

WebUI

**>>> Views**

\_Layout => Views + Shared + sag + Add New Item + web + Razor \_Layout

\_ViewStart => Views + sag + Add New ITem + web + Razor \_ViewStart

\_ViewImport => Views + sag +Add New Item + web + Razor \_VıewImport

Shared

**\_Layout**

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>@ViewBag.Title</title>

</head>

<body>

<div>

@RenderBody()

</div>

</body>

</html

**\_ViewStart**

@{

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

}

**\_VıewImport**

@using MovieApp.Models

@addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers

-wwwroot

css + js + img

-startup

**public IConfiguration Configuration { get; }**

**public Startup(IConfiguration configuration)**

**{**

**Configuration = configuration;**

**}**

public void ConfigureServices(IServiceCollection services)

{

**services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);**

}

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment()) //Uyg. Geliştirme aşamasında hataları kullanıcıya gösterme.

{

app.UseDeveloperExceptionPage();

}

app.UseStatusCodePages(); //404 hatası sayfaları gostermek ıcın

app.UseStaticFiles(); // /css/style.css

app.UseMvcWithDefaultRoute(); // aşağıdakine eşittir.kendi url yapısınıa göre olusturur

/\*

app.UseMvc(routes => {

routes.MapRoute(

name: "default",

template: "{controller=Home}/{action=Index}/{id?}"

);

});

\*/

}

**-Bootstrap**

C:\Users\hakan\Desktop\CoreProject\BlogApp\BlogApp.WebUI ==>> cmd.

**npm i bootstrap //bootstrap@4.3.1 or bootstrap@4.0.0-alpha.6**

**npm init --yes //package.json dosyası içinde de yazılıp build edince paketin yüklenmesi**

**npm install**

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

//node\_modules ismini kullanmıcaz. yoksa show AllFiles + node\_modules + sag + ınclude

// /modules/bootstrap/dist/css/bootstrap.min.css

app.UseStaticFiles(new StaticFileOptions

{

FileProvider = new PhysicalFileProvider(Path.Combine

(Directory.GetCurrentDirectory(), "node\_modules")),

RequestPath = "/modules"

});

...

}

.Entity

Solution + Add + New P + Class Library (.Net Core) + P Name : .Entity =>Veri Tabanındaki tablolara karşılık gelen, sınıflar oluşturulur.

|  |
| --- |
| using Microsoft.AspNetCore.Mvc.ModelBinding;  using System;  namespace BlogApp.Entity  {  public class Blog  {  public int BlogId { get; set; }  public string Title { get; set; }  public string Description { get; set; }  public string Body { get; set; }  public string Image { get; set; }  public DateTime Date { get; set; }  public bool isApproved { get; set; }  //1 Blog 1 Kategoriye sahiptir.  public int CategoryId { get; set; }  public Category Category { get; set; }  }  }  ////////////  using System.Collections.Generic;  namespace BlogApp.Entity  {  public class Category  {  public int CategoryId { get; set; }  public string Name { get; set; }  //Her 1 Kategorinin \* den fazla blog olabilir.  public List<Blog> Blogs { get; set; }  }  } |

.Data

Solution + Add + New P + Class Library (.Net Core) + PName : .Data

-Abstract

Interface'ler olusturulur . : I.. burda metod tanımı(CRUD veya başka)

-Concrete

-EntityFramework

Context sınıfı olusturulur.

Sonra Ef.. adında sınıf olusuturulur, I dan implement edilir. Metod ne yaptıgını açıklarız.

Edit ile

<ItemGroup>

<PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="2.1.1" />

</ItemGroup>

Eklenir yada . Manage nuget => Microsoft.EntityFrameworkCore.SqlServer (2.1) yada Console

Install package Microsoft.EntityFrameworkCore.SqlServer.

Bunların hepsi **DbContext** kullanmak için bu kutuphane şart.

*Referenca ile .Entity ekle (WebUI içinde Entity ekle)*

Abstract – Interface. => Interfaceler var

Concret – EfCore => Interfaceden kalıtım alır metodları ve tanımlar bulunur.

Contcrete-EfCore + **BlogContext.cs**

|  |
| --- |
| using BlogApp.Entity;  using Microsoft.EntityFrameworkCore;  namespace BlogApp.Data.Concrete.EfCore  {  //Entity proje referans edilmeli.(.WebUI da .Entity ve.Data ekle)  public class BlogContext:DbContext  {  //Bu bloga gelen ConntectionString ile VT oluşacak  public BlogContext(DbContextOptions<BlogContext> options)  :base(options)  {  }  public DbSet<Blog> Blogs { get; set; }  public DbSet<Category> Categories { get; set; }  }  } |

**ConnectionString için** :

.WebUI + Add + New Item + AppSettingFile (An appsetting.json file used to configure Web Applications)

|  |
| --- |
| {  "ConnectionStrings": {  "DefaultConnection": "Server=(localdb)\\MSSQLLocalDB;Database=**BlogAppDb1**;Trusted\_Connection=True;MultipleActiveResultSets=true"  }  } |

.WebUI + **startup**.cs de

|  |
| --- |
| public void ConfigureServices(IServiceCollection services)  {  **services.AddDbContext<BlogContext>(options => options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"))); //ConnectionString ile BlogContext ilişkili oldu**.  } |

??Repository tanıtmak içinde startupda ayarı var.

public void ConfigureServices(IServiceCollection services)

{ services.AddTransient<I".."Repository, Ef".."Repository>();

}

Repository sonra yapılacak.

**Code First İle Migration**

DbContext => Database Migrations ile olur..

DbSet => Table

DbSet<Product> => Product(Table)

<<< Buradaki amaç DbContexte uygun bir Veritabanı oluşturmak. >>>

1.Kütüphane Kurma .

.WebUI+ sag + edit .csproj

|  |
| --- |
| <DotNetCliToolReference Include="Microsoft.EntityFrameworkCore.Tools.Dotnet" Version="2.0.0"></DotNetCliToolReference>  </ItemGroup> |

yada manage nuget ile Microsoft.EntityFrameworkCore.Tools.Dotnet

<<< bu paketi uygulamaya indirir.

2.Projede migration aktif etme.

|  |
| --- |
| public void ConfigureServices(IServiceCollection services)  {  **services.AddDbContext<BlogContext>(options => options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"), b => b.MigrationsAssembly("BlogApp.WebUI"))); //ConnectionString ile BlogContext ilişkili oldu**.  } |

C:\Users\hakan\Desktop\CoreProject\EFCore\EFCore yerine cmd yazılır ve komut sistemi açılır.

**dotnet ef migrations add InitialCreate** <<< yazılır ve Migration eklenir projeye. aşahıdakiler DbSet ve Model deki lere göre otomatik oluşur.

**dotnet ef database update** : komut ile VT Localde oluşmus olur .

//SeedData ile yapıyorduk burda komut ile yaptık.

