#1 Asp.Net Core 5.0 Proje Kampı Giriş

1. Dinamik Blog Projesi Yapılacaktır.
2. Discord Grubu oluşturulacaktır.
3. Admin – Kullanıcı – Yazar Panelleri Olacaktır.
4. Ödevler verilecektir.

#2 Asp.Net Core 5.0 Proje Kampı Başlamadan Önce Bilinmesi Gerekenler

1. Visual Studio 2019 Programı kullanılacaktır
2. Sql Server kurulumu yapılmış olmalıdır
3. EntityFramework – Katmanlı Mimari – Core yapısı kullanılacaktır
4. MVC ile ilgili bilginiz olmalıdır.

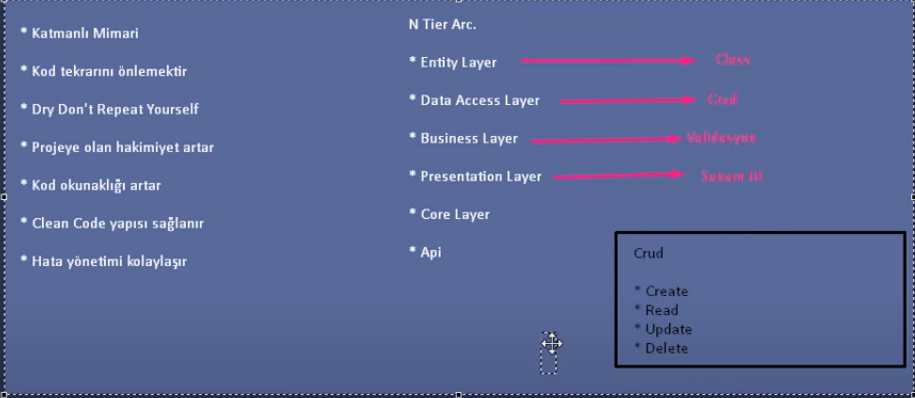
#3 Asp.Net Core 5.0 Proje Kampı Yeni Proje Oluşturma

1. Visual Studio 2019 açılır.
2. Asp.Net Core Web App ( Model – View – Controller ) kısmı seçilir
3. .Net Core 5.0 seçilir.
4. MVC kısımlarının anlatılması

#4 Asp.Net Core 5.0 Proje Kampı Controller View Ve Layout

1. HomeController -> IActionResult adında türünde Test isminde bir metot oluşturalım
2. Test e sağ tıklayıp Add View -> Razor View ekleyelim.
3. Test adında bir görünüm dosyamız oluştu.
4. Bu sayfada bir tablo yapalım ve test amaçlı görünümü çalıştıralım.

#5 Asp.Net Core 5.0 Proje Kampı Katmanlı Mimariye Bakış Ve N Tier Architecture



#6 Asp.Net Core 5.0 Proje Kampı Katmanların Oluşturulması

1. Solution Sağ tık 🡪 New Project 🡪 Class Library (.Net Core ) 🡪 EntityLayer(.Net 5.0)
2. Solution Sağ tık 🡪 New Project 🡪 Class Library (.Net Core ) 🡪 DataAccessLayer(.Net 5.0)
3. Solution Sağ tık 🡪 New Project 🡪 Class Library (.Net Core ) 🡪 BusinessLayer(.Net 5.0)
4. Solution Sağ tık 🡪 New Project 🡪 Class Library (.Net Core ) 🡪 CoreLayer(.Net 5.0)
5. Her katmandaki Class1.cs leri silelim

#7 Asp.Net Core 5.0 Proje Kampı Entitylerin Oluşturulması Category – Blog

1. EntityLayer(Sağ tık) 🡪 Add 🡪 New Folder 🡪 Concrete ismi verilir
2. Concrete(Sağ tık) 🡪 Add 🡪 Class 🡪 Category ismi verilir.

