Add BO Class

Contents

[Abbreviations: 4](#_Toc1755247)

[Class Layout 4](#_Toc1755248)

[DataGrid View 4](#_Toc1755249)

[Item View 5](#_Toc1755250)

[Naming Convention 5](#_Toc1755251)

[Location in Project 5](#_Toc1755252)

[Data Table Base Structure 6](#_Toc1755253)

[Information 6](#_Toc1755254)

[Usings 6](#_Toc1755255)

[Class Header Attributes 6](#_Toc1755256)

[Class Structure 8](#_Toc1755257)

[Class Properties 8](#_Toc1755258)

[Base Attributes 8](#_Toc1755259)

[Common 8](#_Toc1755260)

[DisplayName 9](#_Toc1755261)

[DisplayGroup 9](#_Toc1755262)

[EditTemplate 9](#_Toc1755263)

[EditCssClass 9](#_Toc1755264)

[ViewCssClass 9](#_Toc1755265)

[*Order* 9](#_Toc1755266)

[PropertyDescription 9](#_Toc1755267)

[DB 10](#_Toc1755268)

[ParamName 10](#_Toc1755269)

[ParamSize 10](#_Toc1755270)

[ParamType 10](#_Toc1755271)

[Prefix 10](#_Toc1755272)

[Sort 10](#_Toc1755273)

[Access 10](#_Toc1755274)

[SearchableFor 10](#_Toc1755275)

[EditableFor 10](#_Toc1755276)

[VisibleFor 10](#_Toc1755277)

[DisplayMode 10](#_Toc1755278)

[Validation 11](#_Toc1755279)

[ValidationType 11](#_Toc1755280)

[None 11](#_Toc1755281)

[Required 11](#_Toc1755282)

[RegularExpression 11](#_Toc1755283)

[Function 11](#_Toc1755284)

[Confirmation 12](#_Toc1755285)

[RegularExpressionRequired 12](#_Toc1755286)

[AlertMessage 12](#_Toc1755287)

[Controls 12](#_Toc1755288)

[Name 12](#_Toc1755289)

[String 12](#_Toc1755290)

[Description 12](#_Toc1755291)

[ColorPicker 13](#_Toc1755292)

[Number 13](#_Toc1755293)

[Decimal 13](#_Toc1755294)

[Link 14](#_Toc1755295)

[LinkItems 14](#_Toc1755296)

[DropDown 14](#_Toc1755297)

[ParentDropDown 14](#_Toc1755298)

[SearchDropDown 14](#_Toc1755299)

[SelectList 15](#_Toc1755300)

[ParentSelectList 15](#_Toc1755301)

[SearchSelectList 15](#_Toc1755302)

[Parent 15](#_Toc1755303)

[DateTime 16](#_Toc1755304)

[Date 16](#_Toc1755305)

[DateTimeRange 17](#_Toc1755306)

[DateRange 17](#_Toc1755307)

[Email 18](#_Toc1755308)

[MultiCheck 18](#_Toc1755309)

[MultiSelect 18](#_Toc1755310)

[PermissionsSelector 19](#_Toc1755311)

[Image 19](#_Toc1755312)

[Document 20](#_Toc1755313)

