# **Nuget:**

# **Microsoft.AspNet.EntityDataSource**

* Microsoft.AspNet.FriendlyUrls
* System.Net.Http.Formatting 5.2.3
* WebFormsMVP

# **Local DB Server name:** (LocalDb)\MSSQLLocalDB

Bug fix .edmx file => ProviderManifestToken="20~~12~~"2008

Page

# Application Structure

AppName.Data => dbContext, Migration, Repository, UoF

AppName.Data.Models => dbModels

AppName.Data.Services

AppName.Web

# DB first for Web Forms

Remove User

|  |  |
| --- | --- |
| [NotMapped](http://www.entityframeworktutorial.net/code-first/notmapped-dataannotations-attribute-in-code-first.aspx) | Specify that property will not be mapped with database |

[Column(TypeName = "ntext")]

[RegularExpression(Constants.EnBgDigitSpaceMinus)]

[ForeignKey("Category")]

public int CategoryId { get; set; }

public virtual Category Category { get; set; }

IDbContext

int SaveChanges(); => to go to Unit of work

IDbSet<User> Users { get; set; } => for each table

DbSet<TEntity> Set<TEntity>() where TEntity : class;

DbEntityEntry<TEntity> Entry<TEntity>(TEntity entity) where TEntity : class;

Migration from PM manager => enable-migrations

After migration => configuration => from internal to public

public Configuration()

{

AutomaticMigrationsEnabled = true;

}

protected override void Seed(NewsSystem.Data.NewsSystemDbContext context)

{

if (context.Articles.Any())

{

return;

}

var user = new User()

{

UserName = "Kolkoto"

};

var seed = new SeedData(user); => from .cs data file

context.Users.Add(user);

context.SaveChanges();

seed.Categories.ForEach(x => context.Categories.Add(x));

seed.Articles.ForEach(x => context.Articles.Add(x));

context.SaveChanges();

}

**App\_Start = DbConfig file**

public static void Initilize()

{

Database.SetInitializer(new MigrateDatabaseToLatestVersion<NewsSystemDbContext, Configuration>());

NewsSystemDbContext.Create().Database.Initialize(true);

}

**Global.asax =>**  DbConfig.Initilize();

Start the App and DB will be created

connectionString=".;Initial Catalog=OnLineShop

<connectionStrings>

<add name="DefaultConnection" connectionString="Data Source=(LocalDb)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\aspnet-NewsSystem.Web-20170202110702.mdf;Initial Catalog=aspnet-NewsSystem.Web-20170202110702;Integrated Security=True; **MultipleActiveResultSets=true "** providerName="System.Data.SqlClient" />

</connectionStrings>

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

modelBuilder.Conventions.Remove<OneToManyCascadeDeleteConvention>();

modelBuilder.Conventions.Remove<ManyToManyCascadeDeleteConvention>();

base.OnModelCreating(modelBuilder);

}

# MVP Pattern

1. Class PageNameViewModel

public class CategoryViewModel

{

public IEnumerable<Category> Categories { get; set; }

}

1. Interface IPageNameViwe : IView< PageNameViewModel> => from WebFormsMVP –държи модела на нашата страничка или всички подмодели при по сложнo view и е generic

public interface ICategoryView : IView<CategoryViewModel>

{

event EventHandler myInit;

or if have custom event args event EventHandler< PageNameEventArgs> myInit;

}

1. Един Predsenter отговаря за едно view

Той знае чрез кого да го свърши а не как

public class HomePresenter : Presenter<ICategoryView> -> from WebFormsMVP

{

private readonly ICategoryView view;

private readonly ICategoryDataProvider categoryProvider; -> from service

public HomePresenter(ICategoryView view, ICategoryDataProvider categoryProvider)

: base(view)

{

this.view = view;

this.categoryProvider = categoryProvider;

this.view.myInit += View\_Init;

}

private void View\_Init(object sender, EventArgs e)

{

this.view.**Model(**from WebFormsMVP **)**.Categories = this.categoryProvider.GetAll();

}

}

Този модел вече се е напълнил с данни от PageNameViewModel

PageName.aspx ~~: Page :~~ MvpPage<CategoryViewModel>, ICategoryView

Presenter го регистрираме в MvpNinjectModul

[PresenterBinding(typeof(HomePresenter))]

Чрез атрибут на на PageName.aspx казваме към кой presenter е свързана нашата станичка

(или името на страницата = името на presenter HomePage => HomePresenter

protected void Page\_Load(object sender, EventArgs e)

{

this.myInit?.Invoke(sender, e);