3. Modelin tablosuna otomatik veri ekleme test verileri için static SeedData sınıfında Seed metodu olusturma

|  |
| --- |
| using BlogApp.Entity;  using Microsoft.AspNetCore.Builder;  using Microsoft.EntityFrameworkCore;  using Microsoft.Extensions.DependencyInjection;  using System;  using System.Linq;  namespace BlogApp.Data.Concrete.EfCore  {  public static class SeedData //Test sınıfı.  {  public static void Seed (IApplicationBuilder app)  {  BlogContext context = app.ApplicationServices.GetRequiredService<BlogContext>();  **context.Database.Migrate();** //VT nın oluştuğu an.  if (!context.Categories.Any()) //Herhangi Bir kayıt yoksa.  {  context.Categories.AddRange(  new Category() { Name="Category 1"},  new Category() { Name = "Category 2" },  new Category() { Name = "Category 3" }  );  context.SaveChanges();  }  if (!context.Blogs.Any())  {  context.Blogs.AddRange(  new Blog() { Title = "T1", Description = "D1", Body = "B1", Image = "r1.jpg", Date = DateTime.Now.AddDays(-5), isApproved=true, CategoryId=1 },  new Blog() { Title = "T2", Description = "D2", Body = "B2", Image = "r2.jpg", Date = DateTime.Now.AddDays(-6), isApproved = true, CategoryId = 1 },  new Blog() { Title = "T3", Description = "D3", Body = "B3", Image = "r3.jpg", Date = DateTime.Now.AddDays(-7), isApproved = false, CategoryId = 2 },  new Blog() { Title = "T4", Description = "D4", Body = "B4", Image = "r4.jpg", Date = DateTime.Now.AddDays(-8), isApproved = true, CategoryId = 3 }  );  context.SaveChanges();  }  }  }  } |

Sonra Startup + Configurede SeedData ayarı yapılmalı.

|  |
| --- |
| public void Configure(IApplicationBuilder app, IHostingEnvironment env)  {  SeedData.Seed(app);  } |

Guncellenme komutu yapmadan SeedData migrate ile proje calısınca aynı görevi yapar.

Sonra!!!

<<< Program.cs de UseDefaultServiseProvider eklenir.

|  |
| --- |
| public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>  WebHost.CreateDefaultBuilder(args)  .UseStartup<Startup>()  **.UseDefaultServiceProvider(options => options.ValidateScopes = false);** |

**Şimdi Modele properyti ekleyenince yada değişiklik olunca**

>>>Sonra projenın bulundugu klasörün yoluna C:\Users\hakan\Desktop\CoreProject\EFCore\EFCore cmd ile açalım

Yeni bir migration ekleyelim

**dotnet ef migrations add Yeni**

Sonra eklenmedi Proje çalısınca context.Database.Migrate() ile Vt eklenir yada cmd ile **dotnet ef database update** eğer Migration içinde önceki sınıflara gitmek ıstersek **dotnet ef database update "sınıfAdı"**

Örn : dotnet ef database update initial

Sonuc : DbContext’e uygun Migration ile Veri Tabanı oluşturduk.

Repository

Sorgular bu sınıflarda tanımlanır.

Abstract – Interface

|  |
| --- |
| using BlogApp.Entity;  using System.Linq;  namespace BlogApp.Data.Abstract  {  public interface ICategoryRepository  {  Category GetById(int categoryId); // Id ile getirme. Detaylandırma  IQueryable<Category> GetAll(); // Tümünü getir. Listeleme  void AddCategory(Category entity); //Model ile ekleeme. Ekleme  void UpdateCategory(Category entity); //Model ile güncelleme. Güncelleme  void DeleteCategory(int categoryId); //Id ile seçip silme. Silme  }  }  using BlogApp.Entity;  using System.Linq;  namespace BlogApp.Data.Abstract  {  public interface IBlogRepository  {  Blog GetById(int blogId); // Id ile getirme. Detaylandırma  IQueryable<Blog> GetAll(); // Tümünü getir. Listeleme  void AddBlog(Blog entity); //Model ile ekleeme. Ekleme  void UpdateBlog(Blog entity); //Model ile güncelleme. Güncelleme  void DeleteBlog(int blogId); //Id ile seçip silme. Silme  }  } |

Concrete – EfCore

Interfaceden kalıtım yoluyla implement edilir, metodların tanımı ne yaptıkları tanıtılır.

|  |
| --- |
| using BlogApp.Data.Abstract;  using BlogApp.Entity;  using Microsoft.EntityFrameworkCore;  using System.Linq;  namespace BlogApp.Data.Concrete.EfCore  {  public class EfCategoryRepository : ICategoryRepository  {  **private BlogContext context;**  **public EfCategoryRepository(BlogContext \_context)**  **{**  **context = \_context;**  **}**  public void **AddCategory**(Category entity)  {  context.Categories.Add(entity);  context.SaveChanges();  }  public void **DeleteCategory**(int categoryId)  {  var category = context.Categories.FirstOrDefault(c => c.CategoryId == categoryId);  if(category != null)  {  context.Categories.Remove(category);  context.SaveChanges();  }  }  public IQueryable<Category> **GetAll**()  {  return context.Categories;  }  public Category **GetById**(int categoryId)  {  return context.Categories.FirstOrDefault(c => c.CategoryId == categoryId);  }  public void **UpdateCategory**(Category entity)  {  context.Entry(entity).State = EntityState.Modified; //State => Modified,Added,Deleted,...  context.SaveChanges();  }  } // Modified : Entity üzerinde değişiklik olduğunu vurgular.  } |

|  |
| --- |
| using BlogApp.Data.Abstract;  using BlogApp.Entity;  using System;  using System.Linq;  namespace BlogApp.Data.Concrete.EfCore  {  public class EfBlogRepository : IBlogRepository  {  **private BlogContext context;**  **public EfBlogRepository(BlogContext \_context)**  **{**  **context = \_context;**  **}**  public void **AddBlog**(Blog entity)  {  context.Blogs.Add(entity);  context.SaveChanges();  }  public void **DeleteBlog**(int blogId)  {  var Blog = context.Blogs.FirstOrDefault(c => c.BlogId == blogId);  if (Blog != null)  {  context.Blogs.Remove(Blog);  context.SaveChanges();  }  }  public IQueryable<Blog> **GetAll**()  {  return context.Blogs;  }  **public Blog GetById(int BlogId)**  {  return context.Blogs.FirstOrDefault(c => c.BlogId == BlogId);  }  public void **UpdateBlog**(Blog entity)  {  /\*  Eski UpdateBlog ??? Burda tarıhı guncelliyor istemesemde.  context.Entry(entity).State = EntityState.Modified; //State => Modified,Added,Deleted,...  context.SaveChanges();  Neden sıkıntılı çünkü? Edit formunda alanı olmayan Date var.  Güncelle butonu basınca default bir deger atara ama biz güncellemedik sıkıntı onun için yukardaki yöntem  \*/  }  }  } |