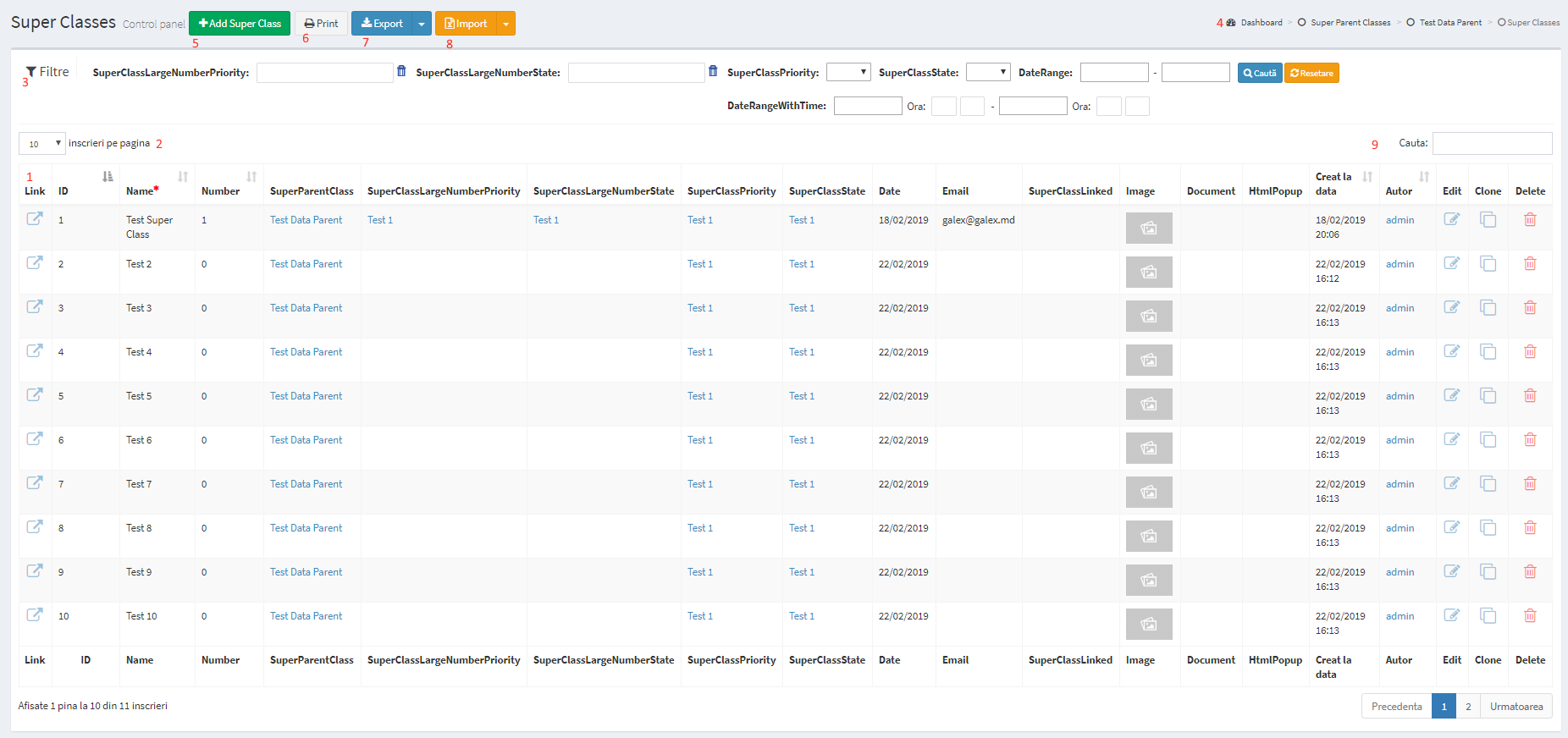
[Html 20](#_Toc1755314)

# Abbreviations:

* BO = business objects class
* CP = Control Panel
* SMI = System Management Interface
* PM = Project Name (ex MedCore)

# Class Layout

## DataGrid View



1 – DataGrid (Columns DisplayMode Simple)

2 – Paging

3 – Fitering (DisplayMode Search)

