSzamaTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgOktatasiFerohelyekSzamaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgOktatasiFerohelyekSzamaTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgOktatasiFerohelyekSzamaTableMap::OM\_CLASS;

/\*\* @var StgOktatasiFerohelyekSzama $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgOktatasiFerohelyekSzamaTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgOktatasiFerohelyekSzamaTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgOktatasiFerohelyekSzamaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgOktatasiFerohelyekSzama $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgOktatasiFerohelyekSzamaTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgOktatasiFerohelyekSzamaTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgOktatasiFerohelyekSzamaTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA);

$criteria->addSelectColumn(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA);

$criteria->addSelectColumn(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA);

$criteria->addSelectColumn(StgOktatasiFerohelyekSzamaTableMap::COL\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN);

$criteria->addSelectColumn(StgOktatasiFerohelyekSzamaTableMap::COL\_NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.NAPPALI\_ALT\_ISKOLAI\_OSZTALYOK\_SZAMA');

$criteria->addSelectColumn($alias . '.NAPPALI\_KOZEPISKOLAI\_OSZTALYOK\_SZAMA');

$criteria->addSelectColumn($alias . '.NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_OSZTALYOK\_SZAMA');

$criteria->addSelectColumn($alias . '.NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN');

$criteria->addSelectColumn($alias . '.NEM\_NAPPALI\_HALLGATOK\_SZAMA\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgOktatasiFerohelyekSzamaTableMap::DATABASE\_NAME)->getTable(StgOktatasiFerohelyekSzamaTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgOktatasiFerohelyekSzamaTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgOktatasiFerohelyekSzamaTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgOktatasiFerohelyekSzamaTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgOktatasiFerohelyekSzama or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgOktatasiFerohelyekSzama object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgOktatasiFerohelyekSzamaTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgOktatasiFerohelyekSzama) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgOktatasiFerohelyekSzama object has no primary key');

}

$query = StgOktatasiFerohelyekSzamaQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgOktatasiFerohelyekSzamaTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgOktatasiFerohelyekSzamaTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_oktatasi\_ferohelyek\_szama table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgOktatasiFerohelyekSzamaQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgOktatasiFerohelyekSzama or Criteria object.

\*

\* @param mixed $criteria Criteria or StgOktatasiFerohelyekSzama object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgOktatasiFerohelyekSzamaTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgOktatasiFerohelyekSzama object

}

// Set the correct dbName

$query = StgOktatasiFerohelyekSzamaQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgOktatasiFerohelyekSzamaTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgOktatasiFerohelyekSzamaTableMap::buildTableMap();

### StgSzabadIskolavalasztasValtozoTableMap.php

<?php

namespace Map;

use \StgSzabadIskolavalasztasValtozo;

use \StgSzabadIskolavalasztasValtozoQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_szabad\_iskolavalasztas\_valtozo' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgSzabadIskolavalasztasValtozoTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgSzabadIskolavalasztasValtozoTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_szabad\_iskolavalasztas\_valtozo';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgSzabadIskolavalasztasValtozo';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgSzabadIskolavalasztasValtozo';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 10;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 10;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_szabad\_iskolavalasztas\_valtozo.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_szabad\_iskolavalasztas\_valtozo.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA field

\*/

const COL\_ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA = 'stg\_szabad\_iskolavalasztas\_valtozo.ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA';

/\*\*

\* the column name for the ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA field

\*/

const COL\_ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA = 'stg\_szabad\_iskolavalasztas\_valtozo.ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA';

/\*\*

\* the column name for the ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA field

\*/

const COL\_ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA = 'stg\_szabad\_iskolavalasztas\_valtozo.ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA';

/\*\*

\* the column name for the NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA field

\*/

const COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA = 'stg\_szabad\_iskolavalasztas\_valtozo.NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA';

/\*\*

\* the column name for the NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA field

\*/

const COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA = 'stg\_szabad\_iskolavalasztas\_valtozo.NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA';

/\*\*

\* the column name for the NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA field

\*/

const COL\_NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA = 'stg\_szabad\_iskolavalasztas\_valtozo.NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA';

/\*\*

\* the column name for the MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA field

\*/

const COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA = 'stg\_szabad\_iskolavalasztas\_valtozo.MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA';

/\*\*

\* the column name for the MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA field

\*/

const COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA = 'stg\_szabad\_iskolavalasztas\_valtozo.MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'AllandoNepessegbol613EvesekSzama', 'AllandoNepessegbol14EvesekSzama', 'AllandoNepessegbol1517EvesekSzama', 'NappaliAltalanosIskolaiTanulokSzama', 'NappaliKozepiskolaiTanulokSzama', 'NappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama', 'MasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama', 'MasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'allandoNepessegbol613EvesekSzama', 'allandoNepessegbol14EvesekSzama', 'allandoNepessegbol1517EvesekSzama', 'nappaliAltalanosIskolaiTanulokSzama', 'nappaliKozepiskolaiTanulokSzama', 'nappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama', 'masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama', 'masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama', ),

self::TYPE\_COLNAME => array(StgSzabadIskolavalasztasValtozoTableMap::COL\_TELEPULES\_NEV, StgSzabadIskolavalasztasValtozoTableMap::COL\_TELEPULES\_KSHKOD, StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA, StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA, StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA, StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA, StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA, StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA', 'COL\_ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA', 'COL\_ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA', 'COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA', 'COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA', 'COL\_NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA', 'COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA', 'COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'allando\_nepessegbol\_6\_13\_evesek\_szama', 'allando\_nepessegbol\_14\_evesek\_szama', 'allando\_nepessegbol\_15\_17\_evesek\_szama', 'nappali\_altalanos\_iskolai\_tanulok\_szama', 'nappali\_kozepiskolai\_tanulok\_szama', 'nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama', 'mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama', 'mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'AllandoNepessegbol613EvesekSzama' => 2, 'AllandoNepessegbol14EvesekSzama' => 3, 'AllandoNepessegbol1517EvesekSzama' => 4, 'NappaliAltalanosIskolaiTanulokSzama' => 5, 'NappaliKozepiskolaiTanulokSzama' => 6, 'NappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama' => 7, 'MasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama' => 8, 'MasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama' => 9, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'allandoNepessegbol613EvesekSzama' => 2, 'allandoNepessegbol14EvesekSzama' => 3, 'allandoNepessegbol1517EvesekSzama' => 4, 'nappaliAltalanosIskolaiTanulokSzama' => 5, 'nappaliKozepiskolaiTanulokSzama' => 6, 'nappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama' => 7, 'masTelepulesrolBejaroNappaliAltIskolaiTanulokSzama' => 8, 'masTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama' => 9, ),

self::TYPE\_COLNAME => array(StgSzabadIskolavalasztasValtozoTableMap::COL\_TELEPULES\_NEV => 0, StgSzabadIskolavalasztasValtozoTableMap::COL\_TELEPULES\_KSHKOD => 1, StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA => 2, StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA => 3, StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA => 4, StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA => 5, StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA => 6, StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA => 7, StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA => 8, StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA => 9, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA' => 2, 'COL\_ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA' => 3, 'COL\_ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA' => 4, 'COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA' => 5, 'COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA' => 6, 'COL\_NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA' => 7, 'COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA' => 8, 'COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA' => 9, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'allando\_nepessegbol\_6\_13\_evesek\_szama' => 2, 'allando\_nepessegbol\_14\_evesek\_szama' => 3, 'allando\_nepessegbol\_15\_17\_evesek\_szama' => 4, 'nappali\_altalanos\_iskolai\_tanulok\_szama' => 5, 'nappali\_kozepiskolai\_tanulok\_szama' => 6, 'nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama' => 7, 'mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama' => 8, 'mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama' => 9, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_szabad\_iskolavalasztas\_valtozo');

$this->setPhpName('StgSzabadIskolavalasztasValtozo');

$this->setClassName('\\StgSzabadIskolavalasztasValtozo');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA', 'AllandoNepessegbol613EvesekSzama', 'INTEGER', false, null, null);

$this->addColumn('ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA', 'AllandoNepessegbol14EvesekSzama', 'INTEGER', false, null, null);

$this->addColumn('ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA', 'AllandoNepessegbol1517EvesekSzama', 'INTEGER', false, null, null);

$this->addColumn('NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA', 'NappaliAltalanosIskolaiTanulokSzama', 'INTEGER', false, null, null);

$this->addColumn('NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA', 'NappaliKozepiskolaiTanulokSzama', 'INTEGER', false, null, null);

$this->addColumn('NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA', 'NappaliSzakiskolaiEsSpecSzakiskolaiTanulokSzama', 'INTEGER', false, null, null);

$this->addColumn('MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA', 'MasTelepulesrolBejaroNappaliAltIskolaiTanulokSzama', 'INTEGER', false, null, null);

$this->addColumn('MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA', 'MasTelepulesrolBejaroNappaliKozepiskolaiTanulokSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgSzabadIskolavalasztasValtozoTableMap::CLASS\_DEFAULT : StgSzabadIskolavalasztasValtozoTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgSzabadIskolavalasztasValtozo object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgSzabadIskolavalasztasValtozoTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgSzabadIskolavalasztasValtozoTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgSzabadIskolavalasztasValtozoTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgSzabadIskolavalasztasValtozoTableMap::OM\_CLASS;

/\*\* @var StgSzabadIskolavalasztasValtozo $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgSzabadIskolavalasztasValtozoTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgSzabadIskolavalasztasValtozoTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgSzabadIskolavalasztasValtozoTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgSzabadIskolavalasztasValtozo $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgSzabadIskolavalasztasValtozoTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgSzabadIskolavalasztasValtozoTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgSzabadIskolavalasztasValtozoTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA);

$criteria->addSelectColumn(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA);

$criteria->addSelectColumn(StgSzabadIskolavalasztasValtozoTableMap::COL\_ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA);

$criteria->addSelectColumn(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA);

$criteria->addSelectColumn(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA);

$criteria->addSelectColumn(StgSzabadIskolavalasztasValtozoTableMap::COL\_NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA);

$criteria->addSelectColumn(StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA);

$criteria->addSelectColumn(StgSzabadIskolavalasztasValtozoTableMap::COL\_MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.ALLANDO\_NEPESSEGBOL\_6\_13\_EVESEK\_SZAMA');

$criteria->addSelectColumn($alias . '.ALLANDO\_NEPESSEGBOL\_14\_EVESEK\_SZAMA');

$criteria->addSelectColumn($alias . '.ALLANDO\_NEPESSEGBOL\_15\_17\_EVESEK\_SZAMA');

$criteria->addSelectColumn($alias . '.NAPPALI\_ALTALANOS\_ISKOLAI\_TANULOK\_SZAMA');

$criteria->addSelectColumn($alias . '.NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA');

$criteria->addSelectColumn($alias . '.NAPPALI\_SZAKISKOLAI\_ES\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA');

$criteria->addSelectColumn($alias . '.MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA');

$criteria->addSelectColumn($alias . '.MAS\_TELEPULESROL\_BEJARO\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgSzabadIskolavalasztasValtozoTableMap::DATABASE\_NAME)->getTable(StgSzabadIskolavalasztasValtozoTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgSzabadIskolavalasztasValtozoTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgSzabadIskolavalasztasValtozoTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgSzabadIskolavalasztasValtozoTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgSzabadIskolavalasztasValtozo or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgSzabadIskolavalasztasValtozo object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzabadIskolavalasztasValtozoTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgSzabadIskolavalasztasValtozo) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgSzabadIskolavalasztasValtozo object has no primary key');

}

$query = StgSzabadIskolavalasztasValtozoQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgSzabadIskolavalasztasValtozoTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgSzabadIskolavalasztasValtozoTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_szabad\_iskolavalasztas\_valtozo table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgSzabadIskolavalasztasValtozoQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgSzabadIskolavalasztasValtozo or Criteria object.

\*

\* @param mixed $criteria Criteria or StgSzabadIskolavalasztasValtozo object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzabadIskolavalasztasValtozoTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgSzabadIskolavalasztasValtozo object

}

// Set the correct dbName

$query = StgSzabadIskolavalasztasValtozoQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgSzabadIskolavalasztasValtozoTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgSzabadIskolavalasztasValtozoTableMap::buildTableMap();

### StgSzallashelyTableMap.php

<?php

namespace Map;

use \StgSzallashely;

use \StgSzallashelyQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_szallashely' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgSzallashelyTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgSzallashelyTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_szallashely';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgSzallashely';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgSzallashely';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 4;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 4;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_szallashely.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_szallashely.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA field

\*/

const COL\_FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA = 'stg\_szallashely.FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA';

/\*\*

\* the column name for the OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA field

\*/

const COL\_OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA = 'stg\_szallashely.OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'FizetovendeglatasSzallashelyeinekSzama', 'OsszesKereskedelmiSzallashelyFerohelyeinekSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'fizetovendeglatasSzallashelyeinekSzama', 'osszesKereskedelmiSzallashelyFerohelyeinekSzama', ),

self::TYPE\_COLNAME => array(StgSzallashelyTableMap::COL\_TELEPULES\_NEV, StgSzallashelyTableMap::COL\_TELEPULES\_KSHKOD, StgSzallashelyTableMap::COL\_FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA, StgSzallashelyTableMap::COL\_OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA', 'COL\_OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'fizetovendeglatas\_szallashelyeinek\_szama', 'osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'FizetovendeglatasSzallashelyeinekSzama' => 2, 'OsszesKereskedelmiSzallashelyFerohelyeinekSzama' => 3, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'fizetovendeglatasSzallashelyeinekSzama' => 2, 'osszesKereskedelmiSzallashelyFerohelyeinekSzama' => 3, ),

self::TYPE\_COLNAME => array(StgSzallashelyTableMap::COL\_TELEPULES\_NEV => 0, StgSzallashelyTableMap::COL\_TELEPULES\_KSHKOD => 1, StgSzallashelyTableMap::COL\_FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA => 2, StgSzallashelyTableMap::COL\_OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA => 3, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA' => 2, 'COL\_OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA' => 3, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'fizetovendeglatas\_szallashelyeinek\_szama' => 2, 'osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama' => 3, ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_szallashely');

$this->setPhpName('StgSzallashely');

$this->setClassName('\\StgSzallashely');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA', 'FizetovendeglatasSzallashelyeinekSzama', 'INTEGER', false, null, null);

$this->addColumn('OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA', 'OsszesKereskedelmiSzallashelyFerohelyeinekSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgSzallashelyTableMap::CLASS\_DEFAULT : StgSzallashelyTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgSzallashely object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgSzallashelyTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgSzallashelyTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgSzallashelyTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgSzallashelyTableMap::OM\_CLASS;

/\*\* @var StgSzallashely $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgSzallashelyTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgSzallashelyTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgSzallashelyTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgSzallashely $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgSzallashelyTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgSzallashelyTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgSzallashelyTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgSzallashelyTableMap::COL\_FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA);

$criteria->addSelectColumn(StgSzallashelyTableMap::COL\_OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.FIZETOVENDEGLATAS\_SZALLASHELYEINEK\_SZAMA');

$criteria->addSelectColumn($alias . '.OSSZES\_KERESKEDELMI\_SZALLASHELY\_FEROHELYEINEK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgSzallashelyTableMap::DATABASE\_NAME)->getTable(StgSzallashelyTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgSzallashelyTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgSzallashelyTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgSzallashelyTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgSzallashely or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgSzallashely object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzallashelyTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgSzallashely) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgSzallashely object has no primary key');

}

$query = StgSzallashelyQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgSzallashelyTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgSzallashelyTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_szallashely table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgSzallashelyQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgSzallashely or Criteria object.