**Category Class ı:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace EntityLayer.Concrete

{

public class Category

{

public int CategoryID { get; set; }

public string CategoryName { get; set; }

public string CategoryDescription { get; set; }

public bool CategoryStatus { get; set; }

}

}

1. Concrete(Sağ tık) 🡪 Add 🡪 Class 🡪 Blog ismi verilir.

**Blog Class ı :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace EntityLayer.Concrete

{

public class Blog

{

public int BlogID { get; set; }

public string BlogTitle { get; set; }

public string BlogContent { get; set; }

public string BlogTrumbnailImage { get; set; }

public string BlogImage { get; set; }

public DateTime BlogCreateDate { get; set; }

public bool BlogStatus { get; set; }

}

}

#8 Asp.Net Core 5.0 Proje Kampı Entitylerin Oluşturulması Writer - Comment - About - Contact

1. Concrete(Sağ tık) 🡪 Add 🡪 Class 🡪 Writer ismi verilir.

**Writer class ı:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace EntityLayer.Concrete

{

public class Writer

{

public int WriterID { get; set; }

public string WriterName { get; set; }

public string WriterAbout { get; set; }

public string WriterImage { get; set; }

public string WriterMail { get; set; }

public string WriterPassword { get; set; }

public bool WriterStatus { get; set; }

}

}

1. Concrete(Sağ tık) 🡪 Add 🡪 Class 🡪 Comment ismi verilir.

**Comment Class ı :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace EntityLayer.Concrete

{

public class Comment

{

public int CommentID { get; set; }

public string CommentUserName { get; set; }

public string CommentTitle { get; set; }

public string CommentContent { get; set; }

public DateTime CommentDate { get; set; }

public bool CommentStatus { get; set; }

}

}

1. Concrete(Sağ tık) 🡪 Add 🡪 Class 🡪 Contact ismi verilir.

**Contact Class ı :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace EntityLayer.Concrete

{

public class Contact

{

public int ContactID { get; set; }

public string ContactUserName { get; set; }

public string ContactMail { get; set; }

public string ContactSubject { get; set; }

public string ContactMessage { get; set; }

public DateTime ContactDate { get; set; }

public bool ContactStatus { get; set; }

}

}

1. Concrete(Sağ tık) 🡪 Add 🡪 Class 🡪 About ismi verilir.

**About Class ı :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace EntityLayer.Concrete

{

public class About

{

public int AboutID { get; set; }

public string AboutDetails1 { get; set; }

public string AboutDetails2 { get; set; }

public string AboutImage1 { get; set; }

public string AboutImage2 { get; set; }

public string AboutMapLocation { get; set; }

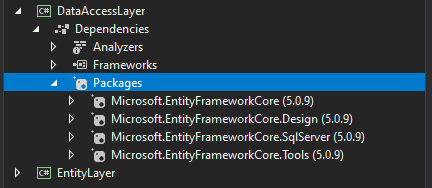
public bool AboutStatus { get; set; }

}

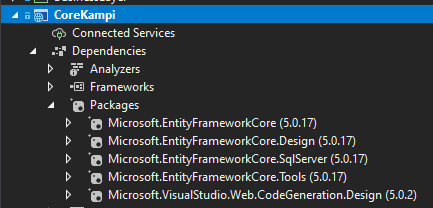
}

#9 Asp.Net Core 5.0 Proje Kampı Entity Framework Core Paketleri

1. Her sınıfa [Key] kısmını id leri üzerine ekleyelim
2. DataAccessLayer (Sağ Tık ) 🡪 Add 🡪 Manage Nuget Packages 🡪Browse aşağıdaki paketler ilgili sürüme göre yüklenir.



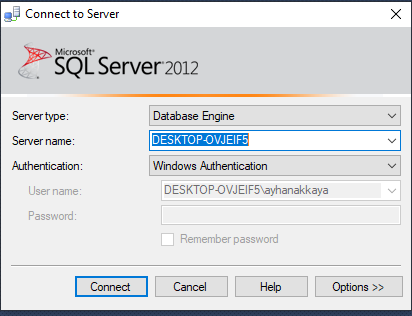
1. Yukarıdaki paketler CoreKampi katmanına da kurulmalı



1. DataAccessLayer ( Sağ tık ) 🡪 Add 🡪 New Folder 🡪Concrete adında klasör oluştur.
2. DataAccessLayer ( Sağ tık ) 🡪 Add 🡪 New Folder 🡪Abstract adında klasör oluştur.
3. DataAccessLayer ( Sağ tık ) 🡪 Add 🡪 New Folder 🡪Repositories adında klasör oluştur.

#10 Asp.Net Core 5.0 Proje Kampı Connection String Ve Context Sınıfı

1. DataAccessLayer 🡪 Concrete ( Sağ tık ) 🡪 Add 🡪Class 🡪Context.cs oluşturalım. Sql Server Management Studio programını açıp server ismini kopyalayalım.