4 – BreadCrumbs

5 – Add new record

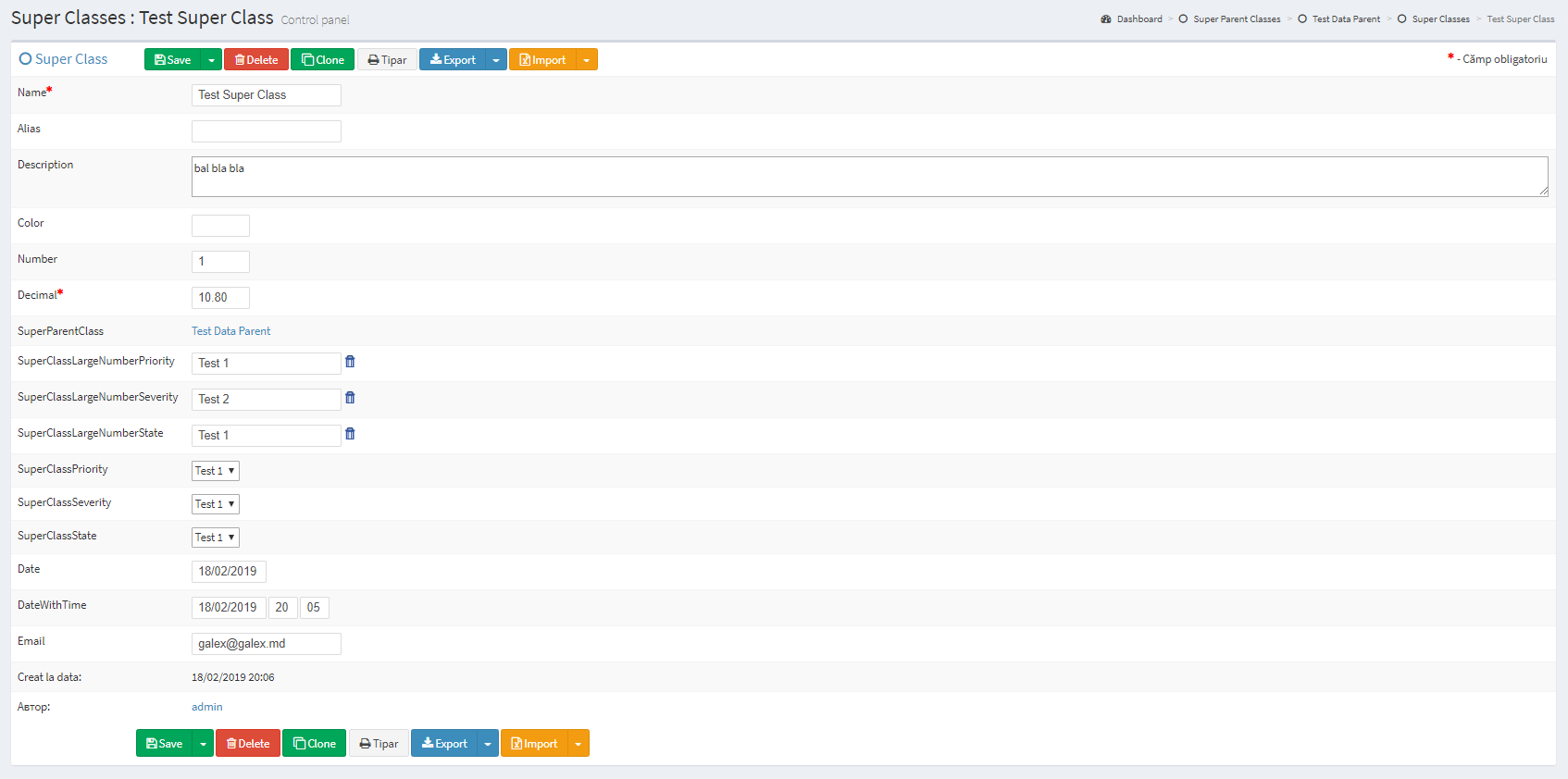
6 - Print All filtered Records(DisplayMode Print)

7- Export to Excell, CSV(DisplayMode Excell, CSV)

8 - Import – do not work

9 – Simple Search, by all text fields.

## Item View



# Naming Convention

For C# class names and corresponding table names are used only English singular terminology in Camel Case (<https://ru.wikipedia.org/wiki/CamelCase>). The same approach is used for properties and table columns.

For example, if it is required in project to add a Payment state with name and active flag. Should be used following names:

Class file: **PaymentState**.cs

Class name: **PaymentState**

Database table: **PaymentState**.sql

string Name

bool Active

Class name should be the same like table name. Class properties names should also be the same like Table columns names.