\*

\* @param mixed $criteria Criteria or StgSzallashely object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzallashelyTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgSzallashely object

}

// Set the correct dbName

$query = StgSzallashelyQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgSzallashelyTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgSzallashelyTableMap::buildTableMap();

### StgSzemelygepkocsikTableMap.php

<?php

namespace Map;

use \StgSzemelygepkocsik;

use \StgSzemelygepkocsikQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_szemelygepkocsik' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgSzemelygepkocsikTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgSzemelygepkocsikTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_szemelygepkocsik';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgSzemelygepkocsik';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgSzemelygepkocsik';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 3;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 3;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_szemelygepkocsik.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_szemelygepkocsik.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the SZEMELYGEPKOCSIK\_SZAMA field

\*/

const COL\_SZEMELYGEPKOCSIK\_SZAMA = 'stg\_szemelygepkocsik.SZEMELYGEPKOCSIK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'SzemelygepkocsikSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'szemelygepkocsikSzama', ),

self::TYPE\_COLNAME => array(StgSzemelygepkocsikTableMap::COL\_TELEPULES\_NEV, StgSzemelygepkocsikTableMap::COL\_TELEPULES\_KSHKOD, StgSzemelygepkocsikTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_SZEMELYGEPKOCSIK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'szemelygepkocsik\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'SzemelygepkocsikSzama' => 2, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'szemelygepkocsikSzama' => 2, ),

self::TYPE\_COLNAME => array(StgSzemelygepkocsikTableMap::COL\_TELEPULES\_NEV => 0, StgSzemelygepkocsikTableMap::COL\_TELEPULES\_KSHKOD => 1, StgSzemelygepkocsikTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA => 2, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_SZEMELYGEPKOCSIK\_SZAMA' => 2, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'szemelygepkocsik\_szama' => 2, ),

self::TYPE\_NUM => array(0, 1, 2, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_szemelygepkocsik');

$this->setPhpName('StgSzemelygepkocsik');

$this->setClassName('\\StgSzemelygepkocsik');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('SZEMELYGEPKOCSIK\_SZAMA', 'SzemelygepkocsikSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgSzemelygepkocsikTableMap::CLASS\_DEFAULT : StgSzemelygepkocsikTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgSzemelygepkocsik object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgSzemelygepkocsikTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgSzemelygepkocsikTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgSzemelygepkocsikTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgSzemelygepkocsikTableMap::OM\_CLASS;

/\*\* @var StgSzemelygepkocsik $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgSzemelygepkocsikTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgSzemelygepkocsikTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgSzemelygepkocsikTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgSzemelygepkocsik $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgSzemelygepkocsikTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgSzemelygepkocsikTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgSzemelygepkocsikTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgSzemelygepkocsikTableMap::COL\_SZEMELYGEPKOCSIK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.SZEMELYGEPKOCSIK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgSzemelygepkocsikTableMap::DATABASE\_NAME)->getTable(StgSzemelygepkocsikTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgSzemelygepkocsikTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgSzemelygepkocsikTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgSzemelygepkocsikTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgSzemelygepkocsik or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgSzemelygepkocsik object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzemelygepkocsikTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgSzemelygepkocsik) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgSzemelygepkocsik object has no primary key');

}

$query = StgSzemelygepkocsikQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgSzemelygepkocsikTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgSzemelygepkocsikTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_szemelygepkocsik table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgSzemelygepkocsikQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgSzemelygepkocsik or Criteria object.

\*

\* @param mixed $criteria Criteria or StgSzemelygepkocsik object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgSzemelygepkocsikTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgSzemelygepkocsik object

}

// Set the correct dbName

$query = StgSzemelygepkocsikQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgSzemelygepkocsikTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgSzemelygepkocsikTableMap::buildTableMap();

### StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap.php

<?php

namespace Map;

use \StgTanulasiCeluUtazasokAltalanosJellemzese;

use \StgTanulasiCeluUtazasokAltalanosJellemzeseQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgTanulasiCeluUtazasokAltalanosJellemzese';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgTanulasiCeluUtazasokAltalanosJellemzese';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 6;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 6;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS field

\*/

const COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS = 'stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese.NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS';

/\*\*

\* the column name for the NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS field

\*/

const COL\_NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS = 'stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese.NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS';

/\*\*

\* the column name for the FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA field

\*/

const COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA = 'stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese.FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA';

/\*\*

\* the column name for the NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS field

\*/

const COL\_NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS = 'stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese.NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'NappaliAltalanosIskolaiOktatas', 'NappaliKozepiskolaiEsFelnottoktatasiOktatas', 'FelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama', 'NappaliEsNemNappaliFelsooktatas', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'nappaliAltalanosIskolaiOktatas', 'nappaliKozepiskolaiEsFelnottoktatasiOktatas', 'felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama', 'nappaliEsNemNappaliFelsooktatas', ),

self::TYPE\_COLNAME => array(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_TELEPULES\_NEV, StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_TELEPULES\_KSHKOD, StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS, StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS, StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS', 'COL\_NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS', 'COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA', 'COL\_NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'nappali\_altalanos\_iskolai\_oktatas', 'nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas', 'felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama', 'nappali\_es\_nem\_nappali\_felsooktatas', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'NappaliAltalanosIskolaiOktatas' => 2, 'NappaliKozepiskolaiEsFelnottoktatasiOktatas' => 3, 'FelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama' => 4, 'NappaliEsNemNappaliFelsooktatas' => 5, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'nappaliAltalanosIskolaiOktatas' => 2, 'nappaliKozepiskolaiEsFelnottoktatasiOktatas' => 3, 'felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama' => 4, 'nappaliEsNemNappaliFelsooktatas' => 5, ),

self::TYPE\_COLNAME => array(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_TELEPULES\_NEV => 0, StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_TELEPULES\_KSHKOD => 1, StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS => 2, StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS => 3, StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA => 4, StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS => 5, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS' => 2, 'COL\_NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS' => 3, 'COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA' => 4, 'COL\_NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS' => 5, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'nappali\_altalanos\_iskolai\_oktatas' => 2, 'nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas' => 3, 'felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama' => 4, 'nappali\_es\_nem\_nappali\_felsooktatas' => 5, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese');

$this->setPhpName('StgTanulasiCeluUtazasokAltalanosJellemzese');

$this->setClassName('\\StgTanulasiCeluUtazasokAltalanosJellemzese');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS', 'NappaliAltalanosIskolaiOktatas', 'INTEGER', false, null, null);

$this->addColumn('NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS', 'NappaliKozepiskolaiEsFelnottoktatasiOktatas', 'INTEGER', false, null, null);

$this->addColumn('FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA', 'FelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama', 'INTEGER', false, null, null);

$this->addColumn('NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS', 'NappaliEsNemNappaliFelsooktatas', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::CLASS\_DEFAULT : StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgTanulasiCeluUtazasokAltalanosJellemzese object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::OM\_CLASS;

/\*\* @var StgTanulasiCeluUtazasokAltalanosJellemzese $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgTanulasiCeluUtazasokAltalanosJellemzese $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS);

$criteria->addSelectColumn(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS);

$criteria->addSelectColumn(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA);

$criteria->addSelectColumn(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::COL\_NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.NAPPALI\_ALTALANOS\_ISKOLAI\_OKTATAS');

$criteria->addSelectColumn($alias . '.NAPPALI\_KOZEPISKOLAI\_ES\_FELNOTTOKTATASI\_OKTATAS');

$criteria->addSelectColumn($alias . '.FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA');

$criteria->addSelectColumn($alias . '.NAPPALI\_ES\_NEM\_NAPPALI\_FELSOOKTATAS');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::DATABASE\_NAME)->getTable(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgTanulasiCeluUtazasokAltalanosJellemzese or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgTanulasiCeluUtazasokAltalanosJellemzese object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgTanulasiCeluUtazasokAltalanosJellemzese) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgTanulasiCeluUtazasokAltalanosJellemzese object has no primary key');

}

$query = StgTanulasiCeluUtazasokAltalanosJellemzeseQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgTanulasiCeluUtazasokAltalanosJellemzeseQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgTanulasiCeluUtazasokAltalanosJellemzese or Criteria object.

\*

\* @param mixed $criteria Criteria or StgTanulasiCeluUtazasokAltalanosJellemzese object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgTanulasiCeluUtazasokAltalanosJellemzese object

}

// Set the correct dbName

$query = StgTanulasiCeluUtazasokAltalanosJellemzeseQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgTanulasiCeluUtazasokAltalanosJellemzeseTableMap::buildTableMap();

### StgTanulokEsHallgatokSzamaTableMap.php

<?php

namespace Map;

use \StgTanulokEsHallgatokSzama;

use \StgTanulokEsHallgatokSzamaQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_tanulok\_es\_hallgatok\_szama' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgTanulokEsHallgatokSzamaTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgTanulokEsHallgatokSzamaTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_tanulok\_es\_hallgatok\_szama';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgTanulokEsHallgatokSzama';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgTanulokEsHallgatokSzama';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 10;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 10;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_tanulok\_es\_hallgatok\_szama.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_tanulok\_es\_hallgatok\_szama.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA field

\*/

const COL\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA = 'stg\_tanulok\_es\_hallgatok\_szama.NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA';

/\*\*

\* the column name for the FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA field

\*/

const COL\_FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA = 'stg\_tanulok\_es\_hallgatok\_szama.FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA';

/\*\*

\* the column name for the NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA field

\*/

const COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA = 'stg\_tanulok\_es\_hallgatok\_szama.NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA';

/\*\*

\* the column name for the FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA field

\*/

const COL\_FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA = 'stg\_tanulok\_es\_hallgatok\_szama.FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA';

/\*\*

\* the column name for the NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA field

\*/

const COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA = 'stg\_tanulok\_es\_hallgatok\_szama.NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA';

/\*\*

\* the column name for the FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA field

\*/

const COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA = 'stg\_tanulok\_es\_hallgatok\_szama.FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA';

/\*\*

\* the column name for the NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA field

\*/

const COL\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA = 'stg\_tanulok\_es\_hallgatok\_szama.NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA';

/\*\*

\* the column name for the NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA field

\*/

const COL\_NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA = 'stg\_tanulok\_es\_hallgatok\_szama.NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'NappaliAltIskolaiTanulokSzama', 'FelnottoktatasbanAltIskolaiTanulokSzama', 'NappaliKozepiskolaiTanulokSzama', 'FelnottoktatasbanKozepiskolaiTanulokSzama', 'NappaliSzakiskolaiSpecSzakiskolaiTanulokSzama', 'FelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama', 'NappaliFelsofokuAlapMesterkepzesbenHallgatokSzama', 'NemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'nappaliAltIskolaiTanulokSzama', 'felnottoktatasbanAltIskolaiTanulokSzama', 'nappaliKozepiskolaiTanulokSzama', 'felnottoktatasbanKozepiskolaiTanulokSzama', 'nappaliSzakiskolaiSpecSzakiskolaiTanulokSzama', 'felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama', 'nappaliFelsofokuAlapMesterkepzesbenHallgatokSzama', 'nemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama', ),

self::TYPE\_COLNAME => array(StgTanulokEsHallgatokSzamaTableMap::COL\_TELEPULES\_NEV, StgTanulokEsHallgatokSzamaTableMap::COL\_TELEPULES\_KSHKOD, StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA, StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA, StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA, StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA, StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA, StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA, StgTanulokEsHallgatokSzamaTableMap::COL\_NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA', 'COL\_FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA', 'COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA', 'COL\_FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA', 'COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA', 'COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA', 'COL\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA', 'COL\_NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'nappali\_alt\_iskolai\_tanulok\_szama', 'felnottoktatasban\_alt\_iskolai\_tanulok\_szama', 'nappali\_kozepiskolai\_tanulok\_szama', 'felnottoktatasban\_kozepiskolai\_tanulok\_szama', 'nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama', 'felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama', 'nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama', 'nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'NappaliAltIskolaiTanulokSzama' => 2, 'FelnottoktatasbanAltIskolaiTanulokSzama' => 3, 'NappaliKozepiskolaiTanulokSzama' => 4, 'FelnottoktatasbanKozepiskolaiTanulokSzama' => 5, 'NappaliSzakiskolaiSpecSzakiskolaiTanulokSzama' => 6, 'FelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama' => 7, 'NappaliFelsofokuAlapMesterkepzesbenHallgatokSzama' => 8, 'NemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama' => 9, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'nappaliAltIskolaiTanulokSzama' => 2, 'felnottoktatasbanAltIskolaiTanulokSzama' => 3, 'nappaliKozepiskolaiTanulokSzama' => 4, 'felnottoktatasbanKozepiskolaiTanulokSzama' => 5, 'nappaliSzakiskolaiSpecSzakiskolaiTanulokSzama' => 6, 'felnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama' => 7, 'nappaliFelsofokuAlapMesterkepzesbenHallgatokSzama' => 8, 'nemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama' => 9, ),

self::TYPE\_COLNAME => array(StgTanulokEsHallgatokSzamaTableMap::COL\_TELEPULES\_NEV => 0, StgTanulokEsHallgatokSzamaTableMap::COL\_TELEPULES\_KSHKOD => 1, StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA => 2, StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA => 3, StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA => 4, StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA => 5, StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA => 6, StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA => 7, StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA => 8, StgTanulokEsHallgatokSzamaTableMap::COL\_NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA => 9, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA' => 2, 'COL\_FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA' => 3, 'COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA' => 4, 'COL\_FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA' => 5, 'COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA' => 6, 'COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA' => 7, 'COL\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA' => 8, 'COL\_NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA' => 9, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'nappali\_alt\_iskolai\_tanulok\_szama' => 2, 'felnottoktatasban\_alt\_iskolai\_tanulok\_szama' => 3, 'nappali\_kozepiskolai\_tanulok\_szama' => 4, 'felnottoktatasban\_kozepiskolai\_tanulok\_szama' => 5, 'nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama' => 6, 'felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama' => 7, 'nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama' => 8, 'nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama' => 9, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_tanulok\_es\_hallgatok\_szama');

$this->setPhpName('StgTanulokEsHallgatokSzama');

$this->setClassName('\\StgTanulokEsHallgatokSzama');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA', 'NappaliAltIskolaiTanulokSzama', 'INTEGER', false, null, null);

$this->addColumn('FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA', 'FelnottoktatasbanAltIskolaiTanulokSzama', 'INTEGER', false, null, null);

$this->addColumn('NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA', 'NappaliKozepiskolaiTanulokSzama', 'INTEGER', false, null, null);

$this->addColumn('FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA', 'FelnottoktatasbanKozepiskolaiTanulokSzama', 'INTEGER', false, null, null);

$this->addColumn('NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA', 'NappaliSzakiskolaiSpecSzakiskolaiTanulokSzama', 'INTEGER', false, null, null);

$this->addColumn('FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA', 'FelnottoktatasbanSzakiskolaiSpecSzakiskolaiTanulokSzama', 'INTEGER', false, null, null);

$this->addColumn('NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA', 'NappaliFelsofokuAlapMesterkepzesbenHallgatokSzama', 'INTEGER', false, null, null);

