**PortalEDU Seguimiento de implementación**

Crear nuevo proyecto Biblioteca Core

**- MOdels**

**- ACCeso a Datos**

**- Utilidades**

**INstalar Ngets**

**- MVC.RAzor.RuntimeCompilation (para guardar cambios y compilarlos en tiempo real)**

**- Microsoft.ASPnetcore.MVC para proyectos ADO y MOdels**

**- ASPnetcore.MVC.NewtonSoftjson para el proyecto principal**

**- Microsoft.Entityframeworkcore para ADO**

**- Microsoft.Extensions.Identity.Stores para ADO y models**

**- Microsoft.Aspnetcore.Identity.EntityframeworkCore para ADO**

**- Microsoft.EntityframworkCore.sqlserver para ADO**

Modificaion de clases (Esto es para que se guarden los cambios en vivo en el HTML “RuntimeCompilation”)

- En starup --> services.AddControllersWithViews().AddRazorRuntimeCompilation();

**Crear Areas para permisos**

* **Clic derecho en áreas y crearlas MVC**

**- Cambiar en el startup el endpoints**

**"{area=Anonimo}/{controller=Home}/{action=Index}/{id?}");**

**- Eliminar sus modelos y data porque usaremos los creados en biblioteca aspnetcore Models y ADO**

**- Mover "la clase" controllers global dentro de la carpeta controller de usuario anonimo y eliminamos la carpeta donde estaba**

**- Copiar ViewImport y viewstart dentro de los views de las areas. cambiar la ruta @{**

**Layout = "~/Views/Shared/\_Layout.cshtml";**

**}**

**- Setear home controller con [Area("Anonimo")]**

**- LLevarse clase errorviewmodels del proyewcto general al proyecto models creado con la biblioteca luego borrar models general**

**- Eliminamos las class1.cs creadas con las bibliotecas**

**- Arrastrar carpeta data a ADO luego elimarla**

**- Cambiar el namespace de applications dbcontext para usar el ADO**

**- Eliminar las migraciones existentes**

**- Mover carpeta Migrations a ADO**

**Crear carpetas dentro de DATA**

**- Inicalizador**

**- Repository**

**- Arrastrar carpeta home del view general para la carpeta view del usuario anonimo**

**Luego debemos de crar referencias en el pryecto general del proyecto ADO**

**Video 29**

**- dependencias del proyecto principasWEB agregarle Acceso a datos ADO**

**- Dendro de las dependencias acceso a datos de ADO agregar modelos y utilidades**

**OJO asegurarse que todos los paquetes estén actualizados en todos los proyectos**

Configurtacion del frontEnd

**Video 30**

**bajar css de** [**https://bootswatch.com/**](https://bootswatch.com/) **copiar el full no el mini y pegarlo en el Bootstrap original**

**Modificar navbar**

**Para que tome los cambios fijarse que se este importando el Bootstrap adecuado en el head**

Modificar la nav:

<a class="navbar-brand" **asp-area**="Anonimo" **asp-controller**="Home" **asp-action**="Index">Navbar</a>

Crear menú con estas características:

<a class="nav-link" **asp-area**="Admin" **asp-controller**="Configuracion" **asp-action**="Index">Configuracion</a>

* Cortar el login partial del origina y pegarlo en el div penultimo

video 31

Instalacion de plugin para frontend Instalar después de Bootstrap del head

<!--Css todos-->

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.16/css/jquery.dataTables.min.css" />

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/jqueryui/1.12.1/jquery-ui.min.css" />

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/css/toastr.min.css" />

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/sweetalert/1.1.3/sweetalert.min.css" />

Instalar en la parte de plugins

<!--Js plugins-->

<script src="https://cdnjs.cloudflare.com/ajax/libs/jqueryui/1.12.1/jquery-ui.min.js"></script>

<script src="https://cdn.datatables.net/1.10.16/js/jquery.dataTables.min.js"></script>

<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/jquery.serializeJSON/2.9.0/jquery.serializejson.min.js"></script>

<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/js/toastr.min.js"></script>

<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/sweetalert/1.1.3/sweetalert.min.js"></script>

<script src="https://kit.fontawesome.com/7588f654f8.js" crossorigin="anonymous"></script>

Inicia Repository

En el proyecto ADO carpeta data - repository agregar una interface (PUBLIC) IRepository donde iran los metodos globales

public interface IRepository<T> where T : class

Importar system.linq.expression

CLASE GLOBAL IREPOSITORY

{

T Get(int id);

IEnumerable<T> GetAll(

Expression<Func<T, bool>> Filter = null,

Func<IQueryable<T>, IOrderedQueryable<T>> orderBy = null,

string includeProperties = null

);

T GetFirstOrDefault(

Expression<Func<T, bool>> Filter = null,

string includeProperties = null

);

void Add(T entity);

void Remove(int id);

void Remove(T entity);

}

}

DENTRO de data crear una clase PUBLICA llamada Repository que debe de heredar de IRerpository

- IMplementar los metodos de la interfaz

- Agregar variable de solo lectura y modelar la Clase Repository

using Microsoft.EntityFrameworkCore;

using PortalEDU.ADO.Data.Repository;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Linq.Expressions;

using System.Text;

namespace PortalEDU.ADO.Data

{

public class Repository<T> : IRepository<T> where T : class

{

protected readonly DbContext Context;

internal DbSet<T> dbSet;

public Repository(DbContext context)

{

Context = context;

this.dbSet = context.Set<T>();

}

public void Add(T entity)

{

dbSet.Add(entity);