HomeController

|  |
| --- |
| namespace BlogApp.WebUI.Controllers  {  public class HomeController : Controller  {  //DEPENDENCY INJECTION  **private IBlogRepository blogRepository;**  **public HomeController(IBlogRepository \_blogRepository)**  **{**  **blogRepository = \_blogRepository;**  **}**  public IActionResult Index()  {  return View(blogRepository.GetAll()); //TUM BLOG LİSTESİ GELİR.  } |

Startup.cs

|  |
| --- |
| public void ConfigureServices(IServiceCollection services)  {  **services.AddTransient<IBlogRepository, EfBlogRepository>();**  **services.AddTransient<ICategoryRepository, EfCategoryRepository>();**  services.AddDbContext<BlogContext>(options => options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"), b => b.MigrationsAssembly("BlogApp.WebUI"))); //ConnectionString ile BlogContext ilişkili oldu.  services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);  } |

Home/Index.cshtml

|  |
| --- |
| @model IEnumerable<Blog>  @foreach (var blog in Model)  {  <div class="card mb-2">  <h5 class="card-header">@blog.Title</h5>  <img class="card-img-top" src="~/img/@blog.Image" alt="@blog.Title">  <div class="card-body">  <h5 class="card-title">@blog.Title</h5>  <p class="card-text">@blog.Description</p>  <**a** **asp-controller**="Blog" **asp-action**="Details" **asp-route-id**="@blog.BlogId" class="btn btn-primary">Blog Details</**a**>  </div>  </div>  } |

Çalışma mantığı:

**C da I dependency uygulanır**

**Ef : I kalıtım alır.**

**Ef de EFDbContext dependency uygulanır.**

Program.cs de UseDefaultServiseProvider eklemeyi unutma.

|  |
| --- |
| public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>  WebHost.CreateDefaultBuilder(args)  .UseStartup<Startup>()  **.UseDefaultServiceProvider(options => options.ValidateScopes = false);** |

SONUC : Repository kullanarak verileri Home/Index de listeledik..

**<<<LISTING DATA>>>**

1.**Blog Controller** oluştur.

Sonra Controller içinde ;

IBlogRepository blogRepository; //null

public BlogController(IBlogRepository \_blogRepository)

{

blogRepository = \_blogRepository;

}

oluşturulur.

Yapıcı metodu : IblogRepository deki metodalara erişilir. Yukardaki sıralama ile çalışır.

2.**Blog Controller** içinde Listeleme işlemi için List metodu oluşturulur ve aynı zamanda List.cshtml view oluşturulur.

public IActionResult List()

{

return View(blogRepository.GetAll());//GetAll() EfBlogRepository’e bak.

}

List metodu : View içindeki komut, EF:I deki metoda erişir. Ef de Context(VT) ile ilişkilidir, verileri listeyen bir metoddur.

!!! Gelen veriler üzerinde burada filtereleme işlemi yapılabilir.

**List.cshtml**

İlk önce gelen sınıfı foreach dongusune sokmak için ;

@model IEnumerable<Blog>//ViewImport.cshtml uzantısı tanımlı, sadece Blog yazdık.

Sonra verileri ;

@foreach (var blog in Model)

{

@blog.BlogId

<img src="~/img/@blog.Image" width="100"/>

@blog.Title</td>

@blog.Date</td>

@blog.isApproved

}

İle verileri listeler.

Ayrıca **Tag Helper** kullanarak bu Viewde Details ve Delete butonları için;

<**a** class="btn btn-primary btn-sm" **asp-controller**="Blog" **asp-action**="Details" **asp-route-id**="@blog.BlogId">

Yapılabilir Delete de aynı sadece class değişik yani görseli. Ayrıntılı View BlogApp.WebUI/Blog/List

Blog Listeleme işlemi yaptık **Category List işlemi aynısıdır.** Detaylar için BlogAppApp.WebUI da CategoryController ve Category List View bak.

?Ayrıntıya bakarken

!!!favicon awsome kullanıldı Ben Intenetten link layout’a ekledim. Yada şu yapımalı WebUI da nuget paket ındırmede favıcon awsome paketi indirilir. (node\_modules altında olur.) Sonra bunun link layouta tanıtılır.

**<<<CREATING DATA>>>**

İlk Önce Blog tablosuna ekleyelim.

Şimdi Formdan gelen verileri tabloya eklicez ancak ben date propety formdan almıcam ayrıca formda isApprved da tanımlamıcam default false geliyor zaten. Bunun için,

[BindNever] //Form üzerinde gönderilmemesi için.

public DateTime Date { get; set; }

public bool isApproved { get; set; } //Default false.

Sonra Blog Controllera gelince

Formda Category Name bir selectList’en seçiceğim için;

//C -> I => EF:I -> Context

IBlogRepository blogRepository;

ICategoryRepository categoryRepository; //BlogControllerda kullanıcagım ıcın

public BlogController(IBlogRepository \_blogRepository,ICategoryRepository \_categoryRepository)

{

blogRepository = \_blogRepository;

categoryRepository = \_categoryRepository;

}

. . .

Sonra Create Metodunda

[HttpGet]

public IActionResult Create() //CREATE FORM OLUSTURMA, VIEWBAG İLE VERİ GONDERIMI VIEW

{

ViewBag.Categories = new SelectList(categoryRepository.GetAll(), "CategoryId", "Name");

return View();

}

[HttpPost]

public IActionResult Create(Blog entity)//FORMDAN GELEN VERİ İLE VT'A CREATE ISLEMI.

{

entity.Date = DateTime.Now; //BINDNEVER OLD. BIZ TANIMLADIK, if(ModelState.IsValid) //MODEL DEKİ VALIDATİONLAR SAGLANMIŞ İSE

{

blogRepository.AddBlog(entity);

return RedirectToAction("List");BU METOD LİST METODUNA GIDER DAHA DA BURAYA GELMEZ.

}

return View(entity);

}

* SelectList(ListeGonder,TutulcakVeri,GörünecekVeri)

**Blog Create.cshtml** de ise

**@model Blog //MODEL GONDERIMI**

@{

ViewData["Title"] = "Create";

}

<h2>Create</h2>

<div class="container">

<h2>Create</h2>

<br />

**//HTML HELPER KULLANILARAK YAPILDI**

<**form** asp-controller="Blog" asp-action="Create" **method="post"> //METOD POST**

<div class="form-group">

<**label** **asp-for**="Title"></**label**>

<**input** **asp-for**="Title" class="form-control">

</div>

<div class="form-group">

<**label** **asp-for**="Description"></**label**>

<**textarea** **asp-for**="Description" class="form-control"></**textarea**>

</div>

<div class="form-group">

<**label** **asp-for**="Body"></**label**>

<**textarea** **asp-for**="Body" class="form-control"></**textarea**>

</div>

<div class="form-group">

<**label** **asp-for**="Image"></**label**>

<**input** **asp-for**="Image" class="form-control">

</div>

**//SELECTLİST KULLANIMI**

<div class="form-group">

<**label** **asp-for**="CategoryId"></**label**>

<**select** class="form-control" **asp-for="CategoryId"** **asp-items="ViewBag.Categories**" >