$this->addColumn('NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA', 'NemNappaliFelsofokuAlapMesterkepzesbenHallgatokSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgTanulokEsHallgatokSzamaTableMap::CLASS\_DEFAULT : StgTanulokEsHallgatokSzamaTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgTanulokEsHallgatokSzama object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgTanulokEsHallgatokSzamaTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgTanulokEsHallgatokSzamaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgTanulokEsHallgatokSzamaTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgTanulokEsHallgatokSzamaTableMap::OM\_CLASS;

/\*\* @var StgTanulokEsHallgatokSzama $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgTanulokEsHallgatokSzamaTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgTanulokEsHallgatokSzamaTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgTanulokEsHallgatokSzamaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgTanulokEsHallgatokSzama $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgTanulokEsHallgatokSzamaTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgTanulokEsHallgatokSzamaTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgTanulokEsHallgatokSzamaTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA);

$criteria->addSelectColumn(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA);

$criteria->addSelectColumn(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA);

$criteria->addSelectColumn(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA);

$criteria->addSelectColumn(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA);

$criteria->addSelectColumn(StgTanulokEsHallgatokSzamaTableMap::COL\_FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA);

$criteria->addSelectColumn(StgTanulokEsHallgatokSzamaTableMap::COL\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA);

$criteria->addSelectColumn(StgTanulokEsHallgatokSzamaTableMap::COL\_NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.NAPPALI\_ALT\_ISKOLAI\_TANULOK\_SZAMA');

$criteria->addSelectColumn($alias . '.FELNOTTOKTATASBAN\_ALT\_ISKOLAI\_TANULOK\_SZAMA');

$criteria->addSelectColumn($alias . '.NAPPALI\_KOZEPISKOLAI\_TANULOK\_SZAMA');

$criteria->addSelectColumn($alias . '.FELNOTTOKTATASBAN\_KOZEPISKOLAI\_TANULOK\_SZAMA');

$criteria->addSelectColumn($alias . '.NAPPALI\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA');

$criteria->addSelectColumn($alias . '.FELNOTTOKTATASBAN\_SZAKISKOLAI\_SPEC\_SZAKISKOLAI\_TANULOK\_SZAMA');

$criteria->addSelectColumn($alias . '.NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA');

$criteria->addSelectColumn($alias . '.NEM\_NAPPALI\_FELSOFOKU\_ALAP\_MESTERKEPZESBEN\_HALLGATOK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgTanulokEsHallgatokSzamaTableMap::DATABASE\_NAME)->getTable(StgTanulokEsHallgatokSzamaTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgTanulokEsHallgatokSzamaTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgTanulokEsHallgatokSzamaTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgTanulokEsHallgatokSzamaTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgTanulokEsHallgatokSzama or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgTanulokEsHallgatokSzama object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTanulokEsHallgatokSzamaTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgTanulokEsHallgatokSzama) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgTanulokEsHallgatokSzama object has no primary key');

}

$query = StgTanulokEsHallgatokSzamaQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgTanulokEsHallgatokSzamaTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgTanulokEsHallgatokSzamaTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_tanulok\_es\_hallgatok\_szama table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgTanulokEsHallgatokSzamaQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgTanulokEsHallgatokSzama or Criteria object.

\*

\* @param mixed $criteria Criteria or StgTanulokEsHallgatokSzama object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTanulokEsHallgatokSzamaTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgTanulokEsHallgatokSzama object

}

// Set the correct dbName

$query = StgTanulokEsHallgatokSzamaQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgTanulokEsHallgatokSzamaTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgTanulokEsHallgatokSzamaTableMap::buildTableMap();

### StgTelepulesKoordinataTableMap.php

<?php

namespace Map;

use \StgTelepulesKoordinata;

use \StgTelepulesKoordinataQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_telepules\_koordinata' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgTelepulesKoordinataTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgTelepulesKoordinataTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_telepules\_koordinata';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgTelepulesKoordinata';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgTelepulesKoordinata';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 5;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 5;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_telepules\_koordinata.TELEPULES\_NEV';

/\*\*

\* the column name for the EOV\_X field

\*/

const COL\_EOV\_X = 'stg\_telepules\_koordinata.EOV\_X';

/\*\*

\* the column name for the EOV\_Y field

\*/

const COL\_EOV\_Y = 'stg\_telepules\_koordinata.EOV\_Y';

/\*\*

\* the column name for the WGS84\_LAT field

\*/

const COL\_WGS84\_LAT = 'stg\_telepules\_koordinata.WGS84\_LAT';

/\*\*

\* the column name for the WGS84\_LOB field

\*/

const COL\_WGS84\_LOB = 'stg\_telepules\_koordinata.WGS84\_LOB';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'EovX', 'EovY', 'Wgs84Lat', 'Wgs84Lob', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'eovX', 'eovY', 'wgs84Lat', 'wgs84Lob', ),

self::TYPE\_COLNAME => array(StgTelepulesKoordinataTableMap::COL\_TELEPULES\_NEV, StgTelepulesKoordinataTableMap::COL\_EOV\_X, StgTelepulesKoordinataTableMap::COL\_EOV\_Y, StgTelepulesKoordinataTableMap::COL\_WGS84\_LAT, StgTelepulesKoordinataTableMap::COL\_WGS84\_LOB, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_EOV\_X', 'COL\_EOV\_Y', 'COL\_WGS84\_LAT', 'COL\_WGS84\_LOB', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'EOV\_X', 'EOV\_Y', 'WGS84\_lat', 'WGS84\_lob', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'EovX' => 1, 'EovY' => 2, 'Wgs84Lat' => 3, 'Wgs84Lob' => 4, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'eovX' => 1, 'eovY' => 2, 'wgs84Lat' => 3, 'wgs84Lob' => 4, ),

self::TYPE\_COLNAME => array(StgTelepulesKoordinataTableMap::COL\_TELEPULES\_NEV => 0, StgTelepulesKoordinataTableMap::COL\_EOV\_X => 1, StgTelepulesKoordinataTableMap::COL\_EOV\_Y => 2, StgTelepulesKoordinataTableMap::COL\_WGS84\_LAT => 3, StgTelepulesKoordinataTableMap::COL\_WGS84\_LOB => 4, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_EOV\_X' => 1, 'COL\_EOV\_Y' => 2, 'COL\_WGS84\_LAT' => 3, 'COL\_WGS84\_LOB' => 4, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'EOV\_X' => 1, 'EOV\_Y' => 2, 'WGS84\_lat' => 3, 'WGS84\_lob' => 4, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_telepules\_koordinata');

$this->setPhpName('StgTelepulesKoordinata');

$this->setClassName('\\StgTelepulesKoordinata');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addPrimaryKey('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 100, null);

$this->addColumn('EOV\_X', 'EovX', 'VARCHAR', false, 45, null);

$this->addColumn('EOV\_Y', 'EovY', 'VARCHAR', false, 45, null);

$this->addColumn('WGS84\_LAT', 'Wgs84Lat', 'VARCHAR', false, 45, null);

$this->addColumn('WGS84\_LOB', 'Wgs84Lob', 'VARCHAR', false, 45, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

// If the PK cannot be derived from the row, return NULL.

if ($row[TableMap::TYPE\_NUM == $indexType ? 0 + $offset : static::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)] === null) {

return null;

}

return (string) $row[TableMap::TYPE\_NUM == $indexType ? 0 + $offset : static::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return (string) $row[

$indexType == TableMap::TYPE\_NUM

? 0 + $offset

: self::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)

];

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgTelepulesKoordinataTableMap::CLASS\_DEFAULT : StgTelepulesKoordinataTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgTelepulesKoordinata object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgTelepulesKoordinataTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgTelepulesKoordinataTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgTelepulesKoordinataTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgTelepulesKoordinataTableMap::OM\_CLASS;

/\*\* @var StgTelepulesKoordinata $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgTelepulesKoordinataTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgTelepulesKoordinataTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgTelepulesKoordinataTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgTelepulesKoordinata $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgTelepulesKoordinataTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgTelepulesKoordinataTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgTelepulesKoordinataTableMap::COL\_EOV\_X);

$criteria->addSelectColumn(StgTelepulesKoordinataTableMap::COL\_EOV\_Y);

$criteria->addSelectColumn(StgTelepulesKoordinataTableMap::COL\_WGS84\_LAT);

$criteria->addSelectColumn(StgTelepulesKoordinataTableMap::COL\_WGS84\_LOB);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.EOV\_X');

$criteria->addSelectColumn($alias . '.EOV\_Y');

$criteria->addSelectColumn($alias . '.WGS84\_LAT');

$criteria->addSelectColumn($alias . '.WGS84\_LOB');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgTelepulesKoordinataTableMap::DATABASE\_NAME)->getTable(StgTelepulesKoordinataTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgTelepulesKoordinataTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgTelepulesKoordinataTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgTelepulesKoordinataTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgTelepulesKoordinata or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgTelepulesKoordinata object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTelepulesKoordinataTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgTelepulesKoordinata) { // it's a model object

// create criteria based on pk values

$criteria = $values->buildPkeyCriteria();

} else { // it's a primary key, or an array of pks

$criteria = new Criteria(StgTelepulesKoordinataTableMap::DATABASE\_NAME);

$criteria->add(StgTelepulesKoordinataTableMap::COL\_TELEPULES\_NEV, (array) $values, Criteria::IN);

}

$query = StgTelepulesKoordinataQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgTelepulesKoordinataTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgTelepulesKoordinataTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_telepules\_koordinata table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgTelepulesKoordinataQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgTelepulesKoordinata or Criteria object.

\*

\* @param mixed $criteria Criteria or StgTelepulesKoordinata object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTelepulesKoordinataTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgTelepulesKoordinata object

}

// Set the correct dbName

$query = StgTelepulesKoordinataQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgTelepulesKoordinataTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgTelepulesKoordinataTableMap::buildTableMap();

### StgTelepulesTableMap.php

<?php

namespace Map;

use \StgTelepules;

use \StgTelepulesQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_telepules' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgTelepulesTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgTelepulesTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_telepules';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgTelepules';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgTelepules';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 11;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 11;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_telepules.TELEPULES\_NEV';

/\*\*

\* the column name for the KSH\_KOD field

\*/

const COL\_KSH\_KOD = 'stg\_telepules.KSH\_KOD';

/\*\*

\* the column name for the JOGALLAS field

\*/

const COL\_JOGALLAS = 'stg\_telepules.JOGALLAS';

/\*\*

\* the column name for the MEGYE field

\*/

const COL\_MEGYE = 'stg\_telepules.MEGYE';

/\*\*

\* the column name for the KISTERSEG\_KOD field

\*/

const COL\_KISTERSEG\_KOD = 'stg\_telepules.KISTERSEG\_KOD';

/\*\*

\* the column name for the KISTERSEG\_NEV field

\*/

const COL\_KISTERSEG\_NEV = 'stg\_telepules.KISTERSEG\_NEV';

/\*\*

\* the column name for the KISTERSEG\_SZEKHELYE field

\*/

const COL\_KISTERSEG\_SZEKHELYE = 'stg\_telepules.KISTERSEG\_SZEKHELYE';

/\*\*

\* the column name for the EOV\_X field

\*/

const COL\_EOV\_X = 'stg\_telepules.EOV\_X';

/\*\*

\* the column name for the EOV\_Y field

\*/

const COL\_EOV\_Y = 'stg\_telepules.EOV\_Y';

/\*\*

\* the column name for the WGS84\_LAT field

\*/

const COL\_WGS84\_LAT = 'stg\_telepules.WGS84\_LAT';

/\*\*

\* the column name for the WGS84\_LOB field

\*/

const COL\_WGS84\_LOB = 'stg\_telepules.WGS84\_LOB';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'KshKod', 'Jogallas', 'Megye', 'KistersegKod', 'KistersegNev', 'KistersegSzekhelye', 'EovX', 'EovY', 'Wgs84Lat', 'Wgs84Lob', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'kshKod', 'jogallas', 'megye', 'kistersegKod', 'kistersegNev', 'kistersegSzekhelye', 'eovX', 'eovY', 'wgs84Lat', 'wgs84Lob', ),

self::TYPE\_COLNAME => array(StgTelepulesTableMap::COL\_TELEPULES\_NEV, StgTelepulesTableMap::COL\_KSH\_KOD, StgTelepulesTableMap::COL\_JOGALLAS, StgTelepulesTableMap::COL\_MEGYE, StgTelepulesTableMap::COL\_KISTERSEG\_KOD, StgTelepulesTableMap::COL\_KISTERSEG\_NEV, StgTelepulesTableMap::COL\_KISTERSEG\_SZEKHELYE, StgTelepulesTableMap::COL\_EOV\_X, StgTelepulesTableMap::COL\_EOV\_Y, StgTelepulesTableMap::COL\_WGS84\_LAT, StgTelepulesTableMap::COL\_WGS84\_LOB, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_KSH\_KOD', 'COL\_JOGALLAS', 'COL\_MEGYE', 'COL\_KISTERSEG\_KOD', 'COL\_KISTERSEG\_NEV', 'COL\_KISTERSEG\_SZEKHELYE', 'COL\_EOV\_X', 'COL\_EOV\_Y', 'COL\_WGS84\_LAT', 'COL\_WGS84\_LOB', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'KSH\_kod', 'jogallas', 'megye', 'kisterseg\_kod', 'kisterseg\_nev', 'kisterseg\_szekhelye', 'EOV\_X', 'EOV\_Y', 'WGS84\_lat', 'WGS84\_lob', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'KshKod' => 1, 'Jogallas' => 2, 'Megye' => 3, 'KistersegKod' => 4, 'KistersegNev' => 5, 'KistersegSzekhelye' => 6, 'EovX' => 7, 'EovY' => 8, 'Wgs84Lat' => 9, 'Wgs84Lob' => 10, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'kshKod' => 1, 'jogallas' => 2, 'megye' => 3, 'kistersegKod' => 4, 'kistersegNev' => 5, 'kistersegSzekhelye' => 6, 'eovX' => 7, 'eovY' => 8, 'wgs84Lat' => 9, 'wgs84Lob' => 10, ),

self::TYPE\_COLNAME => array(StgTelepulesTableMap::COL\_TELEPULES\_NEV => 0, StgTelepulesTableMap::COL\_KSH\_KOD => 1, StgTelepulesTableMap::COL\_JOGALLAS => 2, StgTelepulesTableMap::COL\_MEGYE => 3, StgTelepulesTableMap::COL\_KISTERSEG\_KOD => 4, StgTelepulesTableMap::COL\_KISTERSEG\_NEV => 5, StgTelepulesTableMap::COL\_KISTERSEG\_SZEKHELYE => 6, StgTelepulesTableMap::COL\_EOV\_X => 7, StgTelepulesTableMap::COL\_EOV\_Y => 8, StgTelepulesTableMap::COL\_WGS84\_LAT => 9, StgTelepulesTableMap::COL\_WGS84\_LOB => 10, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_KSH\_KOD' => 1, 'COL\_JOGALLAS' => 2, 'COL\_MEGYE' => 3, 'COL\_KISTERSEG\_KOD' => 4, 'COL\_KISTERSEG\_NEV' => 5, 'COL\_KISTERSEG\_SZEKHELYE' => 6, 'COL\_EOV\_X' => 7, 'COL\_EOV\_Y' => 8, 'COL\_WGS84\_LAT' => 9, 'COL\_WGS84\_LOB' => 10, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'KSH\_kod' => 1, 'jogallas' => 2, 'megye' => 3, 'kisterseg\_kod' => 4, 'kisterseg\_nev' => 5, 'kisterseg\_szekhelye' => 6, 'EOV\_X' => 7, 'EOV\_Y' => 8, 'WGS84\_lat' => 9, 'WGS84\_lob' => 10, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_telepules');