# Location in Project

BO could be added to 4 locations, based on class scope:

1. FrontEnd Models

\mvc\Models\Objects

1. FrontEnd Library Models

\mvc\Models\Lib

1. CP&SMI classes

\ PMLib\BusinessObjects

1. Base Classes

\Lib\BusinessObjects

# Data Table Base Structure

For Sample class object, should be added SampleClass.cs file and SampleClass.sql table with minimum set of columns:

CREATE TABLE [dbo].[**SampleClass**]

(

[**SampleClass**Id] BIGINT NOT NULL PRIMARY KEY IDENTITY,

// Other columns

[DeletedBy] BIGINT NULL,

[CreatedBy] BIGINT NOT NULL,

[DateCreated] DATETIME NOT NULL,

[DateUpdated] DATETIME NULL

)

# Information

Each Class should have standard Header with basic Information:

// --------------------------------------------------------------------------------------// <copyright file="SuperClass.cs" company="GALEXSTUDIO">

// Copyright © 2019

// </copyright>

// <summary>

// The SuperClass is a test class with all possible features and controls.

// </summary>

// --------------------------------------------------------------------------------------

# Usings

Minimum Using:

using System;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using LIB.Tools.Utils;

using LIB.AdvancedProperties;

using LIB.BusinessObjects;

using LIB.Tools.BO;

using LIB.Tools.AdminArea;

using System.Collections.Generic;

# Class Header Attributes

[Serializable]

[Bo(Group = AdminAreaGroupenum.Documents

, ModulesAccess = (long)(Modulesenum.ControlPanel)

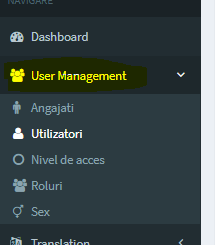
, DisplayName = "Statute Documente"

, SingleName = "Statut Document"

, DoCancel = true

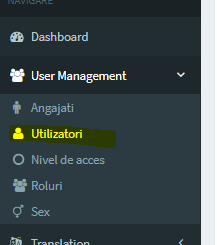
, LogRevisions = true)]

AdminAreaGroupenum – Could be defined for each Group Of Classes. Ex: UserManagement, Document Management, etc… If this attribute is not defined, then this class will not appear in left menu, but anyway can be accessed by URL. This could be used for child objects.



ModulesAccess – Coltrols Class visibility for project Modules. Ex. = (long)(Modulesenum.ControlPanel) will show this class only in Control Panel Module, but (long)(Modulesenum.ControlPanel | Modulesenum.SMI) in Both Control Panel and SMI.

DisplayName – Name how this class will be showed in Menu. Plural.



SingleName – Class Display Name singular.

ReadAccess – what permissions are required to have read access for class data.

CopyAccess– what permissions are required to have to copy class data.

PrintAccess– what permissions are required to have to print class data.

ExportAccess– what permissions are required to have to export class data.

ImportAccess– what permissions are required to have to import class data.

ReadAllAccess– what permissions are required to have read all(no paging) access for class data.

CreateAccess– what permissions are required to add record to class data.

DeleteAllAccess– what permissions are required to have delete all buttong access for class data.

DeleteAccess– what permissions are required to have delete access for class data.

RevisionsAccess– what permissions are required to have revisions access for class data.

RecordsPerPage – Default Records Per Page

MaxRecordsAllowed – Maximum records in this table(when limit is reached Add Button is not displayed)

DoCancel – This bool attribute defines if record will be permanently removed or only marked as deleted (DeletedBy will be set to User who deleted this record).

LogRevisions - This bool attribute defines if revisions will be automatically generated for this class.

# Class Structure

class SuperClass : ItemBase

{

#region Constructors

public SuperClass()

: base(0)

{

}

public SuperClass(long id)

: base(id)

{

}

#endregion

#region Properties

#endregion

}

Two constructors are required.

Regions are not obligatory, but is a good style to add it.

# Class Properties

## Base Attributes

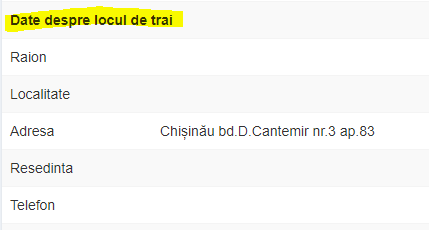
#GoFra framework have background logic for each class that is inherited from ItemBase, that allows objet to be Created, Populated, Canceled, Deleted, Updated In Database. Generated class from Post and Display class using on the controls in UI. All of this logic is based on predefined attributes of class and properties:

### Common

Set of attributes that are responsible for Common settings and display logic.