Raise event за да знае presenter от къде да си напълни данните

}

Unit Tests => services and presenters

Да проверим ако на подадем на presenter provider дали се извиква метода който трябва

На presenter можем да подаваме толкова providers колкото са ни нужни за да си свършим работата

# **Ако взимаме параметри от request**

Public class PageNameEventArgs :EventArgs

{

Public int ID{get; private set;}

Public PageNameEventArgs(int id)

{

This.id=id;

}

}

# Config API Server for JS files

All controllers are in folder Controllers and have the route name

public IHttpActionResult GetEmployee(int id)

{

var employee = data.GetEmployeeById(id).FirstOrDefault();

if (employee != null)

{

var response = new

{

Photo = employee.Photo,

Phone = employee.HomePhone,

Email = employee.Fullname.ToLower()+"@nortwind.com",

};

return Ok(response);

}

else

{

return NotFound();

}

} Address=employee.Address,

Notes=employee.Notes

**App\_Start = > new WebApiConfig file**

public static void Register(HttpConfiguration config)

{

config.MapHttpAttributeRoutes();

config.Routes.MapHttpRoute(

name: "DefaultApi",

routeTemplate: "api/{controller}/{id}",

defaults: new { id = RouteParameter.Optional }

);

}

**Global.asax**

**Nuget:** **System.Net.Http.Formatting.Extention**

GlobalConfiguration.Configure(WebApiConfig.Register);

<script>

$(".name").on("mouseover", function (event) {

console.log(event.target.innerText);

$("#employeeName").val(event.target.innerText);

$("#details").removeClass("hidden");

$("#btns").trigger("click");

});

$(".name").on("mouseout", function () {

$("#details").addClass("hidden");

});

</script>

# Ninject Configuration

Nuget: NinjectWeb, Ninject Extentions

App\_Start => NinjectWebCommon

private static void RegisterServices(IKernel kernel)

{

kernel.Bind<INewsSystemDbContext>().To<NewsSystemDbContext>().InSingletonScope();

kernel.Bind(typeof(IRepository<>)).To(typeof(GenericRepository<>)).InSingletonScope();

kernel.Bind(x =>

x.From("NewsSystem.Data.Services")

.SelectAllClasses()

.BindDefaultInterface());

}

In Code behind

[Inject]

public IArticleServices ArticleServeces { get; set; }

# Data Bindings Controls

**Repeater**

Useful when you want to implement a non-standard visualization of read-only data

<asp:Repeater

ID="MostPopulerArticles"

runat="server"

ItemType="NewsSystem.Data.Models.Article"

SelectMethod="MostPopulerArticles\_GetData">

</asp:Repeater>

Taka se vzima ID v code behind

public Article DetailsViewGetItem([QueryString] string id)

{

return this.ArticleServeces.GetById(int.Parse(id)); }

**Form view**

<asp:FormView

ID="FormView1"

runat="server"

ItemType="NewsSystem.Data.Models.Article"

SelectMethod="DetailsViewGetItem">

<HeaderTemplate></HeaderTemplate>

<ItemTemplate></ItemTemplate>

</asp:FormView>

* You are responsible to define some or all of the templates
  + ItemTemplate
  + InsertItemTemplate
* You could change the view mode at run-time:

*this.FormViewCustomer.ChangeMode(FormViewMode.Edit);*

**ListView**

<asp:ListView

ID="ArtivlesLV"

runat="server"

DataKeyNames="Id"

ItemType="NewsSystem.Data.Models.Article"

SelectMethod="ArtivlesLV\_GetData"

UpdateMethod="ArtivlesLV\_UpdateItem"

DeleteMethod="ArtivlesLV\_DeleteItem"

InsertMethod="ArtivlesLV\_InsertItem"

InsertItemPosition="LastItem">

<LayoutTemplate>

// To implement functionality in code behind

<asp:HyperLink NavigateUrl="?orderBy=Title" Text="Sort by Title" runat="server" CssClass="btn btn-md-2 btn-default" />

<asp:HyperLink NavigateUrl="?orderBy=DateCreated" Text="Sort by Date" runat="server" CssClass="btn btn-md-2 btn-default" />

<asp:HyperLink NavigateUrl="?orderBy=Category.Name" Text="Sort by Category" runat="server" CssClass="btn btn-md-2 btn-default" />

<asp:HyperLink NavigateUrl="?orderBy=Likes.Count()" Text="Sort by Likes" runat="server" CssClass="btn btn-md-2 btn-default" />

<div runat="server" id="itemPlaceholder"></div>

<asp:DataPager runat="server" PageSize="3" PagedControlID="ArtivlesLV">

<Fields>

<asp:NextPreviousPagerField ShowPreviousPageButton="true" ShowNextPageButton="false" ButtonCssClass="btn btn-success" />