**Context class ı :**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

namespace DataAccessLayer.Concrete

{

public class Context : DbContext

{

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer("server=DESKTOP-OVJEIF5; database=CoreBlogDb;integrated security=true;");

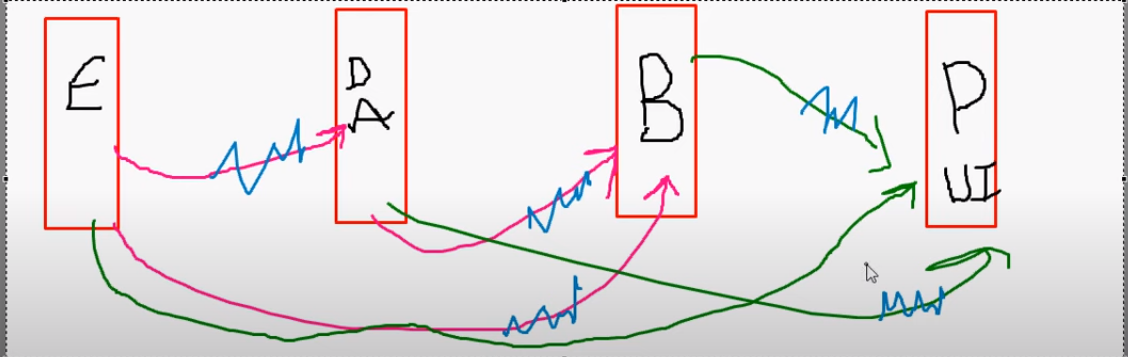
}

}

}

database=CoreBlogDb oluşturacağımız veritabanı adıdır

#11 Asp.Net Core 5.0 Proje Kampı Referansların Tanımlanması Ve Context Dbset Propertyleri



1. DataAccessLayer🡪Dependencies (Sağ tık)🡪 Add Project Reference 🡪 çıkan pencerede EntityLayer seçilip ok tuşuna basılır
2. BusinessLayer🡪Dependencies (Sağ tık)🡪 Add Project Reference 🡪 çıkan pencerede EntityLayer ve DataAccessLayer seçilip ok tuşuna basılır
3. CoreKampi🡪Dependencies (Sağ tık)🡪 Add Project Reference 🡪 çıkan pencerede EntityLayer , DataAccessLayer ve BusinessLayer seçilip ok tuşuna basılır
4. Context.cs de Bağlantıları tanımlayalım.

**Context.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using EntityLayer.Concrete;

using Microsoft.EntityFrameworkCore;

namespace DataAccessLayer.Concrete

{

public class Context : DbContext

{

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer("server-DESKTOP- OVJEIF5;database=CoreBlogDb;integrated security=true;");

}

public DbSet<About> Abouts { get; set; }

public DbSet<Blog> Blogs { get; set; }

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

public DbSet<Comment> Comments { get; set; }

public DbSet<Contact> Contacts { get; set; }

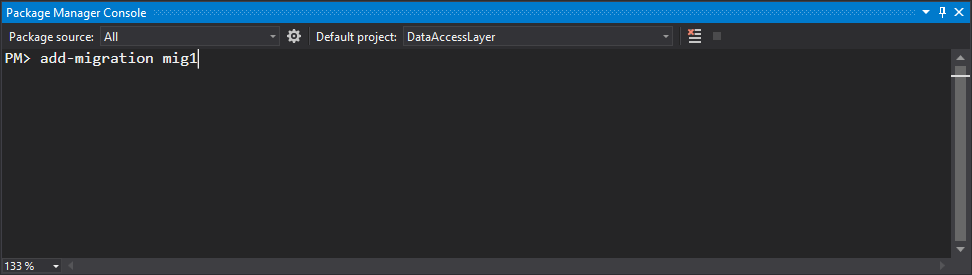
public DbSet<Writer> Writers { get; set; }