<**option** disabled selected **value**="-1">Select Category</**option**>

</**select**>

</div>

<button type="submit" class="btn btn-primary my-1">Create</button>

</**form**>

</div>

**Category Ekleme:** Aynı Blog ekleme gibi sadece kodları yazıcam.

[HttpGet]

public IActionResult Create()

{

return View();

}

[HttpPost]

public IActionResult Create(Category entity)

{

if (ModelState.IsValid)

{

categoryRepository.AddCategory(entity);

return RedirectToAction("List");

}

return View(entity);

}

**@model Category //ONEMLI**

<**form** **asp-controller**="Category" **asp-action**="Create" method="post"> **//ONEMLI**

<div class="form-group">

<**label** **asp-for**="Name"></**label**>

<**input** **asp-for**="Name" class="form-control">

</div>

<button type="submit" class="btn btn-primary my-1">Create</button>

</**form**>

</div>

**<<<UPDATING DATA>>>**

Controllerda

//O ID LI DEGERININ FORM OLUSUTURUP VERİLERİ İLE BIRLIKTE FORMDA GOSTERME

[HttpGet]

public IActionResult Edit(int id)

{

ViewBag.Categories = new SelectList(categoryRepository.GetAll(), "CategoryId", "Name");

return View(blogRepository.GetById(id));

}

[HttpPost]

public IActionResult Edit(Blog entity)

{

if (ModelState.IsValid)

{

blogRepository.UpdateBlog(entity);

//VIEW'A MESAJ YOLLAMA,VERILER GUNCELENDIKTEN SONRA

TempData["msj"] = $"{entity.Title} güncellendi!!!";

//GELİRSE ACTION METOD LİST METODUNU ÇALISTIRIR.

return RedirectToAction("List");

}

//HATA OLUNCAFORMA TEKRAR VERİLERİN GONDERIMI

ViewBag.Categories = new SelectList(categoryRepository.GetAll(), "CategoryId", "Name");

return View(entity);

}

View’da önemli nokta **Id!=0** degerını almamız gerek formdan ama gözükmesine gerek yok.

@model Blog

<**form** **asp-controller**="Blog" **asp-action**="Edit" method="post">

<!--GUNCELLERKEN ID DEGERININDE EKLENMESI GEREK, GOZUKMEINE GEREK YOK-->

<**input** **type**="hidden" **asp-for**="BlogId" />. . .

Gerisi Create View dakinin aynısıdır.

TempData msj \_Layout da yazdım çunku

<div class="container">

<!--BURDA YAZMAKDAKI AMAC TEKRAR TEKRAR DIGER VIEWLARDA KULLANMAK ZORDUNDA KALMAMAM-->

@if (TempData["msj"] != null)

{

<div class="aler alert-success">

@TempData["msj"]

</div>

}

</div>

Birde EfBlogRepositoryde degısıklık oldu.Ayrıntı BlogApp + EfBlogRepository’e bak

public void UpdateBlog(Blog entity)

{

var sonuc = GetById(entity.BlogId);

if(sonuc != null)

{

sonuc.Title = entity.Title;

sonuc.Description = entity.Description;

sonuc.CategoryId = entity.CategoryId;

sonuc.Image = entity.Image;

context.SaveChanges();

}

}

**<<<ADDORUPDATING DATA >>>**

İşlemlerini bir yapabilirdik tek farkı **id degeri 0 eşitse Create yaptırmak !=0 ise yani sıfırdan farklı ise güncelleme işlemi** yaptırmasını sağlarsak yapmış oluruz.

**Blog dan başalıyalım** ilk önce AddOrUpdate içim IblogRepository’e

//AddOrUpdate için

void SaveBlog(Blog entity);

Sonra EfBlogRepository de

public void SaveBlog(Blog entity)

{

if(entity.BlogId == 0) //Kayıtta tarıh eklemek ıstıyorum

{

entity.Date = DateTime.Now;

context.Blogs.Add(entity);

}

else //Güncellemde tarıh guncellemek ıstemıyorum suan.

{

var sonuc = GetById(entity.BlogId);

if (sonuc != null)

{

sonuc.Title = entity.Title;

sonuc.Description = entity.Description;

sonuc.CategoryId = entity.CategoryId;

sonuc.Image = entity.Image;

}

}

context.SaveChanges();

}

Sonra Controller da

[HttpGet]

public IActionResult AddOrUpdate(int? id) //id nullable cunku kayıtta id gondermıcem ama guncellerken id göndercem.

{

ViewBag.Categories = new SelectList(categoryRepository.GetAll(), "CategoryId", "Name");

if (id == null)//Yeni kayıt

{

return View(new Blog());//BOŞ NESNE GONDERILDIGINDE AddOrUpdate View'da id ye 0 yollarız.

}

else //Güncelleme

{

return View(blogRepository.GetById((int)id));

}

}

HttpGet de form olustururuz. Create için forma veri gönderilmez ilke CategoryName hariç Güncellemde id değerine sagip verileri forma gonderiyoruz.

[HttpPost]

public IActionResult AddOrUpdate(Blog entity)

{

if (ModelState.IsValid)

{

blogRepository.SaveBlog(entity);

TempData["msj"] = $"{entity.Title} kayıt edildi!!!"; //VIEW'A MESAJ YOLLAMA,VERILER GUNCELENDIKTEN SONRA

return RedirectToAction("List");

}

ViewBag.Categories = new SelectList(categoryRepository.GetAll(), "CategoryId", "Name");

return View(entity);

}

Burda ise formdan gelen verileri vt kayıt edilir. View’ı ise;

@model Blog

@{

ViewData["Title"] = "AddOrUpdate";

}

<h2>Blog </h2>

<div class="container">

<br />

<**form** **asp-controller**="Blog" **asp-action**="AddOrUpdate" method="post">

<!--GUNCELLERKEN ID DEGERININDE EKLENMESI GEREK, GOZUKMEINE GEREK YOK-->

<**input** **type**="hidden" **asp-for**="BlogId" />

<div class="form-group">

<**label** **asp-for**="Title"></**label**>

<**input** **asp-for**="Title" class="form-control">

</div>

<div class="form-group">

<**label** **asp-for**="Description"></**label**>

<**textarea** **asp-for**="Description" class="form-control"></**textarea**>

</div>

<div class="form-group">

<**label** **asp-for**="Body"></**label**>

<**textarea** **asp-for**="Body" class="form-control"></**textarea**>

</div>

<div class="form-group">

<**label** **asp-for**="Image"></**label**>

<**input** **asp-for**="Image" class="form-control">

</div>

<div class="form-group">

<**label** **asp-for**="CategoryId"></**label**>

<**select** class="form-control" **asp-for**="CategoryId" **asp-items**="ViewBag.Categories">

<**option** disabled selected **value**="-1">Select Category</**option**>

</**select**>

</div>

<button type="submit" class="btn btn-primary my-1">Edit</button>

</**form**>

</div>

Artık Edit Butona(id ile gonderılır) basınca edit algılar Create butona basınca Create algılar.