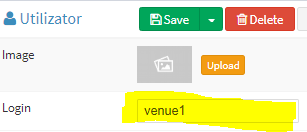
DisplayName - Common(DisplayName="Denumire") – By default in CP and SMI will be shown property name(English name in CamelCase), if it is needed to show friendly name it should be changed Display Name.

DisplayGroup – Display Group for a property. When it is set whole Group will be showed separately

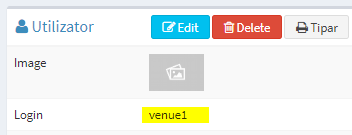


EditTemplate – Control that will be used for display this property.

EditCssClass – override Css Class for edit mode



ViewCssClass - override Css Class for view mode



### Order

Property sequence

PropertyDescription

advanced property information



### DB

ParamName – override property in database. By default property name should be the same like column name.

ParamSize – override size of database column (ex by default string have 200, that could be increased by this property).

ParamType – override param type(nvarchar, decimal, ect..) by default will be defined by property type.

Prefix – prefix used in case of duplicated parameters in one object

\_Ignore – do not try to CRUD object in database. In case of calculated or service object that does not have database column.

\_AllowNull – allow object to be null in database

\_Editable – Update/Insert property. True by default

\_Readable – populate and read(FromDataRow) property. True by default

\_Populate - populate property. True by default

Sort – In case is set will sort by this property by default.

### Access

SearchableFor - what permissions are required to be able to search/filter by this property.



EditableFor- what permissions are required to be able to edit this property.

VisibleFor- what permissions are required to be see this property.

DisplayMode- Enum that controls in what mode (Datagrid, ItemEdit, Filters, etc..) to show this property. Ex. *DisplayMode = DisplayMode.Simple | DisplayMode.FrontEnd | DisplayMode.Advanced | DisplayMode.Print*, that will make this property to be presented in datagrid,Item Layout, Filters and on Print.

### 

public enum DisplayMode : ulong

{

None,

Simple,

Advanced,

Search,

AdvancedEdit,

Prin,t

PrintSearch,

CSV,

Excell ,

FrontEnd ,

}

### Validation

ValidationType – Type of validation for this property:

public enum ValidationTypes : int

{

None = 0,

Required = 1,

RegularExpression = 2,

Function = 3,

Confirmation = 4,

RegularExpressionRequired = 5

}

None

No validation

Required

Should not be empty

RegularExpression

Should be validated by regular expression, that is set in *RegularExpression* attribute.

Ex email validation: ValidationType = ValidationTypes.RegularExpression, RegularExpression = @"^[a-z0-9\_\.-]+\@[a-z\_\.]+\.[\.a-z]+$"

##### Function

Validation by function

Ex Validate duplicated logins: Validation(ValidationType = ValidationTypes.Function, ValidationFunction = "ValidateUserName")

##### 

function ValidateUserName(pInut) {

var Value = $(pInut).val();

var result = false;

if ($.trim(Value) != "") {

var ID = ($(".data-item-container").size() > 0 ? $(".data-item-container").attr("data-id") : $($(pInut).closest("tr").find("td")[1]).html());

$.ajax({

type: "POST",

url: gRootUrl + "ValidationHelper/ValidateUserName/",

data: { Login: Value, Userid: ID },

async: false,

dataType: "json",

success: function (data) {

if (data["Result"] == 1) {

result = true;

gErrorMessage = data["Message"];

}

}

});

}

else {

return true;

}

return result;

}

##### Confirmation

Empty value will be allowed only after confirmation.

##### RegularExpressionRequired

Should pass regular expressions and not be empty.

#### AlertMessage

Message that will be shown in case when validation is failed.