$this->setPhpName('StgTelepules');

$this->setClassName('\\StgTelepules');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addPrimaryKey('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 100, null);

$this->addColumn('KSH\_KOD', 'KshKod', 'VARCHAR', false, 45, null);

$this->addColumn('JOGALLAS', 'Jogallas', 'VARCHAR', false, 45, null);

$this->addColumn('MEGYE', 'Megye', 'VARCHAR', false, 45, null);

$this->addColumn('KISTERSEG\_KOD', 'KistersegKod', 'VARCHAR', false, 45, null);

$this->addColumn('KISTERSEG\_NEV', 'KistersegNev', 'VARCHAR', false, 45, null);

$this->addColumn('KISTERSEG\_SZEKHELYE', 'KistersegSzekhelye', 'VARCHAR', false, 45, null);

$this->addColumn('EOV\_X', 'EovX', 'VARCHAR', false, 45, null);

$this->addColumn('EOV\_Y', 'EovY', 'VARCHAR', false, 45, null);

$this->addColumn('WGS84\_LAT', 'Wgs84Lat', 'VARCHAR', false, 45, null);

$this->addColumn('WGS84\_LOB', 'Wgs84Lob', 'VARCHAR', false, 45, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

// If the PK cannot be derived from the row, return NULL.

if ($row[TableMap::TYPE\_NUM == $indexType ? 0 + $offset : static::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)] === null) {

return null;

}

return (string) $row[TableMap::TYPE\_NUM == $indexType ? 0 + $offset : static::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)];

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return (string) $row[

$indexType == TableMap::TYPE\_NUM

? 0 + $offset

: self::translateFieldName('TelepulesNev', TableMap::TYPE\_PHPNAME, $indexType)

];

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgTelepulesTableMap::CLASS\_DEFAULT : StgTelepulesTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgTelepules object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgTelepulesTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgTelepulesTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgTelepulesTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgTelepulesTableMap::OM\_CLASS;

/\*\* @var StgTelepules $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgTelepulesTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgTelepulesTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgTelepulesTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgTelepules $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgTelepulesTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgTelepulesTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgTelepulesTableMap::COL\_KSH\_KOD);

$criteria->addSelectColumn(StgTelepulesTableMap::COL\_JOGALLAS);

$criteria->addSelectColumn(StgTelepulesTableMap::COL\_MEGYE);

$criteria->addSelectColumn(StgTelepulesTableMap::COL\_KISTERSEG\_KOD);

$criteria->addSelectColumn(StgTelepulesTableMap::COL\_KISTERSEG\_NEV);

$criteria->addSelectColumn(StgTelepulesTableMap::COL\_KISTERSEG\_SZEKHELYE);

$criteria->addSelectColumn(StgTelepulesTableMap::COL\_EOV\_X);

$criteria->addSelectColumn(StgTelepulesTableMap::COL\_EOV\_Y);

$criteria->addSelectColumn(StgTelepulesTableMap::COL\_WGS84\_LAT);

$criteria->addSelectColumn(StgTelepulesTableMap::COL\_WGS84\_LOB);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.KSH\_KOD');

$criteria->addSelectColumn($alias . '.JOGALLAS');

$criteria->addSelectColumn($alias . '.MEGYE');

$criteria->addSelectColumn($alias . '.KISTERSEG\_KOD');

$criteria->addSelectColumn($alias . '.KISTERSEG\_NEV');

$criteria->addSelectColumn($alias . '.KISTERSEG\_SZEKHELYE');

$criteria->addSelectColumn($alias . '.EOV\_X');

$criteria->addSelectColumn($alias . '.EOV\_Y');

$criteria->addSelectColumn($alias . '.WGS84\_LAT');

$criteria->addSelectColumn($alias . '.WGS84\_LOB');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgTelepulesTableMap::DATABASE\_NAME)->getTable(StgTelepulesTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgTelepulesTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgTelepulesTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgTelepulesTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgTelepules or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgTelepules object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTelepulesTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgTelepules) { // it's a model object

// create criteria based on pk values

$criteria = $values->buildPkeyCriteria();

} else { // it's a primary key, or an array of pks

$criteria = new Criteria(StgTelepulesTableMap::DATABASE\_NAME);

$criteria->add(StgTelepulesTableMap::COL\_TELEPULES\_NEV, (array) $values, Criteria::IN);

}

$query = StgTelepulesQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgTelepulesTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgTelepulesTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_telepules table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgTelepulesQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgTelepules or Criteria object.

\*

\* @param mixed $criteria Criteria or StgTelepules object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTelepulesTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgTelepules object

}

// Set the correct dbName

$query = StgTelepulesQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgTelepulesTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgTelepulesTableMap::buildTableMap();

### StgTomegkozlekedesTableMap.php

<?php

namespace Map;

use \StgTomegkozlekedes;

use \StgTomegkozlekedesQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_tomegkozlekedes' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgTomegkozlekedesTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgTomegkozlekedesTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_tomegkozlekedes';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgTomegkozlekedes';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgTomegkozlekedes';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 4;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 4;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_tomegkozlekedes.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_tomegkozlekedes.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the VASUTALLOMAS\_LETE field

\*/

const COL\_VASUTALLOMAS\_LETE = 'stg\_tomegkozlekedes.VASUTALLOMAS\_LETE';

/\*\*

\* the column name for the KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE field

\*/

const COL\_KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE = 'stg\_tomegkozlekedes.KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'VasutallomasLete', 'KozvetlenJaratokSzamaNapontaAMegyeszekhelyre', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'vasutallomasLete', 'kozvetlenJaratokSzamaNapontaAMegyeszekhelyre', ),

self::TYPE\_COLNAME => array(StgTomegkozlekedesTableMap::COL\_TELEPULES\_NEV, StgTomegkozlekedesTableMap::COL\_TELEPULES\_KSHKOD, StgTomegkozlekedesTableMap::COL\_VASUTALLOMAS\_LETE, StgTomegkozlekedesTableMap::COL\_KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_VASUTALLOMAS\_LETE', 'COL\_KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'vasutallomas\_lete', 'kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre', ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'VasutallomasLete' => 2, 'KozvetlenJaratokSzamaNapontaAMegyeszekhelyre' => 3, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'vasutallomasLete' => 2, 'kozvetlenJaratokSzamaNapontaAMegyeszekhelyre' => 3, ),

self::TYPE\_COLNAME => array(StgTomegkozlekedesTableMap::COL\_TELEPULES\_NEV => 0, StgTomegkozlekedesTableMap::COL\_TELEPULES\_KSHKOD => 1, StgTomegkozlekedesTableMap::COL\_VASUTALLOMAS\_LETE => 2, StgTomegkozlekedesTableMap::COL\_KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE => 3, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_VASUTALLOMAS\_LETE' => 2, 'COL\_KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE' => 3, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'vasutallomas\_lete' => 2, 'kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre' => 3, ),

self::TYPE\_NUM => array(0, 1, 2, 3, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_tomegkozlekedes');

$this->setPhpName('StgTomegkozlekedes');

$this->setClassName('\\StgTomegkozlekedes');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('VASUTALLOMAS\_LETE', 'VasutallomasLete', 'BOOLEAN', false, 1, null);

$this->addColumn('KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE', 'KozvetlenJaratokSzamaNapontaAMegyeszekhelyre', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgTomegkozlekedesTableMap::CLASS\_DEFAULT : StgTomegkozlekedesTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgTomegkozlekedes object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgTomegkozlekedesTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgTomegkozlekedesTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgTomegkozlekedesTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgTomegkozlekedesTableMap::OM\_CLASS;

/\*\* @var StgTomegkozlekedes $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgTomegkozlekedesTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgTomegkozlekedesTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgTomegkozlekedesTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgTomegkozlekedes $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgTomegkozlekedesTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgTomegkozlekedesTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgTomegkozlekedesTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgTomegkozlekedesTableMap::COL\_VASUTALLOMAS\_LETE);

$criteria->addSelectColumn(StgTomegkozlekedesTableMap::COL\_KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.VASUTALLOMAS\_LETE');

$criteria->addSelectColumn($alias . '.KOZVETLEN\_JARATOK\_SZAMA\_NAPONTA\_A\_MEGYESZEKHELYRE');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgTomegkozlekedesTableMap::DATABASE\_NAME)->getTable(StgTomegkozlekedesTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgTomegkozlekedesTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgTomegkozlekedesTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgTomegkozlekedesTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgTomegkozlekedes or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgTomegkozlekedes object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTomegkozlekedesTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgTomegkozlekedes) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgTomegkozlekedes object has no primary key');

}

$query = StgTomegkozlekedesQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgTomegkozlekedesTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgTomegkozlekedesTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_tomegkozlekedes table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgTomegkozlekedesQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgTomegkozlekedes or Criteria object.

\*

\* @param mixed $criteria Criteria or StgTomegkozlekedes object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgTomegkozlekedesTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgTomegkozlekedes object

}

// Set the correct dbName

$query = StgTomegkozlekedesQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgTomegkozlekedesTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgTomegkozlekedesTableMap::buildTableMap();

### StgUgyintezesAltalanosJellemzeseTableMap.php

<?php

namespace Map;

use \StgUgyintezesAltalanosJellemzese;

use \StgUgyintezesAltalanosJellemzeseQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_ugyintezes\_altalanos\_jellemzese' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgUgyintezesAltalanosJellemzeseTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgUgyintezesAltalanosJellemzeseTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_ugyintezes\_altalanos\_jellemzese';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgUgyintezesAltalanosJellemzese';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgUgyintezesAltalanosJellemzese';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 15;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 15;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_ugyintezes\_altalanos\_jellemzese.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_ugyintezes\_altalanos\_jellemzese.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the EGESZSEGUGYI\_ELLATASOK\_SZAMA field

\*/

const COL\_EGESZSEGUGYI\_ELLATASOK\_SZAMA = 'stg\_ugyintezes\_altalanos\_jellemzese.EGESZSEGUGYI\_ELLATASOK\_SZAMA';

/\*\*

\* the column name for the POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA field

\*/

const COL\_POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA = 'stg\_ugyintezes\_altalanos\_jellemzese.POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA';

/\*\*

\* the column name for the BUNTETOPEREK\_SZAMA field

\*/

const COL\_BUNTETOPEREK\_SZAMA = 'stg\_ugyintezes\_altalanos\_jellemzese.BUNTETOPEREK\_SZAMA';

/\*\*

\* the column name for the KERESKEDELMI\_SZOLGALTATASOK\_SZAMA field

\*/

const COL\_KERESKEDELMI\_SZOLGALTATASOK\_SZAMA = 'stg\_ugyintezes\_altalanos\_jellemzese.KERESKEDELMI\_SZOLGALTATASOK\_SZAMA';

/\*\*

\* the column name for the SZOCIALIS\_SZOLGALTATASOK\_SZAMA field

\*/

const COL\_SZOCIALIS\_SZOLGALTATASOK\_SZAMA = 'stg\_ugyintezes\_altalanos\_jellemzese.SZOCIALIS\_SZOLGALTATASOK\_SZAMA';

/\*\*

\* the column name for the OKMANYIRODAI\_UGYINTEZESEK\_SZAMA field

\*/

const COL\_OKMANYIRODAI\_UGYINTEZESEK\_SZAMA = 'stg\_ugyintezes\_altalanos\_jellemzese.OKMANYIRODAI\_UGYINTEZESEK\_SZAMA';

/\*\*

\* the column name for the BANKOK\_SZAMA field

\*/

const COL\_BANKOK\_SZAMA = 'stg\_ugyintezes\_altalanos\_jellemzese.BANKOK\_SZAMA';

/\*\*

\* the column name for the BENZINKUTAK\_SZAMA field

\*/

const COL\_BENZINKUTAK\_SZAMA = 'stg\_ugyintezes\_altalanos\_jellemzese.BENZINKUTAK\_SZAMA';

/\*\*

\* the column name for the KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA field

\*/

const COL\_KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA = 'stg\_ugyintezes\_altalanos\_jellemzese.KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA';

/\*\*

\* the column name for the KOZMUSZOLGALTATASOK field

\*/

const COL\_KOZMUSZOLGALTATASOK = 'stg\_ugyintezes\_altalanos\_jellemzese.KOZMUSZOLGALTATASOK';

/\*\*

\* the column name for the MUNKAUGYI\_SZOLGALTATASOK\_SZAMA field

\*/

const COL\_MUNKAUGYI\_SZOLGALTATASOK\_SZAMA = 'stg\_ugyintezes\_altalanos\_jellemzese.MUNKAUGYI\_SZOLGALTATASOK\_SZAMA';

/\*\*

\* the column name for the BOLCSODEI\_SZOLGALTATASOK\_SZAMA field

\*/

const COL\_BOLCSODEI\_SZOLGALTATASOK\_SZAMA = 'stg\_ugyintezes\_altalanos\_jellemzese.BOLCSODEI\_SZOLGALTATASOK\_SZAMA';

/\*\*

\* the column name for the MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA field

\*/

const COL\_MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA = 'stg\_ugyintezes\_altalanos\_jellemzese.MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'EgeszsegugyiEllatasokSzama', 'PostaiGyogyszertariSzolgaltatasokSzama', 'BuntetoperekSzama', 'KereskedelmiSzolgaltatasokSzama', 'SzocialisSzolgaltatasokSzama', 'OkmanyirodaiUgyintezesekSzama', 'BankokSzama', 'BenzinkutakSzama', 'KorjegyzosegekFoldhivatalokSzama', 'Kozmuszolgaltatasok', 'MunkaugyiSzolgaltatasokSzama', 'BolcsodeiSzolgaltatasokSzama', 'MobiltelefonUgyintezesiSzolgaltatasokSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'egeszsegugyiEllatasokSzama', 'postaiGyogyszertariSzolgaltatasokSzama', 'buntetoperekSzama', 'kereskedelmiSzolgaltatasokSzama', 'szocialisSzolgaltatasokSzama', 'okmanyirodaiUgyintezesekSzama', 'bankokSzama', 'benzinkutakSzama', 'korjegyzosegekFoldhivatalokSzama', 'kozmuszolgaltatasok', 'munkaugyiSzolgaltatasokSzama', 'bolcsodeiSzolgaltatasokSzama', 'mobiltelefonUgyintezesiSzolgaltatasokSzama', ),