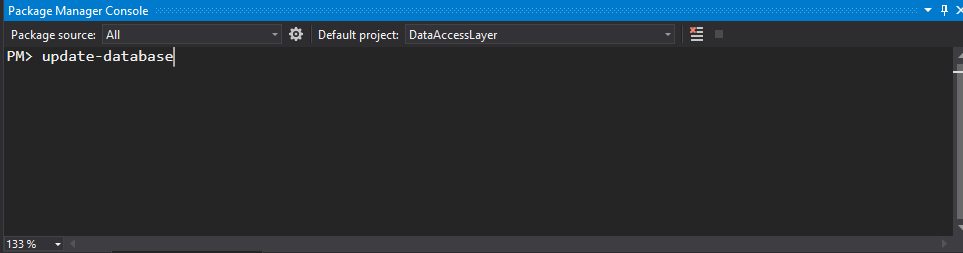
}

}

#12 Asp.Net Core 5.0 Proje Kampı Migration İşlemleri

1. View -> Other Windows -> Package Manager Console seçilir.
2. Gelen alanda Default Project kısmında DataAccessLayer seçilir ve aşağıdaki kodlar yazılır.





#13 Asp.Net Core 5.0 Proje Kampı 13 İlişkilerin Eklenmesi Add Migration İşlemleri

1. Ödev 1 : List ile ICollection arasındaki ilişki nedir?
2. **Category ile Blog** arasındaki ilişki ve **Comment ile Blog** arasındaki ilişki

**Category Class**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace EntityLayer.Concrete

{

public class Category

{

[Key]

public int CategoryID { get; set; }

public string CategoryName { get; set; }

public string CategoryDescription { get; set; }

public bool CategoryStatus { get; set; }

public List<Blog> Blogs { get; set; }

}

}

**Blog Class**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace EntityLayer.Concrete

{

public class Blog

{

[Key]

public int BlogID { get; set; }

public string BlogTitle { get; set; }

public string BlogContent { get; set; }

public string BlogTrumbnailImage { get; set; }

public string BlogImage { get; set; }

public DateTime BlogCreateDate { get; set; }

public bool BlogStatus { get; set; }

public int CategoryID { get; set; }

public Category Category { get; set; }

public List<Comment> Comments { get; set; }//Blog ile Comment ilişkisi

}

}

**Comment Class**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace EntityLayer.Concrete

{

public class Comment

{

[Key]

public int CommentID { get; set; }

public string CommentUserName { get; set; }

public string CommentTitle { get; set; }

public string CommentContent { get; set; }

public DateTime CommentDate { get; set; }

public bool CommentStatus { get; set; }

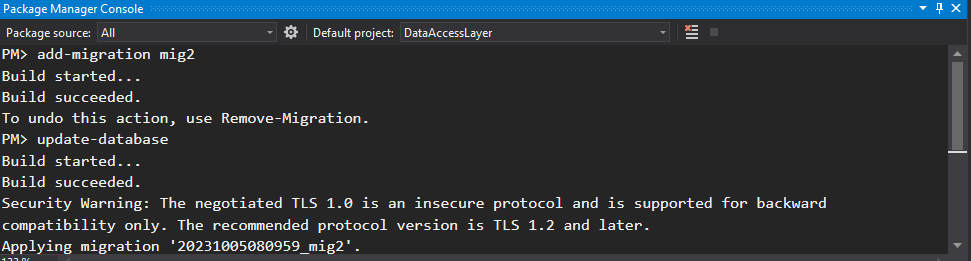
public int BlogID { get; set; }

public Blog Blog { get; set; }

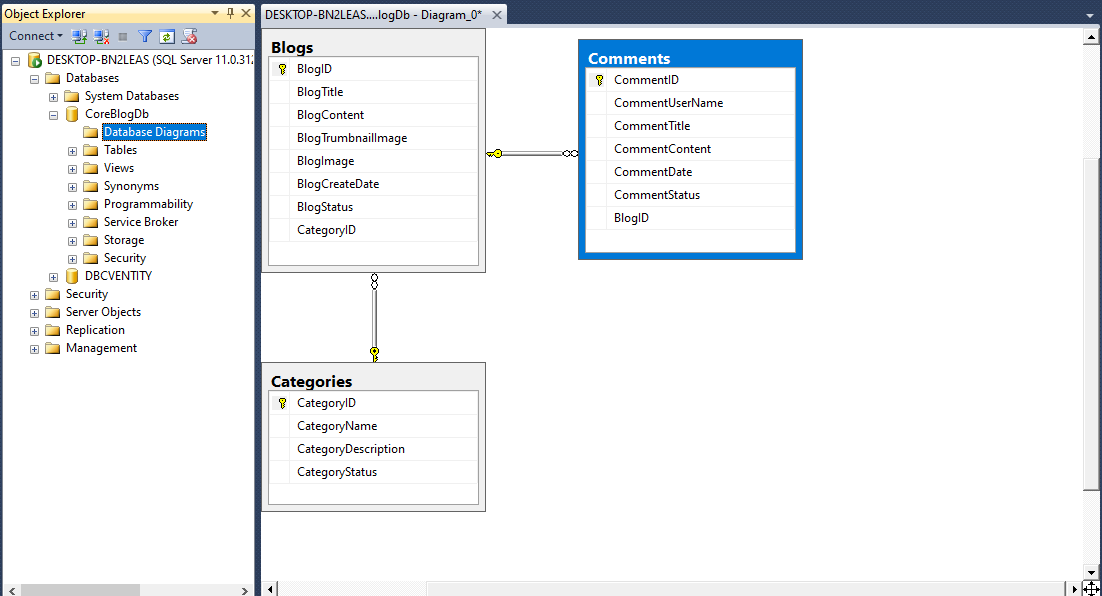
}

}

1. Package Manager Console a gelinip add-migration mig2 diyelim ve sonra update-database diyelim.

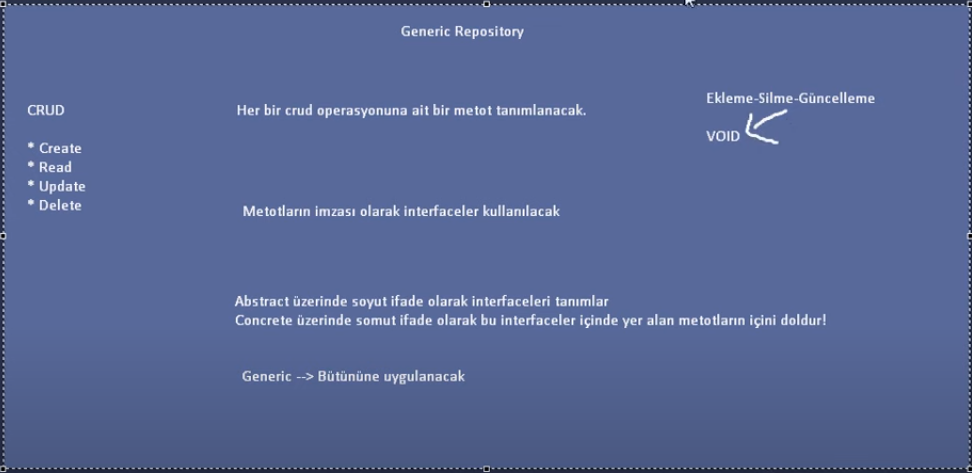


1. Veritabanında ilişki diyagramı aşağıdaki gibi olmuştur.



#14 Asp.Net Core 5.0 Proje Kampı Interfacelerin Tanımlanması

1. Generic Repository Yapısı



1. DataAccessLayer -> Abstract ( Klasör yoksa oluştur) sağ tık-> Add -> New Item -> Interface -> IcategoryDal ismi verilir.

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.Abstract

{

public interface IcategoryDal

{

List<Category> ListAllCategory(); //Tüm Categorileri getir

void AddCategory(Category category); // Ekleme

void DeleteCategory(Category category); // Silme

void UpdateCategory(Category category); //Güncelleme

Category GetById(int id);// id ye göre işlemler

}

}

1. DataAccessLayer -> Abstract sağ tık-> Add -> New Item -> Interface -> IBlogDal ismi verilir.

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.Abstract

{

public interface IBlogDal

{

List<Blog> ListAllCategory(); //Tüm Blogları getir

void Addblog(Blog blog); // Ekleme

void DeleteBlog(Blog blog); // Silme

void Updateblog(Blog blog); //Güncelleme

Blog GetById(int id);// id ye göre işlemler

}

}

// Üstteki Category ve Blog Interfacelerinin tanımlanma şekilleri yanlış olmasa da daha doğru bir yöntem var

#15 Asp.Net Core 5.0 Proje Kampı Repositorylerin Yazılması

1. DataAccessLayer -> Repositories ( Klasör yoksa oluştur) sağ tık-> Add -> New Item -> Class -> CategoryRepository ismi verilir.