**Category de ise;**

IcategoryRepository

//AddOrUpdate için

void SaveCategory(Category entity);

EfCategoryRepository

public void SaveCategory(Category entity)

{

if (entity.CategoryId == 0)

{

context.Categories.Add(entity);

}

else

{

var sonuc = GetById(entity.CategoryId);

if (sonuc != null)

{

sonuc.Name = entity.Name;

}

}

context.SaveChanges();

}

Controller

[HttpGet]

public IActionResult AddOrUpdate(int? id)

{

if (id == null)

{

return View(new Category()); //onemli

}

else

{

return View(categoryRepository.GetById((int)id));

}

}

[HttpPost]

public IActionResult AddOrUpdate(Category entity)

{

if (ModelState.IsValid)

{

categoryRepository.SaveCategory(entity);

TempData["msj"] = $"{entity.Name} kaydedildi";

return RedirectToAction("List");

}

//problem varsa, modeli tekrar aynı sayfaya yollar.

return View(entity);

}

Viewda ise

@model Category

@{

ViewData["Title"] = "AddOrUpdate";

}

<h2>AddOrUpdate</h2>

<br />

<div class="container">

<**form** **asp-controller**="Category" **asp-action**="AddOrUpdate" method="post">

<**input** **type**="hidden" **asp-for**="CategoryId" /> @\*ONEMLI\*@

<div class="form-group">

<**label** **asp-for**="Name"></**label**>

<**input** **asp-for**="Name" class="form-control">

</div>

<button type="submit" class="btn btn-primary my-1">Create</button>

</**form**>

</div>

**<<<DELETING DATA>>>**

Daha önce delete için repositoryde tanımlamıstık şimdi **Blogcontrollerda**

[HttpGet]

public IActionResult Delete(int id)

{

return View(blogRepository.GetById(id));

}

[HttpPost,ActionName("Delete")] //önemli aynı isimli parametrede aynı olunca hata vermemesi için

public IActionResult DeleteConfirmed(int id)

{

blogRepository.DeleteBlog(id);

TempData["msj"] = $"{id} id'li kayıt silindi!!!"; //VIEW'A MESAJ YOLLAMA,VERILER GUNCELENDIKTEN SONRA

return RedirectToAction("List");

}

Sonra **Delete Viewda**

@model Blog

@{

ViewData["Title"] = "Delete";

}

<h2>

<b>@Model.Title</b> isimli kaydı silmek istiyormusunuz?

</h2>

<**form** **asp-controller**="Blog" **asp-action**="Delete" method="post">

<**input** **type**="hidden" **asp-for**="BlogId" /> @\*ONEMLI Id tutmak\*@

<button type="submit" class="btn btn-primary">Delete</button>

</**form**>

Category içinde benzeri işlemler.

**<<< CATEGORY COMPONENT>>>(PartialView)**

Mantık örn : Anasayfadaki Menu yü(başka da olabilir) başka sayfalarda da kullanmak istersem ve dinamik biçimde verileri listelemek iştersem Component kullanıcam (Partial).

Kısacası birden fazla kullanılan alanı component yada partial yapmak avantajlı kod kalabalıgından kurtarır.(Shared de olur)

**.WebUI** projede **ViewComponents** klasör oluşturuyoruz. Sonra içine **CategoryViewComponent.cs** eklenir. Bu aynı Controller gibi davranır Ne demek bu örn . HomeContoroller ise View da /Home/Index gibi yani Viewda **Shared** klasöründe **Components** klasör oluşturulur bunun içinde de **CategoryMenu** klasör olusturup ve içinde **Default.cshtml** (Partial View) olustururuz.

(1 Category Menu)

**CategoryMenuViewComponent.cs**

using BlogApp.Data.Abstract;

using Microsoft.AspNetCore.Mvc;

namespace BlogApp.WebUI.ViewComponents

{

public class CategoryMenuViewComponent : ViewComponent

{

private ICategoryRepository categoryRepository;

public CategoryMenuViewComponent(ICategoryRepository \_categoryRepository)

{

categoryRepository = \_categoryRepository;

}

public IViewComponentResult Invoke()

{

return View(categoryRepository.GetAll()); //Defualt.cshtml gider başka html'e yonlendırebiliriz de.

}

}

}

**Default.cshtml (Partial View)**

@model IEnumerable<Category>

<div class="list-group">

@foreach (var ktg in Model)

{

<a href="#" class="list-group-item">@ktg.Name</a>

}

</div>

**Bunu Diger View’da çağırma Index.cshmtl**

<div class="row">

<div class="col-md-3">

@await Component.InvokeAsync("CategoryMenu")

</div>

Ne yapmış olduk şimdi bir tanem menu olusturduk bunu diğer sayfalardan çağırdık ve Menu deki verileri dinamik hale getirdik.

**<<<(BLOG LIST Partial View)>>>**

Burda **Partial View** kullanıcaz **Neler yapıcaz** öncelikle Home/Index sayfasında sadece isHome True ve is Approved true olanları göstermek isticez sonra Blog/Index de is Approved true olanları listelenicek tabi bunları PartialViewden alarak yapıcaz.

Home/Shared**/\_BlogList.cshtml** (partial view)

@model IEnumerable<Blog>

@foreach (var blog in Model)

{

<div class="card mb-2">

<h5 class="card-header">@blog.Title</h5>

<img class="card-img-top" src="~/img/@blog.Image" alt="Card image cap">

<div class="card-body">

<h5 class="card-title">@blog.Title</h5>

<p class="card-text">@blog.Description</p>

<**a** **asp-controller**="Blog" **asp-action**="Details" asp-route-id="@blog.BlogId" class="btn btn-primary">Blog Details</**a**>

</div>

</div>

}

Bunu Home/Index ve Blog/Indexde çağırıcaz. İlk önce **Home/Index** de

@model IEnumerable<Blog>

<div class="col-md-9">

@Html.Partial("\_BlogList", Model)

</div>

Çağırılır. Koşul vardı Homeda sadece iki alan true ise bunu ise şöyle yapıcaz.

Önceden **Blog** da i sApproved alanı vardı ama is Home yoktu bunu eklicez.

//Hangileri Anasayfada gösterilsin ?

public bool isHome { get; set; }

VT bunu algılaması için

C:\Users\hakan\Desktop\BlogApp\BlogApp.WebUI>**dotnet ef migrations add AddColumnBlog1**

İşlemi yapılır ve **SeedData** daki(program çalısınca çalısır) otomatik güncelleme ile vt kayıt edilir.(komut ile kaydetmeside vardı)

Şimdi **Blog/List** de

<th>is Home</th>

. . .