self::TYPE\_COLNAME => array(StgUgyintezesAltalanosJellemzeseTableMap::COL\_TELEPULES\_NEV, StgUgyintezesAltalanosJellemzeseTableMap::COL\_TELEPULES\_KSHKOD, StgUgyintezesAltalanosJellemzeseTableMap::COL\_EGESZSEGUGYI\_ELLATASOK\_SZAMA, StgUgyintezesAltalanosJellemzeseTableMap::COL\_POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA, StgUgyintezesAltalanosJellemzeseTableMap::COL\_BUNTETOPEREK\_SZAMA, StgUgyintezesAltalanosJellemzeseTableMap::COL\_KERESKEDELMI\_SZOLGALTATASOK\_SZAMA, StgUgyintezesAltalanosJellemzeseTableMap::COL\_SZOCIALIS\_SZOLGALTATASOK\_SZAMA, StgUgyintezesAltalanosJellemzeseTableMap::COL\_OKMANYIRODAI\_UGYINTEZESEK\_SZAMA, StgUgyintezesAltalanosJellemzeseTableMap::COL\_BANKOK\_SZAMA, StgUgyintezesAltalanosJellemzeseTableMap::COL\_BENZINKUTAK\_SZAMA, StgUgyintezesAltalanosJellemzeseTableMap::COL\_KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA, StgUgyintezesAltalanosJellemzeseTableMap::COL\_KOZMUSZOLGALTATASOK, StgUgyintezesAltalanosJellemzeseTableMap::COL\_MUNKAUGYI\_SZOLGALTATASOK\_SZAMA, StgUgyintezesAltalanosJellemzeseTableMap::COL\_BOLCSODEI\_SZOLGALTATASOK\_SZAMA, StgUgyintezesAltalanosJellemzeseTableMap::COL\_MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_EGESZSEGUGYI\_ELLATASOK\_SZAMA', 'COL\_POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA', 'COL\_BUNTETOPEREK\_SZAMA', 'COL\_KERESKEDELMI\_SZOLGALTATASOK\_SZAMA', 'COL\_SZOCIALIS\_SZOLGALTATASOK\_SZAMA', 'COL\_OKMANYIRODAI\_UGYINTEZESEK\_SZAMA', 'COL\_BANKOK\_SZAMA', 'COL\_BENZINKUTAK\_SZAMA', 'COL\_KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA', 'COL\_KOZMUSZOLGALTATASOK', 'COL\_MUNKAUGYI\_SZOLGALTATASOK\_SZAMA', 'COL\_BOLCSODEI\_SZOLGALTATASOK\_SZAMA', 'COL\_MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'egeszsegugyi\_ellatasok\_szama', 'postai\_gyogyszertari\_szolgaltatasok\_szama', 'buntetoperek\_szama', 'kereskedelmi\_szolgaltatasok\_szama', 'szocialis\_szolgaltatasok\_szama', 'okmanyirodai\_ugyintezesek\_szama', 'bankok\_szama', 'benzinkutak\_szama', 'korjegyzosegek\_foldhivatalok\_szama', 'kozmuszolgaltatasok', 'munkaugyi\_szolgaltatasok\_szama', 'bolcsodei\_szolgaltatasok\_szama', 'mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'EgeszsegugyiEllatasokSzama' => 2, 'PostaiGyogyszertariSzolgaltatasokSzama' => 3, 'BuntetoperekSzama' => 4, 'KereskedelmiSzolgaltatasokSzama' => 5, 'SzocialisSzolgaltatasokSzama' => 6, 'OkmanyirodaiUgyintezesekSzama' => 7, 'BankokSzama' => 8, 'BenzinkutakSzama' => 9, 'KorjegyzosegekFoldhivatalokSzama' => 10, 'Kozmuszolgaltatasok' => 11, 'MunkaugyiSzolgaltatasokSzama' => 12, 'BolcsodeiSzolgaltatasokSzama' => 13, 'MobiltelefonUgyintezesiSzolgaltatasokSzama' => 14, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'egeszsegugyiEllatasokSzama' => 2, 'postaiGyogyszertariSzolgaltatasokSzama' => 3, 'buntetoperekSzama' => 4, 'kereskedelmiSzolgaltatasokSzama' => 5, 'szocialisSzolgaltatasokSzama' => 6, 'okmanyirodaiUgyintezesekSzama' => 7, 'bankokSzama' => 8, 'benzinkutakSzama' => 9, 'korjegyzosegekFoldhivatalokSzama' => 10, 'kozmuszolgaltatasok' => 11, 'munkaugyiSzolgaltatasokSzama' => 12, 'bolcsodeiSzolgaltatasokSzama' => 13, 'mobiltelefonUgyintezesiSzolgaltatasokSzama' => 14, ),

self::TYPE\_COLNAME => array(StgUgyintezesAltalanosJellemzeseTableMap::COL\_TELEPULES\_NEV => 0, StgUgyintezesAltalanosJellemzeseTableMap::COL\_TELEPULES\_KSHKOD => 1, StgUgyintezesAltalanosJellemzeseTableMap::COL\_EGESZSEGUGYI\_ELLATASOK\_SZAMA => 2, StgUgyintezesAltalanosJellemzeseTableMap::COL\_POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA => 3, StgUgyintezesAltalanosJellemzeseTableMap::COL\_BUNTETOPEREK\_SZAMA => 4, StgUgyintezesAltalanosJellemzeseTableMap::COL\_KERESKEDELMI\_SZOLGALTATASOK\_SZAMA => 5, StgUgyintezesAltalanosJellemzeseTableMap::COL\_SZOCIALIS\_SZOLGALTATASOK\_SZAMA => 6, StgUgyintezesAltalanosJellemzeseTableMap::COL\_OKMANYIRODAI\_UGYINTEZESEK\_SZAMA => 7, StgUgyintezesAltalanosJellemzeseTableMap::COL\_BANKOK\_SZAMA => 8, StgUgyintezesAltalanosJellemzeseTableMap::COL\_BENZINKUTAK\_SZAMA => 9, StgUgyintezesAltalanosJellemzeseTableMap::COL\_KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA => 10, StgUgyintezesAltalanosJellemzeseTableMap::COL\_KOZMUSZOLGALTATASOK => 11, StgUgyintezesAltalanosJellemzeseTableMap::COL\_MUNKAUGYI\_SZOLGALTATASOK\_SZAMA => 12, StgUgyintezesAltalanosJellemzeseTableMap::COL\_BOLCSODEI\_SZOLGALTATASOK\_SZAMA => 13, StgUgyintezesAltalanosJellemzeseTableMap::COL\_MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA => 14, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_EGESZSEGUGYI\_ELLATASOK\_SZAMA' => 2, 'COL\_POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA' => 3, 'COL\_BUNTETOPEREK\_SZAMA' => 4, 'COL\_KERESKEDELMI\_SZOLGALTATASOK\_SZAMA' => 5, 'COL\_SZOCIALIS\_SZOLGALTATASOK\_SZAMA' => 6, 'COL\_OKMANYIRODAI\_UGYINTEZESEK\_SZAMA' => 7, 'COL\_BANKOK\_SZAMA' => 8, 'COL\_BENZINKUTAK\_SZAMA' => 9, 'COL\_KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA' => 10, 'COL\_KOZMUSZOLGALTATASOK' => 11, 'COL\_MUNKAUGYI\_SZOLGALTATASOK\_SZAMA' => 12, 'COL\_BOLCSODEI\_SZOLGALTATASOK\_SZAMA' => 13, 'COL\_MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA' => 14, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'egeszsegugyi\_ellatasok\_szama' => 2, 'postai\_gyogyszertari\_szolgaltatasok\_szama' => 3, 'buntetoperek\_szama' => 4, 'kereskedelmi\_szolgaltatasok\_szama' => 5, 'szocialis\_szolgaltatasok\_szama' => 6, 'okmanyirodai\_ugyintezesek\_szama' => 7, 'bankok\_szama' => 8, 'benzinkutak\_szama' => 9, 'korjegyzosegek\_foldhivatalok\_szama' => 10, 'kozmuszolgaltatasok' => 11, 'munkaugyi\_szolgaltatasok\_szama' => 12, 'bolcsodei\_szolgaltatasok\_szama' => 13, 'mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama' => 14, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_ugyintezes\_altalanos\_jellemzese');

$this->setPhpName('StgUgyintezesAltalanosJellemzese');

$this->setClassName('\\StgUgyintezesAltalanosJellemzese');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('EGESZSEGUGYI\_ELLATASOK\_SZAMA', 'EgeszsegugyiEllatasokSzama', 'INTEGER', false, null, null);

$this->addColumn('POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA', 'PostaiGyogyszertariSzolgaltatasokSzama', 'INTEGER', false, null, null);

$this->addColumn('BUNTETOPEREK\_SZAMA', 'BuntetoperekSzama', 'INTEGER', false, null, null);

$this->addColumn('KERESKEDELMI\_SZOLGALTATASOK\_SZAMA', 'KereskedelmiSzolgaltatasokSzama', 'INTEGER', false, null, null);

$this->addColumn('SZOCIALIS\_SZOLGALTATASOK\_SZAMA', 'SzocialisSzolgaltatasokSzama', 'INTEGER', false, null, null);

$this->addColumn('OKMANYIRODAI\_UGYINTEZESEK\_SZAMA', 'OkmanyirodaiUgyintezesekSzama', 'INTEGER', false, null, null);

$this->addColumn('BANKOK\_SZAMA', 'BankokSzama', 'INTEGER', false, null, null);

$this->addColumn('BENZINKUTAK\_SZAMA', 'BenzinkutakSzama', 'INTEGER', false, null, null);

$this->addColumn('KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA', 'KorjegyzosegekFoldhivatalokSzama', 'INTEGER', false, null, null);

$this->addColumn('KOZMUSZOLGALTATASOK', 'Kozmuszolgaltatasok', 'INTEGER', false, null, null);

$this->addColumn('MUNKAUGYI\_SZOLGALTATASOK\_SZAMA', 'MunkaugyiSzolgaltatasokSzama', 'INTEGER', false, null, null);

$this->addColumn('BOLCSODEI\_SZOLGALTATASOK\_SZAMA', 'BolcsodeiSzolgaltatasokSzama', 'INTEGER', false, null, null);

$this->addColumn('MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA', 'MobiltelefonUgyintezesiSzolgaltatasokSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgUgyintezesAltalanosJellemzeseTableMap::CLASS\_DEFAULT : StgUgyintezesAltalanosJellemzeseTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgUgyintezesAltalanosJellemzese object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgUgyintezesAltalanosJellemzeseTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgUgyintezesAltalanosJellemzeseTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgUgyintezesAltalanosJellemzeseTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgUgyintezesAltalanosJellemzeseTableMap::OM\_CLASS;

/\*\* @var StgUgyintezesAltalanosJellemzese $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgUgyintezesAltalanosJellemzeseTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgUgyintezesAltalanosJellemzeseTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgUgyintezesAltalanosJellemzeseTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgUgyintezesAltalanosJellemzese $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgUgyintezesAltalanosJellemzeseTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgUgyintezesAltalanosJellemzeseTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgUgyintezesAltalanosJellemzeseTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgUgyintezesAltalanosJellemzeseTableMap::COL\_EGESZSEGUGYI\_ELLATASOK\_SZAMA);

$criteria->addSelectColumn(StgUgyintezesAltalanosJellemzeseTableMap::COL\_POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA);

$criteria->addSelectColumn(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BUNTETOPEREK\_SZAMA);

$criteria->addSelectColumn(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KERESKEDELMI\_SZOLGALTATASOK\_SZAMA);

$criteria->addSelectColumn(StgUgyintezesAltalanosJellemzeseTableMap::COL\_SZOCIALIS\_SZOLGALTATASOK\_SZAMA);

$criteria->addSelectColumn(StgUgyintezesAltalanosJellemzeseTableMap::COL\_OKMANYIRODAI\_UGYINTEZESEK\_SZAMA);

$criteria->addSelectColumn(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BANKOK\_SZAMA);

$criteria->addSelectColumn(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BENZINKUTAK\_SZAMA);

$criteria->addSelectColumn(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA);

$criteria->addSelectColumn(StgUgyintezesAltalanosJellemzeseTableMap::COL\_KOZMUSZOLGALTATASOK);

$criteria->addSelectColumn(StgUgyintezesAltalanosJellemzeseTableMap::COL\_MUNKAUGYI\_SZOLGALTATASOK\_SZAMA);

$criteria->addSelectColumn(StgUgyintezesAltalanosJellemzeseTableMap::COL\_BOLCSODEI\_SZOLGALTATASOK\_SZAMA);

$criteria->addSelectColumn(StgUgyintezesAltalanosJellemzeseTableMap::COL\_MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.EGESZSEGUGYI\_ELLATASOK\_SZAMA');

$criteria->addSelectColumn($alias . '.POSTAI\_GYOGYSZERTARI\_SZOLGALTATASOK\_SZAMA');

$criteria->addSelectColumn($alias . '.BUNTETOPEREK\_SZAMA');

$criteria->addSelectColumn($alias . '.KERESKEDELMI\_SZOLGALTATASOK\_SZAMA');

$criteria->addSelectColumn($alias . '.SZOCIALIS\_SZOLGALTATASOK\_SZAMA');

$criteria->addSelectColumn($alias . '.OKMANYIRODAI\_UGYINTEZESEK\_SZAMA');

$criteria->addSelectColumn($alias . '.BANKOK\_SZAMA');

$criteria->addSelectColumn($alias . '.BENZINKUTAK\_SZAMA');

$criteria->addSelectColumn($alias . '.KORJEGYZOSEGEK\_FOLDHIVATALOK\_SZAMA');

$criteria->addSelectColumn($alias . '.KOZMUSZOLGALTATASOK');

$criteria->addSelectColumn($alias . '.MUNKAUGYI\_SZOLGALTATASOK\_SZAMA');

$criteria->addSelectColumn($alias . '.BOLCSODEI\_SZOLGALTATASOK\_SZAMA');

$criteria->addSelectColumn($alias . '.MOBILTELEFON\_UGYINTEZESI\_SZOLGALTATASOK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgUgyintezesAltalanosJellemzeseTableMap::DATABASE\_NAME)->getTable(StgUgyintezesAltalanosJellemzeseTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgUgyintezesAltalanosJellemzeseTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgUgyintezesAltalanosJellemzeseTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgUgyintezesAltalanosJellemzeseTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgUgyintezesAltalanosJellemzese or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgUgyintezesAltalanosJellemzese object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgUgyintezesAltalanosJellemzeseTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgUgyintezesAltalanosJellemzese) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgUgyintezesAltalanosJellemzese object has no primary key');

}

$query = StgUgyintezesAltalanosJellemzeseQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgUgyintezesAltalanosJellemzeseTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgUgyintezesAltalanosJellemzeseTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_ugyintezes\_altalanos\_jellemzese table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgUgyintezesAltalanosJellemzeseQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgUgyintezesAltalanosJellemzese or Criteria object.