using DataAccessLayer.Abstract;

using DataAccessLayer.Concrete;

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.Repositories

{

public class CategoryRepository : IcategoryDal //implement Interface

{

Context c = new Context();

public void AddCategory(Category category)

{

c.Add(category);

c.SaveChanges();

}

public void DeleteCategory(Category category)

{

c.Remove(category);

c.SaveChanges();

}

public Category GetById(int id)

{

return c.Categories.Find(id);

}

public List<Category> ListAllCategory()

{

return c.Categories.ToList();

}

public void UpdateCategory(Category category)

{

c.Update(category);

c.SaveChanges();

}

}

}

1. DataAccessLayer -> Repositories sağ tık-> Add -> New Item -> Class -> BlogRepository ismi verilir.

using DataAccessLayer.Abstract;

using DataAccessLayer.Concrete;

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.Repositories

{

public class BlogRepository : IBlogDal

{

public void Addblog(Blog blog)

{

using var c = new Context();

c.Add(blog);

c.SaveChanges();

}

public void DeleteBlog(Blog blog)

{

using var c = new Context();

c.Remove(blog);

c.SaveChanges();

}

public Blog GetById(int id)

{

using var c = new Context();

return c.Blogs.Find(id);

}

public List<Blog> ListAllCategory()

{

using var c = new Context();

return c.Blogs.ToList();

}

public void Updateblog(Blog blog)

{

using var c = new Context();

c.Update(blog);

c.SaveChanges();

}

}

}

#16 Asp.Net Core 5.0 Proje Kampı Generic Repository

1. DataAccessLayer -> Repositories sağ tık-> Add -> New Item -> Interface -> IGenericDal ismi verilir.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.Abstract

{

public interface IGenericDal<T> where T:class

{

void Insert(T t);

void Delete(T t);

void Update(T t);

List<T> GetListAll();

T GetById(int id);

}

}

1. DataAccessLayer-> Repositories Sağ Tık -> Add -> Class -> CommentRepository.cs ismi verilir

using DataAccessLayer.Abstract;

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.Repositories

{

public class CommentRepository : IGenericDal<Comment>

{

public void Delete(Comment t)

{

throw new NotImplementedException();

}

public Comment GetById(int id)

{

throw new NotImplementedException();

}

public List<Comment> GetListAll()

{

throw new NotImplementedException();

}

public void Insert(Comment t)

{

throw new NotImplementedException();

}

public void Update(Comment t)

{

throw new NotImplementedException();

}

}

}

1. DataAccessLayer-> Repositories Sağ Tık -> Add -> Class -> GenericRepository.cs ismi verilir

using DataAccessLayer.Abstract;

using DataAccessLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.Repositories

{

public class GenericRepository<T> : IGenericDal<T> where T : class

{

public void Delete(T t)

{

using var c = new Context();

c.Remove(t);

c.SaveChanges();

}

public T GetById(int id)

{

using var c = new Context();

return c.Set<T>().Find(id);

}

public List<T> GetListAll()

{

using var c = new Context();

return c.Set<T>().ToList();

}

public void Insert(T t)

{

using var c = new Context();

c.Add(t);

c.SaveChanges();

}

public void Update(T t)

{

using var c = new Context();

c.Update(t);

c.SaveChanges();

}

}

}

#17 Asp.Net Core 5.0 Proje Kampı Business Katmanı Service ve Managerların Yazılması

1. BusinessLayer sağ tık -> Add -> New Folder -> Abstract ismi verilir.
2. BusinessLayer sağ tık -> Add -> New Folder -> Concrete ismi verilir.
3. BusinessLayer sağ tık -> Add -> New Folder -> ValidationRules ismi verilir.
4. BusinessLayer -> Abstract sağ tık -> Add -> Interface -> ICategoryService ismi verilir.