## Controls

In order to make process of defining properties more simple there were generated a set of templates:

### Name

String that is Required

C#:

[Template(Mode = Template.Name)]

public string Name { get; set; }

SQL:

[Name] NVARCHAR(100) NOT NULL,

Result:



### String

String that is not Required and not shown in DataGrid

C#:

[Template(Mode = Template.String)]

public string Alias { get; set; }

SQL:

[Alias] NVARCHAR(100) NULL,

Result:



### Description

String that is not Required and not shown as textarea

C#:

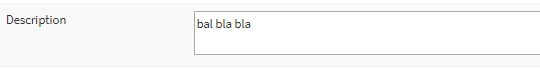
[Template(Mode = Template.Description)]

public string Description { get; set; }

SQL:

[Description] NVARCHAR(2000) NULL,

Result:



### ColorPicker

C#:

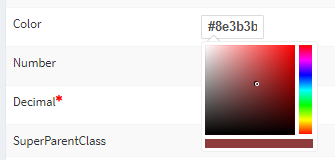
[Template(Mode = Template.ColorPicker]

public string Color { get; set; }

SQL:

[Color] NVARCHAR(100) NULL,

Result:



### Number

Number represented by input type number and validations

C#:

[Template(Mode = Template.Number)]

public int Number { get; set; }

SQL:

[Number] INT NULL,

Result:



### Decimal

Decimal number represented by input type number(allows decimal) and validations

C#:

[Template(Mode = Template.Decimal)]

public decimal Decimal { get; set; }

SQL:

[Decimal] DECIMAL(10,2) NULL,

Result:



### Link

Linked class. Relation – One to One. There is no way to set this object. It used only in case when it is set in SQL.

C#:

[Template(Mode = Template.Link)]

public SuperClass SuperClassLinked { get; set; }

SQL:

[SuperClassLinkedId] BIGINT NULL,

Result:



### LinkItems

Link to items that have Parent of this type

C#:

[Template(Mode = Template.LinkItems), LinkItem(LinkType = typeof(SuperClass))]

public Dictionary<long, ItemBase> SuperClasses { get; set; }

SQL:

No SQL

Result:



### DropDown

### ParentDropDown

### SearchDropDown

ComboBox that with list of available fields from another class. Make sense when there are less then 10-15 options.

C#:

[Template(Mode = Template.SearchDropDown)]

public SuperClassPriority SuperClassPriority { get; set; }

[Template(Mode = Template.DropDown)]

public SuperClassSeverity SuperClassSeverity { get; set; }

[Template(Mode = Template.ParentDropDown)]

public SuperClassState SuperClassState { get; set; }

SQL:

[SuperClassPriorityId] BIGINT NULL,

[SuperClassSeverityId] BIGINT NULL,

[SuperClassStateId] BIGINT NULL,

Result:



### SelectList

### ParentSelectList

### SearchSelectList

Automplete that with list of available fields from another class. Make sense when there are more then 10-15 options.

C#:

[Template(Mode = Template.SearchSelectList)]

public SuperClassLargeNumberPriority SuperClassLargeNumberPriority { get; set; }

[Template(Mode = Template.SelectList)]

public SuperClassLargeNumberSeverity SuperClassLargeNumberSeverity { get; set; }

[Template(Mode = Template.ParentSelectList)]

public SuperClassLargeNumberState SuperClassLargeNumberState { get; set; }

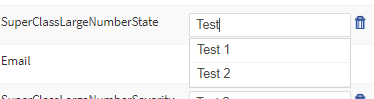
SQL:

[SuperClassLargeNumberPriorityId] BIGINT NULL,

[SuperClassLargeNumberSeverityId] BIGINT NULL,

[SuperClassLargeNumberStateId] BIGINT NULL,

Result:



### Parent

Parent Object that is not shown in layout.