<td>

@if (blog.isHome)

{

<i class="fa fa-check" aria-hidden="true"></i>

}

else

{

<i class="fa fa-times" aria-hidden="true"></i>

}

</td>

Aynı için is Approved içinde yapılır. Tabloda alanları yapılır.

Sonra **AddOrUpdate** sayfasında iki taben checkbox eklenir.

<div class="form-group">

<label class="form-check-label">

<**input** **asp-for**="isApproved" class="form-check-input" />

is Approved

</label>

</div>

Aynısı is Home içinde geçerli

Sonra **EfBlogRepository** de **AddOrUpdate** için kullanılan metodda is home alanı eklenir.

sonuc.isApproved = entity.isApproved;

Sonra **HomeController** da listeleme ve filtreleme yapılır

public IActionResult Index()

{

return View(blogRepository.GetAll().Where(i=>i.isApproved && i.isHome));

}

Şimdi ile **Blog/List** için

@model IEnumerable<Blog>

<div class="col-md-9">

@Html.Partial("\_BlogList", Model)

</div>

**BlogControllerda**

public IActionResult Index()

{

return View(blogRepository.GetAll()

.Where(i=>i.isApproved)

.OrderByDescending(i=>i.Date)

);

}

**<<<BLOG DETAILS >>>**

**Ne yapıldı.** Home/Index sayfasında details butona basılınca idile birlikt o blogun ayrıntılı bir sayfada göstermek bununi için;

**BlogController**

public IActionResult Details(int id)

{

return View(blogRepository.GetById(id));

}

GetById Repository sınıfında ne iş yaptığı var. Bu id ile eşleşen kaydın verilerini tutar.

**Blog/Details**

@model Blog

**…**

<div class="col-md-9">

<div class="card mb-2">

<div class="card-header">

@Model.Title

</div>

<img class="card-img-top" src="~/img/@Model.Image" alt="@Model.Title">

<div class="card-body">

<h5 class="card-title">@Model.Title</h5>

<p class="card-text">@Model.Description</p>

<p class="card-text">@Model.Body</p>

</div>

</div>

</div>

Mantık bu şekilde.

**<<<Kullanıcı Layout Admin Layout>>>**

**Ne Yapıldı.** Şimdi Kullanıcınn gördüğü sayfa ile adminin gördüğü sayfa farklı olmalıdır. Örneğin Create,Delete işlemlerini admin yapar ve admin görmeli sadece ona özel bit layout olmalı bunun için. Bir tane daha \_AdminLayout.cshtml adında tanımladık

Peki \_Layout mu \_AdminLayout mu Diger Viewsler nasıl anlıcak. Şöyle zaten \_ViewStartdan \_Layout tanımlamıstık. \_AdminLayout için o Views’a git ve

@{

ViewData["Title"] = "AddOrUpdate";

Layout = "~/Views/Shared/\_AdminLayout.cshtml";

}

Gibi Layout un adresini ver.

**<<<BLOG FILTERS >>>**

Bu zamana kadar category menu ve bloglistler yapıldı.

**Ne yapıldı.** Seçilen kategoriye göre bloglar listelenecek ve seçilen kategori görsel olarak gösterilecek.

**BlogControler.cs**

public IActionResult Index(int? id)

{ //gelen CategoryId

var sorgu = blogRepository.GetAll()

.Where(i => i.isApproved);

if (id != null)

{

sorgu = sorgu.Where(i => i.CategoryId == id);

}

return View(sorgu.OrderByDescending(i => i.Date));

}

Neden id nullable cunku. Category id işaret etmekdedir. Id degeri gelmese bile bloglar listelenmeli eger id gelirse ona göre listelenmelidir.

Menu yu componentle CategoryMenu ile tasarlamıştık şimdi seçilen kategorinin göresel olarak hem seçildiğini görmek içinde

**CategoryMenu.cs**

public IViewComponentResult Invoke()

{

ViewBag.SelectedCategory = RouteData?.Values["id"];//null olursa Values degerine gelmez.Hata olmaması ııcn.

return View(categoryRepository.GetAll()); //Defualt.cshtml gider başka html'e yonlendırebiliriz de.

}

Ardından

**Default.cshtml**

@foreach (var ktg in Model)

{

<**a** **asp-controller**="Blog"

**asp-action**="Index"

**asp-route-id**="@ktg.CategoryId"

class="list-group-item @(ViewBag.SelectedCategory==ktg.CategoryId.ToString()?"active":"")">@ktg.Name</**a**>

}

Son olarakda **BlogLits** de Category ile eşleşen kayıt yoksa mevcut kayıt yoksa,

@if(Model.Count() == 0)

{

<div class="aler alart-warning">Kayıt yok!!!</div>

}

Else ise blogların listelenmesi yer alır.

**<<<SEARCH FORM>>>**

Ne yapıldı. Bir tane formda Arama kutucuğu olusuturuldu girilien stringe göre blogdaki propertlere göre arama yapıp listeleyebiliriz önce

Formu bir partial view olarak tanımladım. **\_SearchBox.cshtml**

<**form** **asp-controller**="Blog" **asp-action**="Index" method="get" class="form-inline my-2 my-lg-0">

<input id="q" name="q" class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">

<button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>

</**form**>

Ve \_Layout ta @Html.Partial("\_SearchBox") ile partial view çağırdım.

Sonra **BlogController** da

public IActionResult Index(int? id, string q)

{ //gelen CategoryId

var sorgu = blogRepository.GetAll()

.Where(i => i.isApproved);

if (id != null)

{

sorgu = sorgu.Where(i => i.CategoryId == id);

}

if (!string.IsNullOrEmpty(q))

{

//sorgu = sorgu.Where(i => i.Title.Contains(q) || i.Description.Contains(q) || i.Body.Contains(q));

sorgu = sorgu.Where(i => EF.Functions.Like(i.Title, "%" + q + "%")|| EF.Functions.Like(i.Description, "%" + q + "%")|| EF.Functions.Like(i.Body, "%" + q + "%"));

}

return View(sorgu.OrderByDescending(i => i.Date));

}

İle aranmış olur.

**<<<BLOG SLIDER>>>**

* Bootstrap carousel ile slider form.
* Sliderı Partial View tanımlıcam modelide içinde bulunduğu sayfadan alıcak.

Eski de yaptığımız CategoryMenu (ViewComponent) : Bu Menuyu hangi sayfaya eklersek ekleyelim. Gidip Kendi verisini modelini VT bağlanıp alıyor ve içeriği kendi gösteriyor.

* Slider sadece Anasayfada çalışacağı için, ViewComponent Kullanmaya gerek yok.
* <<<Dolayısıyla Slider Partial View Home/Index Viewda tanımlanacak Modelide HomeController + Index Action metodundan verileri alıcak ve Slider da ise Tanımlanan
* Ayrınca Index e gelen model ile Slidere gelen model aslında aynı olmamalı.
* !!! ViewComponent de kullanılabilir ancak tekrar tekrar kullanılacak olan işlemlerde component kullanmak mantıklıdır.

Yapılacaklar :

Shared altından bir \_Slider.cshtml adından partial View oluştur.

|  |
| --- |
| @model IEnumerable<Blog>  <div id="carouselExampleIndicators" class="carousel slide mb-3" data-ride="carousel">  <ol class="carousel-indicators">  @for (int i = 0; i < Model.Count(); i++) //Index Bilgileri Tutar.  {  if (i == 0) //0 Olunca Active class kullansın. Yani ilk eleman.  {  <li data-target="#carouselExampleIndicators" data-slide-to="@i" class="active"></li>  }  else  {  <li data-target="#carouselExampleIndicators" data-slide-to="@i"></li>  }  }  </ol>  <div class="carousel-inner" role="listbox">  @{var index = 0;}  @foreach (var item in Model) //Yukardaki gibi ilk Eleman Active edicek şekilde.  {  if (index == 0)  {  <div class="carousel-item active">  <img class="d-block img-fluid" src="~/img/@item.Image" alt="@item.Title">  <div class="carousel-caption d-none d-md-block">  <h3>@item.Title</h3>  <p>@item.Description</p>  </div>  </div>  index++; //Sırayla Active olmayan divler de aktif olur.  }  else  {  <div class="carousel-item ">  <img class="d-block img-fluid" src="~/img/@item.Image" alt="@item.Title">  <div class="carousel-caption d-none d-md-block">  <h3>@item.Title</h3>  <p>@item.Description</p>  </div>  </div>  }  }  </div>  <a class="carousel-control-prev" href="#carouselExampleIndicators" role="button" data-slide="prev">  <span class="carousel-control-prev-icon" aria-hidden="true"></span>  <span class="sr-only">Previous</span>  </a>  <a class="carousel-control-next" href="#carouselExampleIndicators" role="button" data-slide="next">  <span class="carousel-control-next-icon" aria-hidden="true"></span>  <span class="sr-only">Next</span>  </a>  </div> |

Home/Index

|  |
| --- |
| @model IEnumerable<Blog>  @{  ViewData["Title"] = "Index";  }  <div class="container my-4">  <div class="row">  <div class="col-md-3">  @await Component.InvokeAsync("CategoryMenu")  </div>  <div class="col-md-9">  @Html.Partial("\_Slider",Model) //=>>\_Slider.cshtml burdan çağırılır.  @Html.Partial("\_BlogList", Model)  </div>  </div>  </div> |

Alttaki resim

Sorun ? => Sliderda gösterilen model ile Home/Indexde gösterilen model aynı olmamalı bunun için Blog Model isSlider diye bir özellik eklemeliyim true ise Sliderda gösterilsin false ise Sliderda gösterilmesin gibi özellik eklenecek.

|  |
| --- |
| //Slider da gösterilsin mi?  public bool isSlider { get; set; } //==>> Defult false. |

Bunu VT kayıt için migration işlemleri yapılmalıdır.

C:\Users\hakan\Desktop\BlogApp\BlogApp.WebUI>dotnet ef migrations add AddCloumnIsSlider ile yapılır. SeedData sınıfındaki Migrate komutu ile eger proje çalısınca otomatik veri tabanına kayıt eder. Yada elle girilecek komut sistemine güncellenmesi için.

Şimdi Yeni alan Ekleyince

Blog/List : Eklenen alanı göstermeliyim.

|  |
| --- |
| <th>is Slider</th>  . . .  <td>  @if (blog.isSlider)  {  <i class="fa fa-check" aria-hidden="true"></i>  }  else  {  <i class="fa fa-times" aria-hidden="true"></i>  }  </td> |

Sonra Blog/AddOrUpda : Ekelem ve güncelemem için burdada alanı eklemelıyım.

|  |
| --- |
| <div class="form-group">  <label class="form-check-label">  <**input** **asp-for**="isSlider" class="form-check-input" />  is Slider  </label> |

Güncellemenin ve Eklemenin Gerçekleşesi için EfBlogRepositoryde AddOrUpdate için olan metoda aşağıdaki eklenmeli.

|  |
| --- |
| sonuc.isSlider = entity.isSlider; |

Güzel Özellik .

Şimdi Home/Index’ e hem Blog List gönderilecek hemde Slider List gönderelecek ben bunun için WebUI modelinde bir sınıf olusuturup buna uygun bir model yazmalıyım.

Model/HomeBlog.cs

|  |
| --- |
| using BlogApp.Entity;  using System.Collections.Generic;  namespace BlogApp.WebUI.Models  {  public class HomeBlogModel  {  public List<Blog> SliderBlogs { get; set; }  public List<Blog> HomeBlogs { get; set; }  }} |

Sonra HomeConroller/Index

|  |
| --- |
| public IActionResult Index()  {  var model = new HomeBlogModel();  model.HomeBlogs = blogRepository.GetAll().Where(i => i.isApproved && i.isHome).ToList();  model.SliderBlogs = blogRepository.GetAll().Where(i => i.isApproved && i.isSlider).ToList();  return View(model);  } |

Ve Home/Index de olan değişiklikler ise aşağıdaki gibi \_Slider da bir değişiklik olmaz.

|  |
| --- |
| @model HomeBlogModel    . . .  @Html.Partial("\_Slider",Model.SliderBlogs)  @Html.Partial("\_BlogList", Model.HomeBlogs) |



**<<<UPLOAD BLOG IMAGE>>>**

Resim eklemeyi güncelleme sayfasında oluşturduk.

Blog/Edit.cshtml

|  |
| --- |
| <**form** **asp-controller**="Blog" **asp-action**="Edit" method="post" **enctype="multipart/form-data"** >  <!--GUNCELLERKEN ID DEGERININDE EKLENMESI GEREK, GOZUKMEINE GEREK YOK-->  <**input** **type**="hidden" **asp-for**="BlogId" />  <div class="form-group">  <img **src="~/img/@Model.Image"** width="120"/>  <input **type="file" name="file" id="file"** class="form-control">  <**input** **type="hidden" asp-for="Image**" /> //Form doldururken Image tutmalıyız.  </div>  <button type="submit" class="btn btn-primary my-1">Edit</button>  </**form**> |

BlogController.cs + Edit

|  |
| --- |
| //EDIT  [HttpGet]  public IActionResult Edit(int id)  {  ViewBag.Categories = new SelectList(categoryRepository.GetAll(), "CategoryId", "Name");  return View(blogRepository.GetById(id));  }  [HttpPost]  public async **Task<IActionResult>** Edit(Blog entity, **IFormFile file**) //IFormFile resim yüklemek için. Birden fazla yüklencek ise IENumerable<IFormFile> ve async Task olamsı gerek  {  if (ModelState.IsValid)  {  if(**file != null**)  {  **//Resim için**  var path = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot\\img", file.FileName);  using (var stream = new FileStream(path, FileMode.Create))  {  await file.CopyToAsync(stream); //Kayıt işlemi yapıldığı an.  }  entity.Image = file.FileName; //resim ismini göndermek.  }  **//Resim Bitti**  blogRepository.SaveBlog(entity);  TempData["msj"] = $"{entity.Title} kayıt edildi!!!"; //VIEW'A MESAJ YOLLAMA,VERILER GUNCELENDIKTEN SONRA  return RedirectToAction("List");  }  ViewBag.Categories = new SelectList(categoryRepository.GetAll(), "CategoryId", "Name");  return View(entity);  } |

EfBlogRespository de SaveBlog yapsına bak. Görsel Sonuc(3).



**<<<HTML EDITOR>>>**

* CkEditör Ecosystem Bootstrap kurudugum gibi kurduk

Blog/Edit de Body için CKEditor kullanıcaz

@section scripts{

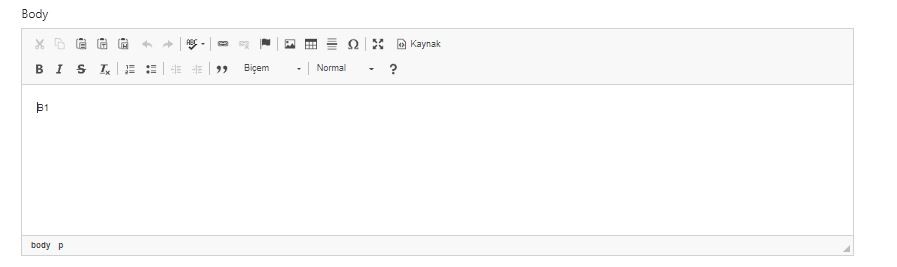
<script src="~/modules/ckeditor/ckeditor.js"></script>

<script>

CKEDITOR.replace("Body");

</script>

}



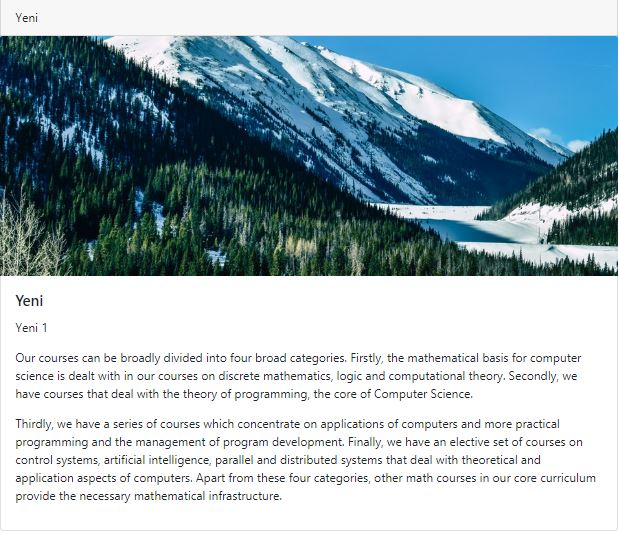
Sonra \_Layout.cshtml de

@RenderSection("scripts",false) <!--Her zaman kullanmak zorunda kalmayız. flase verirsek-->

Blog/Details de

Body ‘e yazılan verilerim html taglerin gözükmemsi için

<p class="card-text">@Html.Raw(Model.Body)</p> <!--Html etiketleri yorumlanmış gözükür.-->



using BlogApp.Data.Abstract;

using BlogApp.Entity;

using System;

using System.Linq;

namespace BlogApp.Data.Concrete.EfCore

{

public class EfBlogRepository : IBlogRepository

{

private BlogContext context;

public EfBlogRepository(BlogContext \_context)

{

context = \_context;

}

public void AddBlog(Blog entity)

{

context.Blogs.Add(entity);

context.SaveChanges();

}

public void DeleteBlog(int blogId)

{

var Blog = context.Blogs.FirstOrDefault(c => c.BlogId == blogId);

if (Blog != null)

{

context.Blogs.Remove(Blog);

context.SaveChanges();

}

}

public IQueryable<Blog> GetAll()

{

return context.Blogs;

}

public Blog GetById(int BlogId)

{

return context.Blogs.FirstOrDefault(c => c.BlogId == BlogId);

}

public void SaveBlog(Blog entity)

{

if(entity.BlogId == 0) //Kayıtta tarıh eklemek ıstıyorum

{

entity.Date = DateTime.Now;

context.Blogs.Add(entity);

}

else //Güncellemde tarıh guncellemek ıstemıyorum suan.

{

var sonuc = GetById(entity.BlogId);

if (sonuc != null)

{

sonuc.Title = entity.Title;

sonuc.Description = entity.Description;

sonuc.Body = entity.Body;

sonuc.CategoryId = entity.CategoryId;

sonuc.Image = entity.Image;

sonuc.isHome = entity.isHome;

sonuc.isApproved = entity.isApproved;

sonuc.isSlider = entity.isSlider;

}

}

context.SaveChanges();

}

public void UpdateBlog(Blog entity)

{

/\*

Eski UpdateBlog ??? Burda tarıhı guncelliyor istemesemde.

context.Entry(entity).State = EntityState.Modified; //State => Modified,Added,Deleted,...

context.SaveChanges();

Neden sıkıntılı çünkü? Edit formunda alanı olmayan Date var.

Güncelle butonu basınca default bir deger atara ama biz güncellemedik sıkıntı onun için yukardaki yöntem

\*/

var sonuc = GetById(entity.BlogId);

if (sonuc != null)

{

sonuc.Title = entity.Title;

sonuc.Description = entity.Description;

sonuc.CategoryId = entity.CategoryId;

sonuc.Image = entity.Image;

context.SaveChanges();

}

}

}

}

using BlogApp.Data.Abstract;

using BlogApp.Entity;

using Microsoft.EntityFrameworkCore;

using System.Linq;

namespace BlogApp.Data.Concrete.EfCore

{

public class EfCategoryRepository : ICategoryRepository

{

private BlogContext context;

public EfCategoryRepository(BlogContext \_context)

{

context = \_context;

}

public void AddCategory(Category entity)

{

context.Categories.Add(entity);

context.SaveChanges();

}

public void DeleteCategory(int categoryId)

{

var category = context.Categories.FirstOrDefault(c => c.CategoryId == categoryId);

if(category != null)

{

context.Categories.Remove(category);

context.SaveChanges();

}

}

public IQueryable<Category> GetAll()

{

return context.Categories;

}

public Category GetById(int categoryId)

{

return context.Categories.FirstOrDefault(c => c.CategoryId == categoryId);

}

public void SaveCategory(Category entity)

{

if (entity.CategoryId == 0)

{

context.Categories.Add(entity);

}

else

{

var sonuc = GetById(entity.CategoryId);

if (sonuc != null)

{

sonuc.Name = entity.Name;

}

}

context.SaveChanges();

}

public void UpdateCategory(Category entity)

{

context.Entry(entity).State = EntityState.Modified; //State => Modified,Added,Deleted,...

context.SaveChanges();

}

}

}