**ICategoryService Class**

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BusinessLayer.Abstract

{

public interface ICategoryService

{

void CategoryAdd(Category category);

void CategoryDelete(Category category);

void CategoryUpdate(Category category);

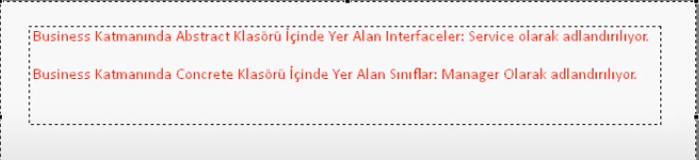
List<Category> GetList();

Category GetById(int id);

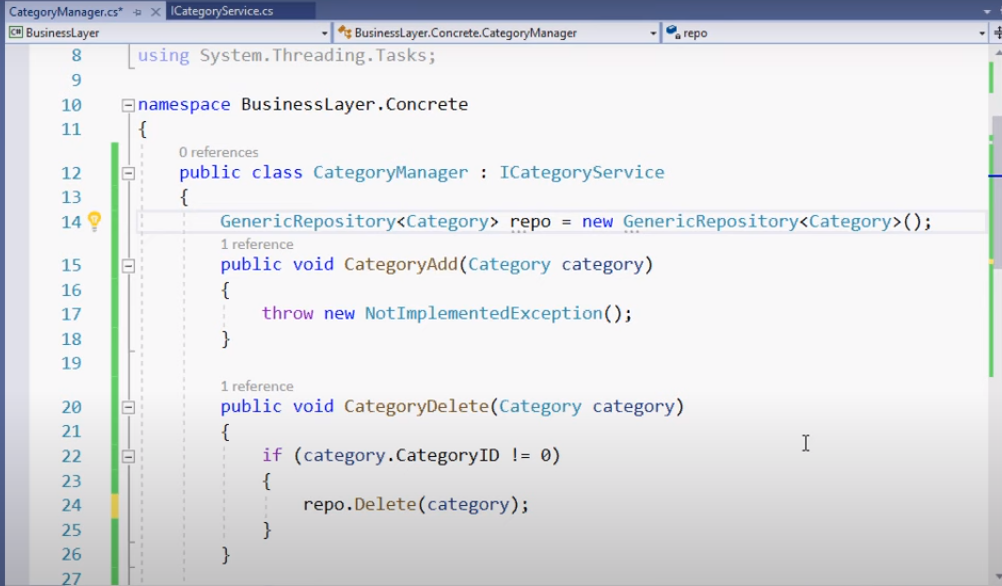
}

}

1. BusinessLayer -> Concrete sağ tık -> Add -> Class -> CategoryManager.cs ismi verilir.



#18 Asp.Net Core 5.0 Proje Kampı Business Katmanı Hatalı Validasyonlar



#19 Asp.Net Core 5.0 Proje Kampı Entity Framework Repository

1. Yine aynı katmanda bulunan Abstract klasöründeki IBolgDal ve ICategoryDal içerikleri aşağıdaki gibi değiştirilir.

**ICategoryDal Class ı**

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.Abstract

{

public interface ICategoryDal:IGenericDal<Category>

{

}

}

**IBlogDal Class**

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.Abstract

{

public interface IBlogDal:IGenericDal<Blog>

{

}

}

1. DataAccessLayer -> Abstract sağ tık -> Add -> New Item -> Interface -> IAboutDal.cs ismi verilir.

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.Abstract

{

public interface IAboutDal : IGenericDal<About>

{

}

}

1. DataAccessLayer -> Abstract sağ tık -> Add -> New Item -> Interface -> ICommentDal.cs ismi verilir.

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.Abstract

{

public interface ICommentDal : IGenericDal<Comment>

{

}

}

1. DataAccessLayer -> Abstract sağ tık -> Add -> New Item -> Interface -> IWriterDal.cs ismi verilir.

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.Abstract

{

public interface IWirterDal : IGenericDal<Writer>

{

}

}

1. DataAccessLayer -> Abstract sağ tık -> Add -> New Item -> Interface -> IContactDal.cs ismi verilir.

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.Abstract

{

public interface IContactDal : IGenericDal<Contact>

{

}

}

1. DataAccessLayer Sağ tık -> Add -> New Folder -> EntityFramework ismi verilir.
2. DataAccessLayer -> EntityFramework sağ tık -> Add -> Class -> EfCategoryRepository eklenir.

using DataAccessLayer.Abstract;

using DataAccessLayer.Repositories;

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.EntitiyFramework

{

public class EfCategoryRepository : GenericRepository<Category>, ICategoryDal

{

}

}

1. DataAccessLayer -> EntityFramework sağ tık -> Add -> Class -> EfBlogRepository eklenir.

using DataAccessLayer.Abstract;

using DataAccessLayer.Repositories;

