**Add to Enums\DashboardEnum.cs**

Test = 4 ( number , integer)

**Add a Dashboard Class in /Dashboards**

(TestDashBoard.cs)

* **Add Class that inherits from BaseDashboard**

*namespace MedCore.Dashboards*

*{*

*public class TestDashBoard : BaseDashboard*

*{*

*public DashboardEnum DashboardType = DashboardEnum.*Test*;*

*(***Implement Load )**

*}*

*}*

*(import all the missing files)*

* **Implement Load – function that is responsible for initial load**

public override void Load(ViewDataDictionary ViewData, User currentUser)

{

var context = new HttpContextWrapper(HttpContext.Current);

var currentPerson = (MedCoreLib.BusinessObjects.Person)context.Session[SessionItems.Person];

ViewData["List"] = Test.LoadLatest(currentPerson);

ViewData["GroupedList"] = TestState.LoadLatests(currentPerson.Id);

ViewData["Dashboard"] = DashboardType;

ViewData["DashboardControl"] = "\_TestList";

}

At the end of this , we will have a TestDashboard.cs in Dashboards

(full code)

----------------------------------------------------------------------------------

----------------------------------------------------------------------------------

using LIB.BusinessObjects;

using MedCoreweblib.Enums;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using weblib.Dashboards;

using LIB.Tools.Utils;

namespace MedCore.Dashboards

{

public class TestDashBoard : BaseDashboard

{

public DashboardEnum DashboardType = DashboardEnum.UserAnmin;

public override void Load(ViewDataDictionary ViewData, User currentUser)

{

var context = new HttpContextWrapper(HttpContext.Current);

var currentPerson = (MedCoreLib.BusinessObjects.Person)context.Session[SessionItems.Person];

ViewData["List"] = Test.LoadLatest(currentPerson);

ViewData["GroupedList"] = TestState.LoadLatests(currentPerson.Id);

ViewData["Dashboard"] = DashboardType;

ViewData["DashboardControl"] = "\_TestList";

}

}

}

----------------------------------------------------------------------------------

----------------------------------------------------------------------------------

**Now we will implement LoadLatest**

**Create a Test.cs in Models/Objects**

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Web.Mvc;

using LIB.AdvancedProperties;

using LIB.BusinessObjects;

using LIB.Connectors;

using LIB.Tools.AdminArea;

using LIB.Tools.BO;

using LIB.Tools.Utils;

using MedCoreLib.BusinessObjects;

using Nest;

using DisplayMode = LIB.AdvancedProperties.DisplayMode;

using Person = LIB.BusinessObjects.Person;

using User = LIB.BusinessObjects.User;

namespace MedCore.Models.Objects

{

[Bo(Group = AdminAreaGroupenum.Test

, ModulesAccess = (long)(Modulesenum.ControlPanel)

, DisplayName = "TestS"

, SingleName = "Test")]

public class Test : PrintBase

{

#region Constructors

/// <summary>

/// Initializes a new instance of the <see cref="LIB.AdvancedProperties.DisplayMode"/> class.

/// </summary>

public Test()

: base(0)

{

}

/// <summary>

/// Initializes a new instance of the <see cref="LIB.AdvancedProperties.DisplayMode"/> class.

/// </summary>

/// <param name="id">

/// The id.

/// </param>

public Test(long id)

: base(id)

{

}

#endregion

[Template(Mode = Template.Name)]

public string Name { get; set; }

[Template(Mode = Template.DateTime)]

public DateTime DateStart { get; set; }

[Template(Mode = Template.Description)]

public string ShortDescription { get; set; }

[Template(Mode = Template.Html)]

public string Description { get; set; }

[Template(Mode = Template.Image)]

public Graphic Photo { get; set; }

[Template(Mode = Template.SearchDropDown)]

public Person Owner { get; set; }

[Template(Mode = Template.ParentDropDown)]

public TestState TestState { get; set; }

[Template(Mode = Template.CheckBox)]

public bool Active { get; set; }

[Common(EditTemplate = EditTemplates.Hidden), Db(\_Editable = false, \_Populate = false)]

public DateTime Timestamp => DateStart;

public override string LoadReport(ControllerContext ControllerContext, ViewDataDictionary ViewData, TempDataDictionary TempData,

ExportType type = ExportType.None)

{

throw new NotImplementedException();

}

public static List<Test> LoadLatest(Person pOwner, long lastestId = 0)

{

var conn = DataBase.ConnectionFromContext();

var tests = new List<Test>();

var cmd = new SqlCommand("Test\_Populate\_Latests", conn) { CommandType = CommandType.StoredProcedure };

cmd.Parameters.Add(new SqlParameter("@OwnerId", SqlDbType.BigInt) { Value = pOwner.Id });

using (var rdr = cmd.ExecuteReader(CommandBehavior.SingleResult))

{

while (rdr.Read())

{

var od = (Test)(new Test()).FromDataRow(rdr);

tests.Add(od);

}

rdr.Close();

}

return tests;

}

public override void Insert(ItemBase item, string Comment = "Created", SqlConnection connection = null, User user = null)