<asp:NumericPagerField />

<asp:NextPreviousPagerField ShowPreviousPageButton="false" ShowNextPageButton="true" ButtonCssClass="btn btn-success" />

</Fields>

</asp:DataPager>

</LayoutTemplate>

<ItemTemplate>

<h2><%#:Item.Title %>

<asp:Button runat="server" Text="Edit" CommandName="Edit" CssClass="btn btn-info" />

<asp:Button runat="server" Text="Delete" CommandName="Delete" CssClass="btn btn-danger" />

</h2>

<p>Category: <%#:Item.Category.Name %></p>

<p><%#: Item.Content.Length>300? Item.Content.Substring(0,300)+"...": Item.Content %></p>

<p>Likes count: <%#:Item.Likes.Count %></p>

<i>by <%#: Item.Author.UserName %> created on: <%#: Item.DateCreated %></i>

</ItemTemplate>

<EditItemTemplate>

</EditItemTemplate>

<InsertItemTemplate>

</InsertItemTemplate>

</asp:ListView>**Paging in List View**

<asp:DataPager runat="server" PageSize="5">

<Fields>

<asp:NextPreviousPagerField ShowPreviousPageButton="true" ShowNextPageButton="false" ButtonCssClass="btn btn-success" />

<asp:NumericPagerField />

<asp:NextPreviousPagerField ShowPreviousPageButton="false" ShowNextPageButton="true" ButtonCssClass="btn btn-success" />

</Fields>

</asp:DataPager>

# **GridView**

* GridView displays tabular data as HTML table
  + Consists of columns, header and footer
  + Columns can be auto-generated according to the data source or can be set explicitly
  + Supports paging, sorting, editing and deleting
  + Set AutoGenerateColumns to false to customize the columns in the GridView

<asp:BoundField DataField="FirstName" HeaderText="First Name">

**Details View**

* Displays a single record
  + Usually used along with GridView
* Supports paging, inserting, updating, deleting
* Uses the same fields as GridView
  + Declared in a <Fields>element
* Easy to change the appearance
* Can auto-generate fields:
  + AutoGenerateRows="true"

**Tree View**

<asp:TreeView ID="TreeView" runat="server" DataSourceID="XmlDataSourceFoodMenu">

<DataBindings>

<asp:TreeNodeBinding DataMember="name" TextField="#InnerText"/>

<asp:TreeNodeBinding DataMember="price" TextField="#InnerText"/>

<asp:TreeNodeBinding DataMember="description" TextField="#InnerText"/>

<asp:TreeNodeBinding DataMember="calories" TextField="#InnerText"/>

</DataBindings>

</asp:TreeView>

**Different Roles**

|  |
| --- |
| <location path="Private"> |
|  | <system.web> |
|  | <authorization> |
|  | <deny users="?" /> |
|  | </authorization> |
|  | </system.web> |
|  | </location> |

**Create folder Private**

**Data Source Controls**

* ASP.NET provides server controls that take care of data binding details
  + Known as data source controls
  + SqlDataSource, EntityDataSource,ObjectDataSource, XmlDataSource, …
* They are an abstraction over the data source
* Data-bound server controls can be associated to a data source control
  + Through the DataSourceID property

## **Other Helpers:**

Redirektvani kym teku6tata stranica

this.Response.Redirect(this.Request.RawUrl);

using Microsoft.AspNet.Identity;

articleToInsert.AuthorId = Page.User.Identity.GetUserId();

## Bootstrap classes

class="pull-right"

Имплецитна имплементация на Интерфейс – достъп до обекта само през интерфейс

public interface IDbContext

{

IDbSet<Employee> Employees { get; }

}

public class MyNorthwindDbContext : NorthwindEntities, IDbContext

{

IDbSet<Employee> IDbContext.Employees

{

get

{

return base.Employees;

}

}

}

public const string DescriptionLengthErrorMessage = "Описанието трябва да бъде с дължина от 10 до 500 символа";

public const string InvalidDescriptionErrorMessage = "Описанието може да съдържа букви (латинеца и китилица), цифри и \_+-.,!@%&\*();/\"' ";

public const string NameLengthErrorMessage = "Името трябва да бъде с дължина от 1 до 20 символа";

public const string InvalidNameErrorMessage = "Името трябва да съдържа букви (латинеца и китилица), цифри и -";

**To manage shopping cart access, you will assign users a unique ID using a globally unique identifier (GUID) when the user accesses the shopping cart for the first time. You'll store this ID using the ASP.NET Session state.**