}

public T Get(int id)

{

return dbSet.Find(id);

}

public IEnumerable<T> GetAll(Expression<Func<T, bool>> filter = null, Func<IQueryable<T>, IOrderedQueryable<T>> orderBy = null, string includeProperties = null)

{

IQueryable<T> query = dbSet;

if (filter != null)

{

query = query.Where(filter);

}

// include property separadas por coma

if (includeProperties != null)

{

foreach (var includeProperty in includeProperties.Split

(new char[] { ',' }, StringSplitOptions.RemoveEmptyEntries))

{

query = query.Include(includeProperty);

}

}

if (orderBy != null)

{

return orderBy(query).ToList();

}

return query.ToList();

}

public T GetFirstOrDefault(Expression<Func<T, bool>> filter = null, string includeProperties = null)

{

IQueryable<T> query = dbSet;

if (filter != null)

{

query = query.Where(filter);

}

// include property separadas por coma

if (includeProperties != null)

{

foreach (var includeProperty in includeProperties.Split

(new char[] { ',' }, StringSplitOptions.RemoveEmptyEntries))

{

query = query.Include(includeProperty);

}

}

return query.FirstOrDefault();

}

public void Remove(int id)

{

T entityToRemove = dbSet.Find(id);

Remove(entityToRemove);

}

public void Remove(T entity)

{

dbSet.Remove(entity);

}

}

}

EMpezamos agregar las interfaces de las entidades (Tablas)

Crear el modelo Categoria en el proyecto Models con la entidades o atributos de la tabla

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Text;

namespace PortalEDU.Models

{

public class Categoria

{

[Key]

public int Id { get; set; }

[Required(ErrorMessage ="Ingrese un nombre para la categoria")]

[Display (Name ="Nombre Categoria")]

public string Nombre { get; set; }

[Required]

[Display(Name = "Orden de Vizualizacion")]

public string Orden { get; set; }

}

}

Creamos una interfaz dentro de repositry

ICategoriaRepository le damos atributo PUBLICA heredamos de IRepository y le pasamos la clase Categoria

Luego Listamos sus metodos

using Microsoft.AspNetCore.Mvc.Rendering;

using PortalEDU.Models;

using System;

using System.Collections.Generic;

using System.Text;

namespace PortalEDU.ADO.Data.Repository

{

public interface ICategoriaRepository : IRepository<Categoria>

{

IEnumerable<SelectListItem> GetListaCategoria();

void Update(Categoria categoria);

}

}

OJO: en la clase Application DBContext debemos de mapear todas las entidades, Ejemplo:

public DbSet<Categoria> Categoria { set; get; } // Asi todas las tablas de la base de datos.

Creamos una clase en carpeta Data de ADO con el nombre de CategoriaRepository para Instanciar la Interfaz

using Microsoft.AspNetCore.Mvc.Rendering;

using PortalEDU.ADO.Data.Repository;

using PortalEDU.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace PortalEDU.ADO.Data

{

public class CategoriaRepository : Repository<Categoria>, ICategoriaRepository

{

private readonly ApplicationDbContext \_db;

public CategoriaRepository(ApplicationDbContext db):base (db)

{

\_db = db;

}

public IEnumerable<SelectListItem> GetListaCategoria()

{

return \_db.Categoria.Select(i => new SelectListItem()

{

Text= i.Nombre,

Value= i.Id.ToString()

});

}

public void Update(Categoria categoria)

{

var ObjDesdeDb = \_db.Categoria.FirstOrDefault(s => s.Id == categoria.Id);

ObjDesdeDb.Nombre = categoria.Nombre;

ObjDesdeDb.Orden = categoria.Orden;

\_db.SaveChanges();

}

}

}

ES EL MOMENTO DE CREAR EL CONTENEDOR DE TRABAJO aqui se crearan todos los repositorios de trabajo EJEMPLO:

ICategoriaRepository Categoria { get; } // Y asi crear para todos los repositorios de las entidades creadas

Asi que esta ira creciendo a medida que crece nuestro proyecto

En IRepository Crear la interfaz IContenedorTrabajo

using System;

using System.Collections.Generic;

using System.Text;

namespace PortalEDU.ADO.Data.Repository

{

interface IContenedorTrabajo : IDisposable

{

ICategoriaRepository Categoria { get; }

void Save();

}

}

LUEGO EN DATA CREAMOS UNA CLASE DEL TIPO ContenedorTrabnajo

Aqui se van a crear el acceso a la interfaz y debe de ir creciendo junto a las nuevas entidades, aqui solo se creo categoria:

using PortalEDU.ADO.Data.Repository;

using System;

using System.Collections.Generic;

using System.Text;

namespace PortalEDU.ADO.Data

{

public class ContenedorTrabajo : IContenedorTrabajo

{

private readonly ApplicationDbContext \_db;

public ContenedorTrabajo(ApplicationDbContext db)

{

\_db = db;

Categoria = new CategoriaRepository(\_db);

}

public ICategoriaRepository Categoria { get; private set; }

public void Dispose()

{

\_db.Dispose();

}

public void Save()

{

\_db.SaveChanges();

}

}

}

----------AQUI TERMINA MODELADO DE LA ESTRUCTURA IREPOSITORYPATERN, SOLO SE DEBE DE IR MODIFICANDO A MEDIDA QUE

VAMOS CREANDO NUEVAS ENTIDADES--------

Vamos a intentar realizar un sccaffold DBFirst

Scaffold-DbContext "Server=localhost;Database=PortalEDU5; user id= sa; password=Ef1c13nc1@;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models