{

base.Insert(item, Comment, connection, user);

//var indexName = this.GetType().Name.ToLower();

//var esclient = ElasticSearchConnector.Client;

//if (!esclient.IndexExists(indexName).Exists)

//{

// esclient.CreateIndex(indexName, c => c

// .Mappings(m => m

// .Map<Venue>(p => p.Properties(x =>

// x.Date(d => d.Name(e => e.DateStart))

// .Keyword(d => d.Name(e => e.Name))

// .Keyword(d => d.Name(e => e.Description)))

// .AutoMap()

// )

// )

// );

//}

//esclient.Index(this, i => i.Index(indexName));

}

public override void Update(ItemBase item, DisplayMode DisplayMode = DisplayMode.Advanced, string Comment = "Updated",

SqlConnection connection = null)

{

base.Update(item, DisplayMode, Comment, connection);

//var indexName = this.GetType().Name.ToLower();

//ElasticSearchConnector.Client.Update(DocumentPath<Venue>.Id(this.Id), u => u.Index(indexName).Doc(this));

}

public override bool Delete(Dictionary<long, ItemBase> dictionary, out string Reason, string Comment = "", SqlConnection connection = null,

User user = null)

{

//var indexName = this.GetType().Name.ToLower();

//ElasticSearchConnector.Client.Delete(DocumentPath<Venue>.Id(this.Id), u => u.Index(indexName));

return base.Delete(dictionary, out Reason, Comment, connection, user);

}

}

}

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**in Database/dbo/Tables/BO/Test**

**-**for Test.cs create Test.sql

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE TABLE [dbo].[Test]

(

[TestId] BIGINT NOT NULL PRIMARY KEY IDENTITY,

[Name] NVARCHAR(100) NOT NULL,

[DateStart] DATETIME NULL,

[ShortDescription] NVARCHAR(2000) NULL,

[Description] NVARCHAR(MAX) NULL,

[PhotoId] BIGINT NULL,

[OwnerId] BIGINT NULL,

[TestStateId] BIGINT NULL,

[Active] BIT NULL,

[CreatedBy] BIGINT NOT NULL,

[DateCreated] DATETIME NOT NULL,

[DeletedBy] BIGINT NULL

)

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**-**for TestState.cs create TestState.sql

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE TABLE [dbo].[TestState]

(

[TestStateId] BIGINT NOT NULL PRIMARY KEY IDENTITY,

[Name] NVARCHAR(100) NOT NULL,

[PredefinedColorId] BIGINT NULL,

[Active] BIT NULL,

[CreatedBy] BIGINT NOT NULL,

[DateCreated] DATETIME NOT NULL,

[DeletedBy] BIGINT NULL

)

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**in Database/Stored Procedures/Test**

**create** Test\_Populate\_Latests.sql

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE PROCEDURE [dbo].[Test\_Populate\_Latests]

@OwnerId bigint

AS

BEGIN

SELECT v.TestId,v.Name,v.DateStart

,v.TestStateId,vs.Name as TestStateName

,pc.PredefinedColorId, pc.Code as PredefinedColorCode

FROM Test v

INNER JOIN TestState vs ON vs.TestStateId=v.TestStateId

INNER JOIN PredefinedColor pc ON pc.PredefinedColorId=vs.PredefinedColorId

WHERE v.OwnerId = @OwnerId AND v.DeletedBy IS NULL

END

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**for TestStates**

create TestState\_Populate\_Latests.sql

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE PROCEDURE [dbo].[TestState\_Populate\_Latests]

@OwnerId bigint

AS

BEGIN

SELECT vs.TestStateId,vs.Name,COUNT(v.TestId) as TestsCount

,pc.Color

FROM [Test] v

INNER JOIN [TestState] vs ON vs.TestStateId=v.TestStateId

INNER JOIN PredefinedColor pc ON pc.PredefinedColorId=vs.PredefinedColorId

WHERE v.DeletedBy iS NULL AND vs.DeletedBy IS NULL

GROUP BY vs.TestStateId,vs.Name,pc.Color

END

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

AFTER WE CREATED THE TABLES IN DB WE HAVE TO PUBLISH IT .

---RIGHT CLICK ON Databases -> Publish..

target datanase connection

* **Server Name -> mysql Server (ip /host)**
* **Authentication -> SQL Server Authenticaion**
* **Database Name: (the database where we want to publish ) in our case MedCore**

**After that we have to go to --> Advanced... and unselect Block incremental deployement if data loss might occur , then press Ok**

**And now we can publish it (Press Publish)**

**If we want TestState we create a TestState as BusinessObjects**

**create in MedCoreLib/BusinessObjects/Test/TestState.cs**

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using LIB.AdvancedProperties;

using LIB.BusinessObjects;

using LIB.Tools.AdminArea;

using LIB.Tools.BO;

using LIB.Tools.Utils;

namespace MedCoreLib.BusinessObjects

{

[Bo(Group = AdminAreaGroupenum.Test

, ModulesAccess = (long) (Modulesenum.ControlPanel)

, DisplayName = "Test Statuses"

, SingleName = "Test Status")]

public class TestState : AggregateBase

{

#region Constructors

/// <summary>

/// Initializes a new instance of the <see cref="DisplayMode"/> class.

/// </summary>

public TestState()

: base(0)

{

}

/// <summary>

/// Initializes a new instance of the <see cref="DisplayMode"/> class.

/// </summary>

/// <param name="id">

/// The id.

/// </param>

public TestState(long id)

: base(id)

{

}

#endregion

public override int GetCount()

{

return TestCount;

}

#region Properties

[Template(Mode = Template.Name)]

public string Name { get; set; }

[Common(EditTemplate = EditTemplates.Hidden), Db(\_Editable = false, \_Populate = false)]

public int TestCount { get; set; }

[Template(Mode = Template.DropDown)]

public PredefinedColor PredefinedColor { get; set; }

#endregion

public static Dictionary<long, AggregateBase> LoadLatests(long pOwner)

{

var conn = DataBase.ConnectionFromContext();

var testStates = new Dictionary<long, AggregateBase>();

var cmd = new SqlCommand("TestState\_Populate\_Latests", conn) { CommandType = CommandType.StoredProcedure };

if (pOwner > 0)

cmd.Parameters.Add(new SqlParameter("@OwnerId", SqlDbType.BigInt) { Value = pOwner });

using (var rdr = cmd.ExecuteReader(CommandBehavior.SingleResult))

{

while (rdr.Read())

{

var state = (TestState)(new TestState()).FromDataRow(rdr);

state.Color = rdr["Color"].ToString();

testStates.Add(state.Id, state);

}

rdr.Close();

}

return testStates;

}

}

}

--------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Add in lib/tools/AdminArea/**AdminAreaGroupenum

Add Test

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

namespace LIB.Tools.AdminArea

{

/// <summary>

/// The Permission Enumerator.

/// </summary>

public enum AdminAreaGroupenum

{

None,

UserManagement,

Translate,

Settings,

Documents,

Rapoarte,

Navigation,

System,

Venue,

Fitness,

Test //added adminAreaGroupenum

}

}

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

* **Add DashBoardController Logic: DashBoardController.cs**

(you need to create TestAdmin , or other user, that can access the created dashboard )

else if (currentUser.HasAtLeastOnePermission((long)BasePermissionenum.TestAdmin))

{

return new TestDashBoard();

}

* **Add List View ex(\_TestList.cshtml)**

**MedCore/Views/DashBoard/\_TestList.cshtml**

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

@using LIB.BusinessObjects

@using MedCore.Models.Objects

@if (((List<Test>)ViewData["List"]).Count > 0)

{

<script type="text/javascript">

var gLatestDashboardId = @(((List<Test>)ViewData["List"]).First(i => i.Id>0).Id);

var gDashboardItemsCount = @(((List<Test>)ViewData["List"]).Count);

</script>

<form class="table-widget">

<div class="dashboard-ico">

<div class="outer-ico">

<div class="widget-caption">

Tests

</div>

<div class="data-grid-container">

<div class="data-grid">

<div class="data-grid-title-row">

<div class="data-grid-title">N</div>

<div class="data-grid-title">Name</div>

<div class="data-grid-title">Date Start</div>

<div class="data-grid-title">Curent State</div>

</div>

@Html.Partial("Controls/\_TestsRows")

</div>

</div>

</div>

</div>

<div class="clear" style="height:1px"></div>

</form>

}

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

* **Add Row View (\_TestsRows.cshtml)**

**MedCore/Views/DashBoard/Controls/(\_TestsRows.cshtml**

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

@using MedCore.Models.Objects

@using MedCoreLib.BusinessObjects

@{

Layout = "~/Views/Master/\_Ajax.cshtml";

}

@{ var index = ((List<Test>)ViewData["List"]).Count + (int)ViewData["DashboardListCount"];}

@foreach (var test in ((List<Test>)ViewData["List"]))

{

<a href="@LIB.Tools.Utils.URLHelper.GetUrl("DocControl/Test/" + test.Id )" target="\_blank" class="data-grid-data-row data-grid-data-row-@(test.Id) data-grid-data-row-type-@(test.TestState.PredefinedColor.Code)">

<div class="data-grid-data"><b>@(test.Id)</b></div>

<div class="data-grid-data">@(test.Name)</div>

<div class="data-grid-data">@(test.DateStart.ToString("dd.MM.yyyy HH:mm"))</div>

<div class="data-grid-data">@(test.TestState.Name)</div>

<input type="hidden" name="widgetitems" value="@(test.Id):@(test.TestState.Id)" />

</a>

}

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**In MedcoreLib/Tools/AdminArea/AdminAreaGroup.cs**

**add**

public static AdminAreaGroup Test = new AdminAreaGroup() { Group = AdminAreaGroupenum.Test, Name = LIB.Tools.Utils.Translate.GetTranslatedValue("Test", "BO") };

//////////////////////////////

list.Add(Test.Group, Test);