C#:

[Template(Mode = Template.Parent)]

public SuperParentClass SuperParentClass { get; set; }

SQL:

[SuperParentClassId] BIGINT NULL,

Result:



### DateTime

Date Picker with validation with Minutes and Hours

C#:

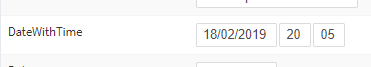
[Template(Mode = Template.DateTime)]

public DateTime DateWithTime { get; set; }

SQL:

[DateWithTime] DATETIME NULL,

Result:



### Date

Date Picker with validation

C#:

[Template(Mode = Template.Date)]

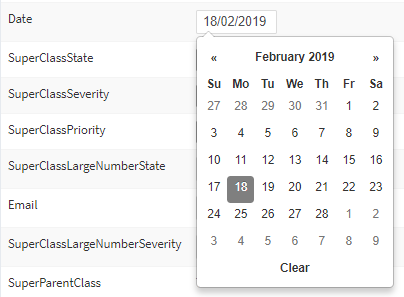
public DateTime Date { get; set; }

SQL:

[Template(Mode = Template.Date)]

public DateTime Date { get; set; }

Result:



### DateTimeRange

The same like DateTime, but with difference that it have filters by date range by default.

C#:

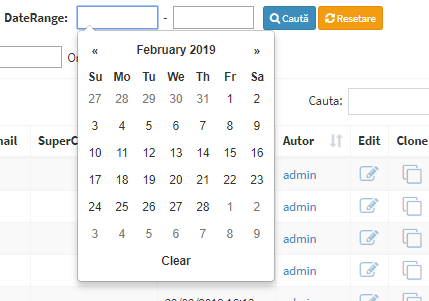
[Template(Mode = Template.DateTimeRange)]

public DateRange DateRangeWithTime { get; set; }

SQL:

[DateRangeWithTime] DATETIME NULL,

Result:



### DateRange

The same like Date, but with difference that it have filters by date range by default.

C#:

[Template(Mode = Template.DateRange)]

public DateRange DateRange { get; set; }

SQL:

[DateRange] DATETIME NULL,

Result:



### Email

Email input with validation.

C#:

[Template(Mode = Template.Email)]

public string Email { get; set; }

SQL:

[Email] NVARCHAR(50) NULL,

Result:



### MultiCheck

Multi Select Item with limited number of options. (recommended max 10)

C#:

[Template(Mode = Template.MultiCheck), MultiCheck(ItemType = typeof(ChildClass))]

public Dictionary<long, ItemBase> ChildClasses { get; set; }

SQL:

CREATE TABLE [dbo].[SuperClassChildClass]

(

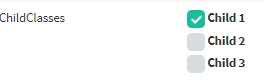
[SuperClassChildClassId] BIGINT NOT NULL PRIMARY KEY IDENTITY,

[SuperClassId] BIGINT NOT NULL,

[ChildClassId] BIGINT NOT NULL

)

Result:



### MultiSelect

String that is not Required and not shown in DataGrid

C#:

[Template(Mode = Template.MultiSelect), MultiCheck(ItemType = typeof(ChildClassMulty))]

public Dictionary<long, ItemBase> ChildClassMultys { get; set; }

SQL:

CREATE TABLE [dbo].[SuperClassChildClassMulty]

(

[SuperClassChildClassMultyId] BIGINT NOT NULL PRIMARY KEY IDENTITY,

[SuperClassId] BIGINT NOT NULL,

[ChildClassMultyId] BIGINT NOT NULL

)

Result:



### PermissionsSelector

Permission Selector

C#:

[Template(Mode = Template.PermissionsSelector)]

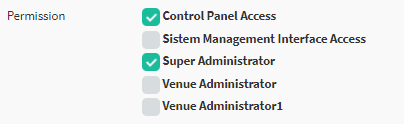
[Access(EditableFor = (long)BasePermissionenum.SuperAdmin, VisibleFor = (long)BasePermissionenum.SuperAdmin, DisplayMode = DisplayMode.Advanced)]

public long Permission { get; set; }

SQL:

[Permission] BIGINT NULL,

Result:



### Image

Image Upload control

C#:

[Template(Mode = Template.Image)]

public Graphic Image { get; set; }

SQL:

[ImageId] BIGINT NULL,

Result:



### Document

File Upload Control

C#:

[Template(Mode = Template.Document)]

public Document Document { get; set; }

SQL:

[DocumentId] BIGINT NULL,

Result:



### Html

WYIWYG Editor

C#:

[Template(Mode = Template.Html)]

public string Html { get; set; }

SQL:

[Html] NVARCHAR(MAX) NULL,

Result:

