Add Report List

Contents

[1. Introduction 2](#_Toc14184825)

[2. Add Class 3](#_Toc14184826)

[Type of attributes: 4](#_Toc14184827)

[3. Add Procedure 6](#_Toc14184828)

[4. Add Multiple Elements 7](#_Toc14184829)

[5. Front End Report List 9](#_Toc14184830)

[6. Attachments 10](#_Toc14184831)

[Class TestList 10](#_Toc14184832)

# 1. Introduction

The report list is composed of the list of data, enclosed in a table, and several filters used do search data by definite criteria. The elaboration of this part of the project consists of projecting the class and the procedure. The view is common for all reports. It is generated automatically.

\* Replace Test by your class

# 2. Add Class

1. Add a class in /Models/Reports/TestList.cs

[Bo(DisplayName = "Lista Test")] //Name of Report List

public class TestList : ReportBase

{

public TestList()

: base()

{

}

public override string GetLink() //link for editing chosen object

{

return "SystemManagement/EditItem/Test/GofraVersionsLib.BusinessObjects/" + TestId.ToString();

}

}

2. Add properties that will be used in the report:

public string Name { get; set; }

public User User { get; set; }

public Person Person { get; set; }

3. Add the attributes to each property:

[Common(Order = 1, DisplayName = "Name", \_Sortable = true, \_Searchable = true),

Template(Mode = Template.Name),

Access(DisplayMode = LIB.AdvancedProperties.DisplayMode.Simple

| LIB.AdvancedProperties.DisplayMode.Advanced

| LIB.AdvancedProperties.DisplayMode.Search

| LIB.AdvancedProperties.DisplayMode.Print)]

# Type of 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 = "State Name")

\_Searchable = true - possibility to make Search by records in column (true by default ).

\_Sortable = true - possibility to Sort column (false by default ).

Controls

In order to make process of defining properties more simple there were generated a set of templates (for exemple Name, String, Description, Number, Decimal, DropDown, Date etc.):

Template - Template(Mode = Template.Name)

Access

DisplayMode - Enum that controls in what mode (Datagrid, ItemEdit, Filters, etc.) to show this property.

DisplayMode = DisplayMode.Simple | DisplayMode.FrontEnd | DisplayMode.Advanced | DisplayMode.Print

It will make this property be present in datagrid, Item Layout, Filters and on Print.

4. Color the rows

Add into the class the following code to color the rows:

[Common(DisplayName = "Predefined Color"), Template(Mode = Template.DropDown), Access(DisplayMode = LIB.AdvancedProperties.DisplayMode.Advanced)]

public PredefinedColor PredefinedColor { get; set; }

public override string getConditionalClass()

{

if (PredefinedColor != null)

{

return " data-grid-data-row-type-" + PredefinedColor.Code;

}

return "";

}

The staining of the rows takes place in the css class Reports.css, in the method getConditionalClass() there is a change in style in the file Reports.css.

# 3. Add Procedure

CREATE VIEW [dbo].[TestList]

AS

SELECT

t.TestId

,t.Name

,t.TestTypeId

,t.Description

,tt.PredefinedColorId

FROM [Test] t

INNER JOIN TestType tt on tt.TestTypeId = t.TestTypeId

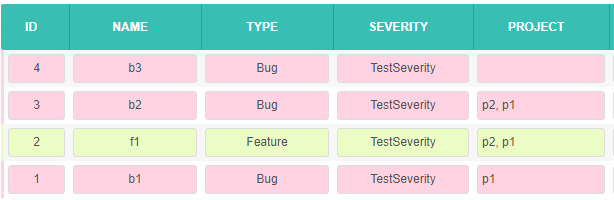
INNER JOIN PredefinedColor pc on pc.PredefinedColorId = tt.PredefinedColorId

WHERE t.DeletedBy IS NULL

\* PredefinedColor responds for the color of the rows. The rows will be colored in dependency of the procedure

# 4. Add Multiple Elements

The multiple elements are important when it is necessary to attach to any entity more objects, as is shown in the following example:

****

1. Add into namespace of the report the class and write a cross apply procedure by the following example:

namespace MedCore.Models.Reports

{

public class ListConst

{

public const string lists = "CROSS APPLY( SELECT t.Name +', ' " +

"FROM [TestFirst] tf " +

"INNER JOIN [Test] t ON t.TestFirstId = tf.TestFirstId " +

"WHERE p.DeletedBy IS NULL AND t.TestSecondId = ts.TestSecondId FOR XML PATH(''))[TestSeconds](TestSeconds) ";

}

}

2. Add before the main class this attribute:

[Bo(DisplayName = "List of Tests", AdditionalJoin = ListConst.lists, AfterPaginAdditionalQuery = ",[Tests].Tests")]

public class TestList : ReportBase

{

}

3. Add into the main class of report:

private string testFirsts;

[Common(Order = 4, DisplayName = "TestFirst", \_Sortable = true, \_Searchable = true, ViewCssClass = "investigations-list-report"), Template(Mode = Template.Name), Db(\_Populate = false),

Access(DisplayMode = LIB.AdvancedProperties.DisplayMode.Simple | LIB.AdvancedProperties.DisplayMode.Advanced | LIB.AdvancedProperties.DisplayMode.Print)]

public string TestFirsts

{

get

{

return testFirsts?.TrimEnd(' ', ',');

}

set

{

testFirsts = value;

}

}

[Common(DisplayName = "TestFirst", \_Sortable = true), Template(Mode = Template.SearchSelectList), Access(DisplayMode = LIB.AdvancedProperties.DisplayMode.Search),

LookUp(SearchQuery = "(SELECT t.TestFirstsId FROM [Test] t WHERE t.TestSecondId=ts.TestSecondId AND t.DeletedBy IS NULL)")]

public TestFirst TestFirst { get; set; }

# 5. Front End Report List



1. List name [Bo(DisplayName = “StudentList” )]

2. Set column for Preview or Print

3. Filters Access(DisplayMode = LIB.AdvancedProperties.DisplayMode.Search)

4. Column name (DisplayName = "State Name")

5. Report records

# 6. Attachments

## Class TestList

[Bo(DisplayName = "Lista Test")] //Name of Report List

public class TestList : ReportBase

{

public TestList()

: base()

{

}

[Common(Order = 1, DisplayName = "Name", \_Sortable = true, \_Searchable = true),

Template(Mode = Template.Name),

Access(DisplayMode = LIB.AdvancedProperties.DisplayMode.Simple

| LIB.AdvancedProperties.DisplayMode.Advanced

| LIB.AdvancedProperties.DisplayMode.Search

| LIB.AdvancedProperties.DisplayMode.Print)]

public string Name { get; set; }

[Common(Order = 1, DisplayName = "Name", \_Sortable = true, \_Searchable = true),

Template(Mode = Template.Name),

Access(DisplayMode = LIB.AdvancedProperties.DisplayMode.Simple

| LIB.AdvancedProperties.DisplayMode.Advanced

| LIB.AdvancedProperties.DisplayMode.Search

| LIB.AdvancedProperties.DisplayMode.Print)]

public User User { get; set; }

[Common(Order = 1, DisplayName = "Name", \_Sortable = true, \_Searchable = true),

Template(Mode = Template.Name),

Access(DisplayMode = LIB.AdvancedProperties.DisplayMode.Simple

| LIB.AdvancedProperties.DisplayMode.Advanced

| LIB.AdvancedProperties.DisplayMode.Search

| LIB.AdvancedProperties.DisplayMode.Print)]

public Person Person { get; set; }

public override string GetLink() //link for editing chosen object

{

return "SystemManagement/EditItem/Test/GofraVersionsLib.BusinessObjects/" + TestId.ToString();

}

[Common(DisplayName = "Predefined Color"), Template(Mode = Template.DropDown), Access(DisplayMode = LIB.AdvancedProperties.DisplayMode.Advanced)]

public PredefinedColor PredefinedColor { get; set; }

public override string getConditionalClass()

{

if (PredefinedColor != null)

{

return " data-grid-data-row-type-" + PredefinedColor.Code;

}

return "";

}

}