\*

\* @param mixed $criteria Criteria or StgUgyintezesAltalanosJellemzese object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgUgyintezesAltalanosJellemzeseTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgUgyintezesAltalanosJellemzese object

}

// Set the correct dbName

$query = StgUgyintezesAltalanosJellemzeseQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgUgyintezesAltalanosJellemzeseTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgUgyintezesAltalanosJellemzeseTableMap::buildTableMap();

### StgUzletekVendeglatohelyekGyogyszertarakTableMap.php

<?php

namespace Map;

use \StgUzletekVendeglatohelyekGyogyszertarak;

use \StgUzletekVendeglatohelyekGyogyszertarakQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_uzletek\_vendeglatohelyek\_gyogyszertarak' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgUzletekVendeglatohelyekGyogyszertarakTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgUzletekVendeglatohelyekGyogyszertarakTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_uzletek\_vendeglatohelyek\_gyogyszertarak';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgUzletekVendeglatohelyekGyogyszertarak';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgUzletekVendeglatohelyekGyogyszertarak';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 7;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 7;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_uzletek\_vendeglatohelyek\_gyogyszertarak.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_uzletek\_vendeglatohelyek\_gyogyszertarak.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA field

\*/

const COL\_ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA = 'stg\_uzletek\_vendeglatohelyek\_gyogyszertarak.ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA';

/\*\*

\* the column name for the KISKERESKEDELMI\_UZLETEK\_SZAMA field

\*/

const COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA = 'stg\_uzletek\_vendeglatohelyek\_gyogyszertarak.KISKERESKEDELMI\_UZLETEK\_SZAMA';

/\*\*

\* the column name for the VENDEGLATOHELYEK\_SZAMA field

\*/

const COL\_VENDEGLATOHELYEK\_SZAMA = 'stg\_uzletek\_vendeglatohelyek\_gyogyszertarak.VENDEGLATOHELYEK\_SZAMA';

/\*\*

\* the column name for the FIOKGYOGYSZERTARAK\_SZAMA field

\*/

const COL\_FIOKGYOGYSZERTARAK\_SZAMA = 'stg\_uzletek\_vendeglatohelyek\_gyogyszertarak.FIOKGYOGYSZERTARAK\_SZAMA';

/\*\*

\* the column name for the GYOGYSZERTARAK\_SZAMA field

\*/

const COL\_GYOGYSZERTARAK\_SZAMA = 'stg\_uzletek\_vendeglatohelyek\_gyogyszertarak.GYOGYSZERTARAK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'ElelmiszerVegyesuzletekAruhazakSzama', 'KiskereskedelmiUzletekSzama', 'VendeglatohelyekSzama', 'FiokgyogyszertarakSzama', 'GyogyszertarakSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'elelmiszerVegyesuzletekAruhazakSzama', 'kiskereskedelmiUzletekSzama', 'vendeglatohelyekSzama', 'fiokgyogyszertarakSzama', 'gyogyszertarakSzama', ),

self::TYPE\_COLNAME => array(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_TELEPULES\_NEV, StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_TELEPULES\_KSHKOD, StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA, StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA, StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_VENDEGLATOHELYEK\_SZAMA, StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_FIOKGYOGYSZERTARAK\_SZAMA, StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_GYOGYSZERTARAK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA', 'COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA', 'COL\_VENDEGLATOHELYEK\_SZAMA', 'COL\_FIOKGYOGYSZERTARAK\_SZAMA', 'COL\_GYOGYSZERTARAK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'elelmiszer\_vegyesuzletek\_aruhazak\_szama', 'kiskereskedelmi\_uzletek\_szama', 'vendeglatohelyek\_szama', 'fiokgyogyszertarak\_szama', 'gyogyszertarak\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'ElelmiszerVegyesuzletekAruhazakSzama' => 2, 'KiskereskedelmiUzletekSzama' => 3, 'VendeglatohelyekSzama' => 4, 'FiokgyogyszertarakSzama' => 5, 'GyogyszertarakSzama' => 6, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'elelmiszerVegyesuzletekAruhazakSzama' => 2, 'kiskereskedelmiUzletekSzama' => 3, 'vendeglatohelyekSzama' => 4, 'fiokgyogyszertarakSzama' => 5, 'gyogyszertarakSzama' => 6, ),

self::TYPE\_COLNAME => array(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_TELEPULES\_NEV => 0, StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_TELEPULES\_KSHKOD => 1, StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA => 2, StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA => 3, StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_VENDEGLATOHELYEK\_SZAMA => 4, StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_FIOKGYOGYSZERTARAK\_SZAMA => 5, StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_GYOGYSZERTARAK\_SZAMA => 6, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA' => 2, 'COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA' => 3, 'COL\_VENDEGLATOHELYEK\_SZAMA' => 4, 'COL\_FIOKGYOGYSZERTARAK\_SZAMA' => 5, 'COL\_GYOGYSZERTARAK\_SZAMA' => 6, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'elelmiszer\_vegyesuzletek\_aruhazak\_szama' => 2, 'kiskereskedelmi\_uzletek\_szama' => 3, 'vendeglatohelyek\_szama' => 4, 'fiokgyogyszertarak\_szama' => 5, 'gyogyszertarak\_szama' => 6, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, 6, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_uzletek\_vendeglatohelyek\_gyogyszertarak');

$this->setPhpName('StgUzletekVendeglatohelyekGyogyszertarak');

$this->setClassName('\\StgUzletekVendeglatohelyekGyogyszertarak');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA', 'ElelmiszerVegyesuzletekAruhazakSzama', 'INTEGER', false, null, null);

$this->addColumn('KISKERESKEDELMI\_UZLETEK\_SZAMA', 'KiskereskedelmiUzletekSzama', 'INTEGER', false, null, null);

$this->addColumn('VENDEGLATOHELYEK\_SZAMA', 'VendeglatohelyekSzama', 'INTEGER', false, null, null);

$this->addColumn('FIOKGYOGYSZERTARAK\_SZAMA', 'FiokgyogyszertarakSzama', 'INTEGER', false, null, null);

$this->addColumn('GYOGYSZERTARAK\_SZAMA', 'GyogyszertarakSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgUzletekVendeglatohelyekGyogyszertarakTableMap::CLASS\_DEFAULT : StgUzletekVendeglatohelyekGyogyszertarakTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgUzletekVendeglatohelyekGyogyszertarak object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgUzletekVendeglatohelyekGyogyszertarakTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgUzletekVendeglatohelyekGyogyszertarakTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgUzletekVendeglatohelyekGyogyszertarakTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgUzletekVendeglatohelyekGyogyszertarakTableMap::OM\_CLASS;

/\*\* @var StgUzletekVendeglatohelyekGyogyszertarak $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgUzletekVendeglatohelyekGyogyszertarakTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgUzletekVendeglatohelyekGyogyszertarakTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgUzletekVendeglatohelyekGyogyszertarakTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgUzletekVendeglatohelyekGyogyszertarak $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgUzletekVendeglatohelyekGyogyszertarakTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA);

$criteria->addSelectColumn(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_KISKERESKEDELMI\_UZLETEK\_SZAMA);

$criteria->addSelectColumn(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_VENDEGLATOHELYEK\_SZAMA);

$criteria->addSelectColumn(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_FIOKGYOGYSZERTARAK\_SZAMA);

$criteria->addSelectColumn(StgUzletekVendeglatohelyekGyogyszertarakTableMap::COL\_GYOGYSZERTARAK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.ELELMISZER\_VEGYESUZLETEK\_ARUHAZAK\_SZAMA');

$criteria->addSelectColumn($alias . '.KISKERESKEDELMI\_UZLETEK\_SZAMA');

$criteria->addSelectColumn($alias . '.VENDEGLATOHELYEK\_SZAMA');

$criteria->addSelectColumn($alias . '.FIOKGYOGYSZERTARAK\_SZAMA');

$criteria->addSelectColumn($alias . '.GYOGYSZERTARAK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgUzletekVendeglatohelyekGyogyszertarakTableMap::DATABASE\_NAME)->getTable(StgUzletekVendeglatohelyekGyogyszertarakTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgUzletekVendeglatohelyekGyogyszertarakTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgUzletekVendeglatohelyekGyogyszertarakTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgUzletekVendeglatohelyekGyogyszertarakTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgUzletekVendeglatohelyekGyogyszertarak or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgUzletekVendeglatohelyekGyogyszertarak object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgUzletekVendeglatohelyekGyogyszertarakTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgUzletekVendeglatohelyekGyogyszertarak) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgUzletekVendeglatohelyekGyogyszertarak object has no primary key');

}

$query = StgUzletekVendeglatohelyekGyogyszertarakQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgUzletekVendeglatohelyekGyogyszertarakTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgUzletekVendeglatohelyekGyogyszertarakTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_uzletek\_vendeglatohelyek\_gyogyszertarak table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgUzletekVendeglatohelyekGyogyszertarakQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgUzletekVendeglatohelyekGyogyszertarak or Criteria object.

\*

\* @param mixed $criteria Criteria or StgUzletekVendeglatohelyekGyogyszertarak object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgUzletekVendeglatohelyekGyogyszertarakTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgUzletekVendeglatohelyekGyogyszertarak object

}

// Set the correct dbName

$query = StgUzletekVendeglatohelyekGyogyszertarakQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgUzletekVendeglatohelyekGyogyszertarakTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgUzletekVendeglatohelyekGyogyszertarakTableMap::buildTableMap();

### StgVandorlasiEgyenlegAranyaTableMap.php

<?php

namespace Map;

use \StgVandorlasiEgyenlegAranya;

use \StgVandorlasiEgyenlegAranyaQuery;

use Propel\Runtime\Propel;

use Propel\Runtime\ActiveQuery\Criteria;

use Propel\Runtime\ActiveQuery\InstancePoolTrait;

use Propel\Runtime\Connection\ConnectionInterface;

use Propel\Runtime\DataFetcher\DataFetcherInterface;

use Propel\Runtime\Exception\LogicException;

use Propel\Runtime\Exception\PropelException;

use Propel\Runtime\Map\RelationMap;

use Propel\Runtime\Map\TableMap;

use Propel\Runtime\Map\TableMapTrait;

/\*\*

\* This class defines the structure of the 'stg\_vandorlasi\_egyenleg\_aranya' table.

\*

\*

\*

\* This map class is used by Propel to do runtime db structure discovery.

\* For example, the createSelectSql() method checks the type of a given column used in an

\* ORDER BY clause to know whether it needs to apply SQL to make the ORDER BY case-insensitive

\* (i.e. if it's a text column type).

\*

\*/

class StgVandorlasiEgyenlegAranyaTableMap extends TableMap

{

use InstancePoolTrait;

use TableMapTrait;

/\*\*

\* The (dot-path) name of this class

\*/

const CLASS\_NAME = '.Map.StgVandorlasiEgyenlegAranyaTableMap';

/\*\*

\* The default database name for this class

\*/

const DATABASE\_NAME = 'etraffic\_v1';

/\*\*

\* The table name for this class

\*/

const TABLE\_NAME = 'stg\_vandorlasi\_egyenleg\_aranya';

/\*\*

\* The related Propel class for this table

\*/

const OM\_CLASS = '\\StgVandorlasiEgyenlegAranya';

/\*\*

\* A class that can be returned by this tableMap

\*/

const CLASS\_DEFAULT = 'StgVandorlasiEgyenlegAranya';

/\*\*

\* The total number of columns

\*/

const NUM\_COLUMNS = 6;

/\*\*

\* The number of lazy-loaded columns

\*/

const NUM\_LAZY\_LOAD\_COLUMNS = 0;

/\*\*

\* The number of columns to hydrate (NUM\_COLUMNS - NUM\_LAZY\_LOAD\_COLUMNS)

\*/

const NUM\_HYDRATE\_COLUMNS = 6;

/\*\*

\* the column name for the TELEPULES\_NEV field

\*/

const COL\_TELEPULES\_NEV = 'stg\_vandorlasi\_egyenleg\_aranya.TELEPULES\_NEV';

/\*\*

\* the column name for the TELEPULES\_KSHKOD field

\*/

const COL\_TELEPULES\_KSHKOD = 'stg\_vandorlasi\_egyenleg\_aranya.TELEPULES\_KSHKOD';

/\*\*

\* the column name for the ODAVANDORLASOK\_SZAMA field

\*/

const COL\_ODAVANDORLASOK\_SZAMA = 'stg\_vandorlasi\_egyenleg\_aranya.ODAVANDORLASOK\_SZAMA';

/\*\*

\* the column name for the ELVANDORLASOK\_SZAMA field

\*/

const COL\_ELVANDORLASOK\_SZAMA = 'stg\_vandorlasi\_egyenleg\_aranya.ELVANDORLASOK\_SZAMA';

/\*\*

\* the column name for the ELVESZULETESEK\_SZAMA field

\*/

const COL\_ELVESZULETESEK\_SZAMA = 'stg\_vandorlasi\_egyenleg\_aranya.ELVESZULETESEK\_SZAMA';

/\*\*

\* the column name for the HALALOZASOK\_SZAMA field

\*/

const COL\_HALALOZASOK\_SZAMA = 'stg\_vandorlasi\_egyenleg\_aranya.HALALOZASOK\_SZAMA';

/\*\*

\* The default string format for model objects of the related table

\*/

const DEFAULT\_STRING\_FORMAT = 'YAML';

/\*\*

\* holds an array of fieldnames

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldNames[self::TYPE\_PHPNAME][0] = 'Id'

\*/

protected static $fieldNames = array (

self::TYPE\_PHPNAME => array('TelepulesNev', 'TelepulesKshkod', 'OdavandorlasokSzama', 'ElvandorlasokSzama', 'ElveszuletesekSzama', 'HalalozasokSzama', ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev', 'telepulesKshkod', 'odavandorlasokSzama', 'elvandorlasokSzama', 'elveszuletesekSzama', 'halalozasokSzama', ),

self::TYPE\_COLNAME => array(StgVandorlasiEgyenlegAranyaTableMap::COL\_TELEPULES\_NEV, StgVandorlasiEgyenlegAranyaTableMap::COL\_TELEPULES\_KSHKOD, StgVandorlasiEgyenlegAranyaTableMap::COL\_ODAVANDORLASOK\_SZAMA, StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVANDORLASOK\_SZAMA, StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVESZULETESEK\_SZAMA, StgVandorlasiEgyenlegAranyaTableMap::COL\_HALALOZASOK\_SZAMA, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV', 'COL\_TELEPULES\_KSHKOD', 'COL\_ODAVANDORLASOK\_SZAMA', 'COL\_ELVANDORLASOK\_SZAMA', 'COL\_ELVESZULETESEK\_SZAMA', 'COL\_HALALOZASOK\_SZAMA', ),

self::TYPE\_FIELDNAME => array('telepules\_nev', 'telepules\_KSHKOD', 'odavandorlasok\_szama', 'elvandorlasok\_szama', 'elveszuletesek\_szama', 'halalozasok\_szama', ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, )

);

/\*\*

\* holds an array of keys for quick access to the fieldnames array

\*

\* first dimension keys are the type constants

\* e.g. self::$fieldKeys[self::TYPE\_PHPNAME]['Id'] = 0

\*/

protected static $fieldKeys = array (

self::TYPE\_PHPNAME => array('TelepulesNev' => 0, 'TelepulesKshkod' => 1, 'OdavandorlasokSzama' => 2, 'ElvandorlasokSzama' => 3, 'ElveszuletesekSzama' => 4, 'HalalozasokSzama' => 5, ),

self::TYPE\_STUDLYPHPNAME => array('telepulesNev' => 0, 'telepulesKshkod' => 1, 'odavandorlasokSzama' => 2, 'elvandorlasokSzama' => 3, 'elveszuletesekSzama' => 4, 'halalozasokSzama' => 5, ),

self::TYPE\_COLNAME => array(StgVandorlasiEgyenlegAranyaTableMap::COL\_TELEPULES\_NEV => 0, StgVandorlasiEgyenlegAranyaTableMap::COL\_TELEPULES\_KSHKOD => 1, StgVandorlasiEgyenlegAranyaTableMap::COL\_ODAVANDORLASOK\_SZAMA => 2, StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVANDORLASOK\_SZAMA => 3, StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVESZULETESEK\_SZAMA => 4, StgVandorlasiEgyenlegAranyaTableMap::COL\_HALALOZASOK\_SZAMA => 5, ),

self::TYPE\_RAW\_COLNAME => array('COL\_TELEPULES\_NEV' => 0, 'COL\_TELEPULES\_KSHKOD' => 1, 'COL\_ODAVANDORLASOK\_SZAMA' => 2, 'COL\_ELVANDORLASOK\_SZAMA' => 3, 'COL\_ELVESZULETESEK\_SZAMA' => 4, 'COL\_HALALOZASOK\_SZAMA' => 5, ),

self::TYPE\_FIELDNAME => array('telepules\_nev' => 0, 'telepules\_KSHKOD' => 1, 'odavandorlasok\_szama' => 2, 'elvandorlasok\_szama' => 3, 'elveszuletesek\_szama' => 4, 'halalozasok\_szama' => 5, ),

self::TYPE\_NUM => array(0, 1, 2, 3, 4, 5, )

);

/\*\*

\* Initialize the table attributes and columns

\* Relations are not initialized by this method since they are lazy loaded

\*

\* @return void

\* @throws PropelException

\*/

public function initialize()

{

// attributes

$this->setName('stg\_vandorlasi\_egyenleg\_aranya');

$this->setPhpName('StgVandorlasiEgyenlegAranya');

$this->setClassName('\\StgVandorlasiEgyenlegAranya');

$this->setPackage('');

$this->setUseIdGenerator(false);

// columns

$this->addColumn('TELEPULES\_NEV', 'TelepulesNev', 'VARCHAR', true, 45, null);

$this->addColumn('TELEPULES\_KSHKOD', 'TelepulesKshkod', 'VARCHAR', false, 45, null);

$this->addColumn('ODAVANDORLASOK\_SZAMA', 'OdavandorlasokSzama', 'INTEGER', false, null, null);

$this->addColumn('ELVANDORLASOK\_SZAMA', 'ElvandorlasokSzama', 'INTEGER', false, null, null);

$this->addColumn('ELVESZULETESEK\_SZAMA', 'ElveszuletesekSzama', 'INTEGER', false, null, null);

$this->addColumn('HALALOZASOK\_SZAMA', 'HalalozasokSzama', 'INTEGER', false, null, null);

} // initialize()

/\*\*

\* Build the RelationMap objects for this table relationships

\*/

public function buildRelations()

{

} // buildRelations()

/\*\*

\* Retrieves a string version of the primary key from the DB resultset row that can be used to uniquely identify a row in this table.

\*

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, a serialize()d version of the primary key will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return string The primary key hash of the row

\*/

public static function getPrimaryKeyHashFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return null;

}

/\*\*

\* Retrieves the primary key from the DB resultset row

\* For tables with a single-column primary key, that simple pkey value will be returned. For tables with

\* a multi-column primary key, an array of the primary key columns will be returned.

\*

\* @param array $row resultset row.

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM

\*

\* @return mixed The primary key of the row

\*/

public static function getPrimaryKeyFromRow($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

return '';

}

/\*\*

\* The class that the tableMap will make instances of.

\*

\* If $withPrefix is true, the returned path

\* uses a dot-path notation which is translated into a path

\* relative to a location on the PHP include\_path.

\* (e.g. path.to.MyClass -> 'path/to/MyClass.php')

\*

\* @param boolean $withPrefix Whether or not to return the path with the class name

\* @return string path.to.ClassName

\*/

public static function getOMClass($withPrefix = true)

{

return $withPrefix ? StgVandorlasiEgyenlegAranyaTableMap::CLASS\_DEFAULT : StgVandorlasiEgyenlegAranyaTableMap::OM\_CLASS;

}

/\*\*

\* Populates an object of the default type or an object that inherit from the default.

\*

\* @param array $row row returned by DataFetcher->fetch().

\* @param int $offset The 0-based offset for reading from the resultset row.

\* @param string $indexType The index type of $row. Mostly DataFetcher->getIndexType().

One of the class type constants TableMap::TYPE\_PHPNAME, TableMap::TYPE\_STUDLYPHPNAME

\* TableMap::TYPE\_COLNAME, TableMap::TYPE\_FIELDNAME, TableMap::TYPE\_NUM.

\*

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\* @return array (StgVandorlasiEgyenlegAranya object, last column rank)

\*/

public static function populateObject($row, $offset = 0, $indexType = TableMap::TYPE\_NUM)

{

$key = StgVandorlasiEgyenlegAranyaTableMap::getPrimaryKeyHashFromRow($row, $offset, $indexType);

if (null !== ($obj = StgVandorlasiEgyenlegAranyaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, $offset, true); // rehydrate

$col = $offset + StgVandorlasiEgyenlegAranyaTableMap::NUM\_HYDRATE\_COLUMNS;

} else {

$cls = StgVandorlasiEgyenlegAranyaTableMap::OM\_CLASS;

/\*\* @var StgVandorlasiEgyenlegAranya $obj \*/

$obj = new $cls();

$col = $obj->hydrate($row, $offset, false, $indexType);

StgVandorlasiEgyenlegAranyaTableMap::addInstanceToPool($obj, $key);

}

return array($obj, $col);

}

/\*\*

\* The returned array will contain objects of the default type or

\* objects that inherit from the default.

\*

\* @param DataFetcherInterface $dataFetcher

\* @return array

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function populateObjects(DataFetcherInterface $dataFetcher)

{

$results = array();

// set the class once to avoid overhead in the loop

$cls = static::getOMClass(false);

// populate the object(s)

while ($row = $dataFetcher->fetch()) {

$key = StgVandorlasiEgyenlegAranyaTableMap::getPrimaryKeyHashFromRow($row, 0, $dataFetcher->getIndexType());

if (null !== ($obj = StgVandorlasiEgyenlegAranyaTableMap::getInstanceFromPool($key))) {

// We no longer rehydrate the object, since this can cause data loss.

// See http://www.propelorm.org/ticket/509

// $obj->hydrate($row, 0, true); // rehydrate

$results[] = $obj;

} else {

/\*\* @var StgVandorlasiEgyenlegAranya $obj \*/

$obj = new $cls();

$obj->hydrate($row);

$results[] = $obj;

StgVandorlasiEgyenlegAranyaTableMap::addInstanceToPool($obj, $key);

} // if key exists

}

return $results;

}

/\*\*

\* Add all the columns needed to create a new object.

\*

\* Note: any columns that were marked with lazyLoad="true" in the

\* XML schema will not be added to the select list and only loaded

\* on demand.

\*

\* @param Criteria $criteria object containing the columns to add.

\* @param string $alias optional table alias

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function addSelectColumns(Criteria $criteria, $alias = null)

{

if (null === $alias) {

$criteria->addSelectColumn(StgVandorlasiEgyenlegAranyaTableMap::COL\_TELEPULES\_NEV);

$criteria->addSelectColumn(StgVandorlasiEgyenlegAranyaTableMap::COL\_TELEPULES\_KSHKOD);

$criteria->addSelectColumn(StgVandorlasiEgyenlegAranyaTableMap::COL\_ODAVANDORLASOK\_SZAMA);

$criteria->addSelectColumn(StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVANDORLASOK\_SZAMA);

$criteria->addSelectColumn(StgVandorlasiEgyenlegAranyaTableMap::COL\_ELVESZULETESEK\_SZAMA);

$criteria->addSelectColumn(StgVandorlasiEgyenlegAranyaTableMap::COL\_HALALOZASOK\_SZAMA);

} else {

$criteria->addSelectColumn($alias . '.TELEPULES\_NEV');

$criteria->addSelectColumn($alias . '.TELEPULES\_KSHKOD');

$criteria->addSelectColumn($alias . '.ODAVANDORLASOK\_SZAMA');

$criteria->addSelectColumn($alias . '.ELVANDORLASOK\_SZAMA');

$criteria->addSelectColumn($alias . '.ELVESZULETESEK\_SZAMA');

$criteria->addSelectColumn($alias . '.HALALOZASOK\_SZAMA');

}

}

/\*\*

\* Returns the TableMap related to this object.

\* This method is not needed for general use but a specific application could have a need.

\* @return TableMap

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function getTableMap()

{

return Propel::getServiceContainer()->getDatabaseMap(StgVandorlasiEgyenlegAranyaTableMap::DATABASE\_NAME)->getTable(StgVandorlasiEgyenlegAranyaTableMap::TABLE\_NAME);

}

/\*\*

\* Add a TableMap instance to the database for this tableMap class.

\*/

public static function buildTableMap()

{

$dbMap = Propel::getServiceContainer()->getDatabaseMap(StgVandorlasiEgyenlegAranyaTableMap::DATABASE\_NAME);

if (!$dbMap->hasTable(StgVandorlasiEgyenlegAranyaTableMap::TABLE\_NAME)) {

$dbMap->addTableObject(new StgVandorlasiEgyenlegAranyaTableMap());

}

}

/\*\*

\* Performs a DELETE on the database, given a StgVandorlasiEgyenlegAranya or Criteria object OR a primary key value.

\*

\* @param mixed $values Criteria or StgVandorlasiEgyenlegAranya object or primary key or array of primary keys

\* which is used to create the DELETE statement

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver). This includes CASCADE-related rows

\* if supported by native driver or if emulated using Propel.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doDelete($values, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgVandorlasiEgyenlegAranyaTableMap::DATABASE\_NAME);

}

if ($values instanceof Criteria) {

// rename for clarity

$criteria = $values;

} elseif ($values instanceof \StgVandorlasiEgyenlegAranya) { // it's a model object

// create criteria based on pk value

$criteria = $values->buildCriteria();

} else { // it's a primary key, or an array of pks

throw new LogicException('The StgVandorlasiEgyenlegAranya object has no primary key');

}

$query = StgVandorlasiEgyenlegAranyaQuery::create()->mergeWith($criteria);

if ($values instanceof Criteria) {

StgVandorlasiEgyenlegAranyaTableMap::clearInstancePool();

} elseif (!is\_object($values)) { // it's a primary key, or an array of pks

foreach ((array) $values as $singleval) {

StgVandorlasiEgyenlegAranyaTableMap::removeInstanceFromPool($singleval);

}

}

return $query->delete($con);

}

/\*\*

\* Deletes all rows from the stg\_vandorlasi\_egyenleg\_aranya table.

\*

\* @param ConnectionInterface $con the connection to use

\* @return int The number of affected rows (if supported by underlying database driver).

\*/

public static function doDeleteAll(ConnectionInterface $con = null)

{

return StgVandorlasiEgyenlegAranyaQuery::create()->doDeleteAll($con);

}

/\*\*

\* Performs an INSERT on the database, given a StgVandorlasiEgyenlegAranya or Criteria object.

\*

\* @param mixed $criteria Criteria or StgVandorlasiEgyenlegAranya object containing data that is used to create the INSERT statement.

\* @param ConnectionInterface $con the ConnectionInterface connection to use

\* @return mixed The new primary key.

\* @throws PropelException Any exceptions caught during processing will be

\* rethrown wrapped into a PropelException.

\*/

public static function doInsert($criteria, ConnectionInterface $con = null)

{

if (null === $con) {

$con = Propel::getServiceContainer()->getWriteConnection(StgVandorlasiEgyenlegAranyaTableMap::DATABASE\_NAME);

}

if ($criteria instanceof Criteria) {

$criteria = clone $criteria; // rename for clarity

} else {

$criteria = $criteria->buildCriteria(); // build Criteria from StgVandorlasiEgyenlegAranya object

}

// Set the correct dbName

$query = StgVandorlasiEgyenlegAranyaQuery::create()->mergeWith($criteria);

// use transaction because $criteria could contain info

// for more than one table (I guess, conceivably)

return $con->transaction(function () use ($con, $query) {

return $query->doInsert($con);

});

}

} // StgVandorlasiEgyenlegAranyaTableMap

// This is the static code needed to register the TableMap for this table with the main Propel class.

//

StgVandorlasiEgyenlegAranyaTableMap::buildTableMap();

## SQL

### stg\_10\_fonel\_nagyobb\_szervezetek

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_10\_fonel\_nagyobb\_szervezetek` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`mukodo\_vallalatok\_szama` INT(11) NULL DEFAULT NULL,

`vallalatok\_atlagos\_letszama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_elelmiszeruzletek\_es\_aruhazak

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_elelmiszeruzletek\_es\_aruhazak` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`kiskereskedelmi\_uzletek\_szama` INT(11) NULL DEFAULT NULL,

`elelmiszer\_vegyesuzletek\_es\_aruhazak\_szama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_jovedelmi\_helyzet

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_jovedelmi\_helyzet` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`osszes\_ado` INT(11) NULL DEFAULT NULL,

`adofizetok\_szama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_kulturalis\_funkciok\_aranya

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_kulturalis\_funkciok\_aranya` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`kulturalis\_rendezvenyeken\_resztvevok\_szama` INT(11) NULL DEFAULT NULL,

`mozilatogatasok\_szama` INT(11) NULL DEFAULT NULL,

`muzeumi\_latogatok\_szama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_munkakepes\_koru\_nepesseg

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_munkakepes\_koru\_nepesseg` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`nepesseg` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_szabad\_iskolavalasztas\_valtozo

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_szabad\_iskolavalasztas\_valtozo` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`allando\_nepessegbol\_6\_13\_evesek\_szama` INT(11) NULL DEFAULT NULL,

`allando\_nepessegbol\_14\_evesek\_szama` INT(11) NULL DEFAULT NULL,

`allando\_nepessegbol\_15\_17\_evesek\_szama` INT(11) NULL DEFAULT NULL,

`nappali\_altalanos\_iskolai\_tanulok\_szama` INT(11) NULL DEFAULT NULL,

`nappali\_kozepiskolai\_tanulok\_szama` INT(11) NULL DEFAULT NULL,

`nappali\_szakiskolai\_es\_spec\_szakiskolai\_tanulok\_szama` INT(11) NULL DEFAULT NULL,

`mas\_telepulesrol\_bejaro\_nappali\_alt\_iskolai\_tanulok\_szama` INT(11) NULL DEFAULT NULL,

`mas\_telepulesrol\_bejaro\_nappali\_kozepiskolai\_tanulok\_szama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_telepules

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_telepules` (

`telepules\_nev` VARCHAR(100) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`KSH\_kod` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`jogallas` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`megye` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`kisterseg\_kod` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`kisterseg\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`kisterseg\_szekhelye` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`EOV\_X` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`EOV\_Y` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`WGS84\_lat` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`WGS84\_lob` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

PRIMARY KEY (`telepules\_nev`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_vandorlasi\_egyenleg\_aranya

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_vandorlasi\_egyenleg\_aranya` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`odavandorlasok\_szama` INT(11) NULL DEFAULT NULL,

`elvandorlasok\_szama` INT(11) NULL DEFAULT NULL,

`elveszuletesek\_szama` INT(11) NULL DEFAULT NULL,

`halalozasok\_szama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_belfoldi\_osszes\_jovedelem

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_belfoldi\_osszes\_jovedelem` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`jovedelem` DOUBLE NULL DEFAULT NULL,

PRIMARY KEY (`telepules\_nev`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_erintett\_nepesseg\_szama

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_erintett\_nepesseg\_szama` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`regisztralt\_bunelkovetok\_szama` INT(11) NULL DEFAULT NULL,

`haztartasi\_gazfogyasztok\_szama` INT(11) NULL DEFAULT NULL,

`haztartasi\_villamosenergia\_fogyasztok\_szama` INT(11) NULL DEFAULT NULL,

`elvandorlasok\_szama` INT(11) NULL DEFAULT NULL,

`allando\_nepesseg\_szama` INT(11) NULL DEFAULT NULL,

`vallalkozasok\_szama` INT(11) NULL DEFAULT NULL,

`nyilvantartott\_allaskeresok\_szama` INT(11) NULL DEFAULT NULL,

`lakasallomany` INT(11) NULL DEFAULT NULL,

`bolcsodebe\_beirt\_gyermekek\_szama` INT(11) NULL DEFAULT NULL,

`allando\_nepessegbol\_18\_59\_evesek\_szama` INT(11) NULL DEFAULT NULL,

`mobilelofizetesek\_szama` INT(11) NULL DEFAULT NULL,

`osszes\_adofizeto` INT(11) NULL DEFAULT NULL,

`osszes\_ado` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_keresztmetszeti\_forgalom

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_keresztmetszeti\_forgalom` (

`shapenr` INT(11) NULL DEFAULT NULL,

`partnr` INT(11) NULL DEFAULT NULL,

`nrparts` INT(11) NULL DEFAULT NULL,

`pointnr` INT(11) NULL DEFAULT NULL,

`nrpointS` INT(11) NULL DEFAULT NULL,

`x` DOUBLE NULL DEFAULT NULL,

`y` DOUBLE NULL DEFAULT NULL,

`measure` DOUBLE NULL DEFAULT NULL,

`kszam` VARCHAR(6) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`pkod` DOUBLE NULL DEFAULT NULL,

`kkod` VARCHAR(15) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`vvkod` VARCHAR(15) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`kszelv` VARCHAR(15) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`vszelv` VARCHAR(15) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`RSHOSSZ` DOUBLE NULL DEFAULT NULL,

`ANF` MEDIUMINT(9) NULL DEFAULT NULL,

`ANET` SMALLINT(6) NULL DEFAULT NULL,

`MOF` SMALLINT(6) NULL DEFAULT NULL,

`ONGJ` SMALLINT(6) NULL DEFAULT NULL,

`OJ` MEDIUMINT(9) NULL DEFAULT NULL,

`OMOT` MEDIUMINT(9) NULL DEFAULT NULL,

`EV` SMALLINT(6) NULL DEFAULT NULL,

`ASZ` INT(11) NULL DEFAULT NULL,

`BUSZCS` INT(11) NULL DEFAULT NULL,

`BUSZE` SMALLINT(6) NULL DEFAULT NULL,

`OBUSZ` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`NYSZER` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`POTKTGK` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`KTGK` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`NTGK` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`KNTGK` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`OTGK` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`SZGK` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`OSZGK` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`MKP` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`KPF` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`LASSU` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`SPEC` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`FMEGB` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`ADATFORR` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`SZAMLNAP` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`JELLEG1` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`JELLEG2` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`FMEGJ` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_lakonepesseg

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_lakonepesseg` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`nepesseg` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_nemzetkozi\_tranzitforgalom

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_nemzetkozi\_tranzitforgalom` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`hataratlepok\_szama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_szallashely

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_szallashely` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`fizetovendeglatas\_szallashelyeinek\_szama` INT(11) NULL DEFAULT NULL,

`osszes\_kereskedelmi\_szallashely\_ferohelyeinek\_szama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_telepules\_koordinata

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_telepules\_koordinata` (

`telepules\_nev` VARCHAR(100) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`EOV\_X` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`EOV\_Y` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`WGS84\_lat` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`WGS84\_lob` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

PRIMARY KEY (`telepules\_nev`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_cegautok

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_cegautok` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`szemelygepkocsik\_szama` INT(11) NULL DEFAULT NULL,

`termeszetes\_szemely\_altal\_uzemeltetett\_szemelygepkocsik\_szama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_ertekesites\_netto\_arbevetele

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_ertekesites\_netto\_arbevetele` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`ertekesites\_netto\_arbevetele` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_keresztmetszeti\_forgalom\_v1

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_keresztmetszeti\_forgalom\_v1` (

`partnr` INT(11) NOT NULL,

`nrparts` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`pointnr` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`nrpointS` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`x` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`y` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`measure` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`kezd` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`vege` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`kszam` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`pkod` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

PRIMARY KEY (`partnr`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_keresztmetszeti\_forgalom\_v1

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_keresztmetszeti\_forgalom\_v1` (

`partnr` INT(11) NOT NULL,

`nrparts` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`pointnr` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`nrpointS` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`x` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`y` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`measure` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`kezd` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`vege` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`kszam` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`pkod` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

PRIMARY KEY (`partnr`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_node\_attribs

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_node\_attribs` (

`shape\_id` INT(11) NULL DEFAULT NULL,

`kszam` VARCHAR(6) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`pkod` INT(11) NULL DEFAULT NULL,

`kkod` VARCHAR(15) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`vvkod` VARCHAR(15) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`kszelv` VARCHAR(15) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`vszelv` VARCHAR(15) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`RSHOSSZ` DOUBLE NULL DEFAULT NULL,

`ANF` MEDIUMINT(9) NULL DEFAULT NULL,

`ANET` SMALLINT(6) NULL DEFAULT NULL,

`MOF` SMALLINT(6) NULL DEFAULT NULL,

`ONGJ` SMALLINT(6) NULL DEFAULT NULL,

`OJ` MEDIUMINT(9) NULL DEFAULT NULL,

`OMOT` MEDIUMINT(9) NULL DEFAULT NULL,

`EV` SMALLINT(6) NULL DEFAULT NULL,

`ASZ` INT(11) NULL DEFAULT NULL,

`BUSZCS` INT(11) NULL DEFAULT NULL,

`BUSZE` SMALLINT(6) NULL DEFAULT NULL,

`OBUSZ` SMALLINT(6) NULL DEFAULT NULL,

`NYSZER` SMALLINT(6) NULL DEFAULT NULL,

`POTKTGK` MEDIUMINT(9) NULL DEFAULT NULL,

`KTGK` MEDIUMINT(9) NULL DEFAULT NULL,

`NTGK` MEDIUMINT(9) NULL DEFAULT NULL,

`KNTGK` MEDIUMINT(9) NULL DEFAULT NULL,

`OTGK` MEDIUMINT(9) NULL DEFAULT NULL,

`SZGK` MEDIUMINT(9) NULL DEFAULT NULL,

`OSZGK` MEDIUMINT(9) NULL DEFAULT NULL,

`MKP` MEDIUMINT(9) NULL DEFAULT NULL,

`KPF` MEDIUMINT(9) NULL DEFAULT NULL,

`LASSU` MEDIUMINT(9) NULL DEFAULT NULL,

`SPEC` MEDIUMINT(9) NULL DEFAULT NULL,

`FMEGB` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`ADATFORR` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`SZAMLNAP` SMALLINT(6) NULL DEFAULT NULL,

`JELLEG1` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`JELLEG2` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`FMEGJ` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

INDEX `index1` (`shape\_id` ASC))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_szemelygepkocsik

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_szemelygepkocsik` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`szemelygepkocsik\_szama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_tomegkozlekedes

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_tomegkozlekedes` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`vasutallomas\_lete` TINYINT(1) NULL DEFAULT NULL,

`kozvetlen\_jaratok\_szama\_naponta\_a\_megyeszekhelyre` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_egeszsegugyi\_szocialis\_ellatottak\_elvandoroltak` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`hazi\_gyermekorvosi\_ellatasban\_rendelesen\_megjelentek\_szama` INT(11) NULL DEFAULT NULL,

`hazi\_orvosi\_ellatasban\_rendelesen\_megjelentek\_szama` INT(11) NULL DEFAULT NULL,

`elbocsatott\_betegek\_szama` INT(11) NULL DEFAULT NULL,

`megjelenesi\_esetek\_szama` INT(11) NULL DEFAULT NULL,

`nappali\_ellatasban\_reszesulo\_fogyatekos\_szemelyek\_szama` INT(11) NULL DEFAULT NULL,

`nappali\_ellatasban\_reszesulo\_idoskoruak\_szama` INT(11) NULL DEFAULT NULL,

`nappali\_ellatasban\_reszesulo\_pszichiatriai\_betegek\_szama` INT(11) NULL DEFAULT NULL,

`nappali\_ellatasban\_reszesulo\_szenvedelybetegek\_szama` INT(11) NULL DEFAULT NULL,

`odavandorlasok\_szama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_foglalkoztatottak\_szama

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_foglalkoztatottak\_szama` (

`kisterseg\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`kisterseg\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`foglalkoztatottak\_szama` INT(11) NULL DEFAULT NULL,

`kistersegi\_foglalkoztatasi\_rata` DOUBLE NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_kollegium

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_kollegium` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`kollegiumban\_lako\_nappali\_alt\_tanulok\_szama` INT(11) NULL DEFAULT NULL,

`kollegiumban\_lako\_nappali\_kozepiskolai\_tanulok\_szama` INT(11) NULL DEFAULT NULL,

`kollegiumban\_lako\_nappali\_szakiskolai\_spec\_tanulok\_szama` INT(11) NULL DEFAULT NULL,

`kollegiumban\_lako\_felsofoku\_alap\_mesterkepzesben\_tanulok\_szama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_mukodo\_vallalkozasok\_szama

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_mukodo\_vallalkozasok\_szama` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`vallalkozasok\_szama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_nodes

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_nodes` (

`shape\_id` INT(11) NOT NULL,

`x` DOUBLE NOT NULL,

`y` DOUBLE NOT NULL,

`number` BIGINT(20) NULL,

INDEX `index2` (`shape\_id` ASC),

INDEX `index3` (`x` ASC, `y` ASC),

INDEX `index4` (`shape\_id` ASC, `x` ASC, `y` ASC))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_tanulasi\_celu\_utazasok\_altalanos\_jellemzese` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`nappali\_altalanos\_iskolai\_oktatas` INT(11) NULL DEFAULT NULL,

`nappali\_kozepiskolai\_es\_felnottoktatasi\_oktatas` INT(11) NULL DEFAULT NULL,

`felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama` INT(11) NULL DEFAULT NULL,

`nappali\_es\_nem\_nappali\_felsooktatas` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_ugyintezes\_altalanos\_jellemzese

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_ugyintezes\_altalanos\_jellemzese` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`egeszsegugyi\_ellatasok\_szama` INT(11) NULL DEFAULT NULL,

`postai\_gyogyszertari\_szolgaltatasok\_szama` INT(11) NULL DEFAULT NULL,

`buntetoperek\_szama` INT(11) NULL DEFAULT NULL,

`kereskedelmi\_szolgaltatasok\_szama` INT(11) NULL DEFAULT NULL,

`szocialis\_szolgaltatasok\_szama` INT(11) NULL DEFAULT NULL,

`okmanyirodai\_ugyintezesek\_szama` INT(11) NULL DEFAULT NULL,

`bankok\_szama` INT(11) NULL DEFAULT NULL,

`benzinkutak\_szama` INT(11) NULL DEFAULT NULL,

`korjegyzosegek\_foldhivatalok\_szama` INT(11) NULL DEFAULT NULL,

`kozmuszolgaltatasok` INT(11) NULL DEFAULT NULL,

`munkaugyi\_szolgaltatasok\_szama` INT(11) NULL DEFAULT NULL,

`bolcsodei\_szolgaltatasok\_szama` INT(11) NULL DEFAULT NULL,

`mobiltelefon\_ugyintezesi\_szolgaltatasok\_szama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_egyeb\_intezmenyi\_ellatottsag\_sulya

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_egyeb\_intezmenyi\_ellatottsag\_sulya` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`bankfiok\_lete` TINYINT(1) NULL DEFAULT NULL,

`benzinkut\_lete` TINYINT(1) NULL DEFAULT NULL,

`korjegyzoseg\_szekhelye` TINYINT(1) NULL DEFAULT NULL,

`postahivatal\_lete` TINYINT(1) NULL DEFAULT NULL,

`birosag\_ugyeszseg\_lete` TINYINT(1) NULL DEFAULT NULL,

`okmanyiroda\_lete` TINYINT(1) NULL DEFAULT NULL,

`megyei\_foldhivatal\_illetve\_kirendeltseg\_lete` TINYINT(1) NULL DEFAULT NULL,

`munkaugyi\_kozpont\_illetve\_kirendeltseg\_lete` TINYINT(1) NULL DEFAULT NULL,

`vodafone\_uzlet\_lete` TINYINT(1) NULL DEFAULT NULL,

`t\_mobile\_uzlet\_lete` TINYINT(1) NULL DEFAULT NULL,

`telenor\_uzlet\_lete` TINYINT(1) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_jovedelem\_es\_allaskeresok\_szama`

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_jovedelem\_es\_allaskeresok\_szama` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`osszes\_jovedelem\_foallasbol` INT(11) NULL DEFAULT NULL,

`allando\_nepesseg\_szama` INT(11) NULL DEFAULT NULL,

`p180\_napon\_tuli\_nyilvantartott\_allaskeresok\_szama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_kultura\_es\_sport

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_kultura\_es\_sport` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`filmszinhaz` TINYINT(1) NULL DEFAULT NULL,

`kozmuvelodesi\_intezmeny` TINYINT(1) NULL DEFAULT NULL,

`muzeum` TINYINT(1) NULL DEFAULT NULL,

`piac` TINYINT(1) NULL DEFAULT NULL,

`sportcsarnok\_sportpalya` TINYINT(1) NULL DEFAULT NULL,

`strand` TINYINT(1) NULL DEFAULT NULL,

`telepulesi\_konyvtar` TINYINT(1) NULL DEFAULT NULL,

`uszoda\_furdo\_gyogyfurdo` TINYINT(1) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_mukodo\_vallalkozasok\_szama\_a\_kiemelt\_iparagakban` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`feldolgozoiparban\_mukodo\_vallalkozasok\_szama` INT(11) NULL DEFAULT NULL,

`kereskedelemben\_gepjarmujavitasban\_mukodo\_vallalkozasok\_szama` INT(11) NULL DEFAULT NULL,

`szallitasban\_raktarozasban\_mukodo\_vallalkozasok\_szama` INT(11) NULL DEFAULT NULL,

`adminisztrativ\_es\_szolgaltatasban\_mukodo\_vallalkozasok\_szama` INT(11) NULL DEFAULT NULL,

`egyeb\_szolgaltatasban\_mukodo\_vallalkozasok\_szama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_oktatasi\_ferohelyek\_szama

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_oktatasi\_ferohelyek\_szama` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKod` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`nappali\_alt\_iskolai\_osztalyok\_szama` INT(11) NULL DEFAULT NULL,

`nappali\_kozepiskolai\_osztalyok\_szama` INT(11) NULL DEFAULT NULL,

`nappali\_szakiskolai\_spec\_szakiskolai\_osztalyok\_szama` INT(11) NULL DEFAULT NULL,

`nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben` INT(11) NULL DEFAULT NULL,

`nem\_nappali\_hallgatok\_szama\_felsofoku\_alap\_mesterkepzesben` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_tanulok\_es\_hallgatok\_szama

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_tanulok\_es\_hallgatok\_szama` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`nappali\_alt\_iskolai\_tanulok\_szama` INT(11) NULL DEFAULT NULL,

`felnottoktatasban\_alt\_iskolai\_tanulok\_szama` INT(11) NULL DEFAULT NULL,

`nappali\_kozepiskolai\_tanulok\_szama` INT(11) NULL DEFAULT NULL,

`felnottoktatasban\_kozepiskolai\_tanulok\_szama` INT(11) NULL DEFAULT NULL,

`nappali\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama` INT(11) NULL DEFAULT NULL,

`felnottoktatasban\_szakiskolai\_spec\_szakiskolai\_tanulok\_szama` INT(11) NULL DEFAULT NULL,

`nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama` INT(11) NULL DEFAULT NULL,

`nem\_nappali\_felsofoku\_alap\_mesterkepzesben\_hallgatok\_szama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci

### stg\_uzletek\_vendeglatohelyek\_gyogyszertarak

CREATE TABLE IF NOT EXISTS `mydb`.`stg\_uzletek\_vendeglatohelyek\_gyogyszertarak` (

`telepules\_nev` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NOT NULL,

`telepules\_KSHKOD` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_hungarian\_ci' NULL DEFAULT NULL,

`elelmiszer\_vegyesuzletek\_aruhazak\_szama` INT(11) NULL DEFAULT NULL,

`kiskereskedelmi\_uzletek\_szama` INT(11) NULL DEFAULT NULL,

`vendeglatohelyek\_szama` INT(11) NULL DEFAULT NULL,

`fiokgyogyszertarak\_szama` INT(11) NULL DEFAULT NULL,

`gyogyszertarak\_szama` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_hungarian\_ci