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.EntitiyFramework

{

public class EfBlogRepository : GenericRepository<Blog>, IBlogDal

{

}

}

1. DataAccessLayer -> EntityFramework sağ tık -> Add -> Class -> EfWriterRepository eklenir.

using DataAccessLayer.Abstract;

using DataAccessLayer.Repositories;

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.EntitiyFramework

{

public class EfWriterRepository : GenericRepository<Writer>, IWirterDal

{

}

}

1. DataAccessLayer -> EntityFramework sağ tık -> Add -> Class -> EfAboutRepository eklenir.

using DataAccessLayer.Abstract;

using DataAccessLayer.Repositories;

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.EntitiyFramework

{

public class EfAboutRepository : GenericRepository<About>, IAboutDal

{

}

}

1. DataAccessLayer -> EntityFramework sağ tık -> Add -> Class -> EfCommentRepository eklenir.

using DataAccessLayer.Abstract;

using DataAccessLayer.Repositories;

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.EntityFramework

{

public class EfCommentRepository : GenericRepository<Comment>,ICommentDal

{

}

}

1. DataAccessLayer -> EntityFramework sağ tık -> Add -> Class -> EfContactRepository eklenir.

using DataAccessLayer.Abstract;

using DataAccessLayer.Repositories;

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.EntityFramework

{

public class EfContactRepository : GenericRepository<Contact> ,IContactDal

{

}

}

#20 Asp.Net Core 5.0 Proje Kampı Category Manager Sınıfında Revizeler

using BusinessLayer.Abstract;

using DataAccessLayer.EntitiyFramework;

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BusinessLayer.Concrete

{

public class CategoryManager : ICategoryService

{

EfCategoryRepository efCategoryRepository;

public CategoryManager()

{

efCategoryRepository = new EfCategoryRepository();

}

public void CategoryAdd(Category category)

{

efCategoryRepository.Insert(category);

}

public void CategoryDelete(Category category)

{

efCategoryRepository.Delete(category);

}

public void CategoryUpdate(Category category)

{

efCategoryRepository.Update(category);

}

public Category GetById(int id)

{

return efCategoryRepository.GetById(id);

}

public List<Category> GetList()

{

return efCategoryRepository.GetListAll();

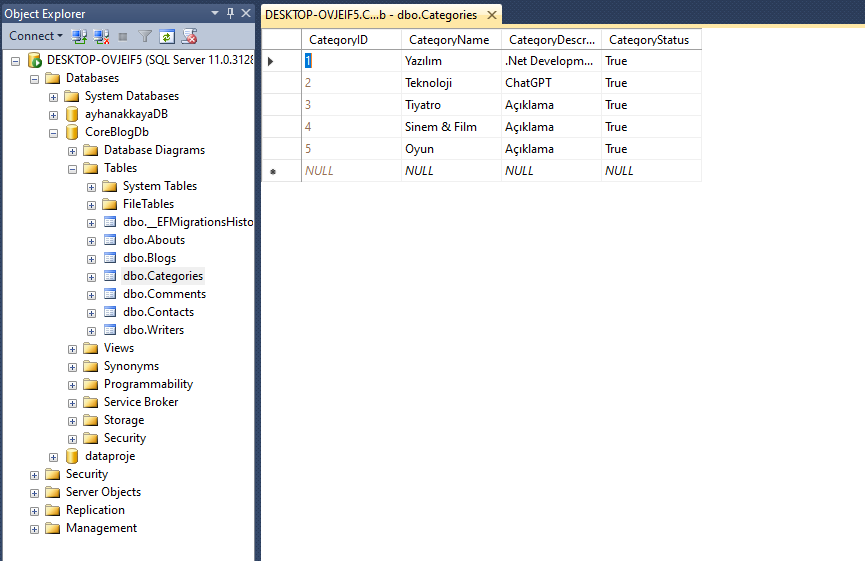
}

}

}

#21 Asp.Net Core 5.0 Proje Kampı Kategorilerin Listelenmesi

1. CoreKampi -> Controller sağ tık -> Add -> Controller -> MVC Empty -> Category ismi verilir.
2. Veritabanına gidip category tablosuna birkaç veri ekleyelim.



1. CategoryManager ı güncelleyelim

using BusinessLayer.Abstract;

using DataAccessLayer.Abstract;

using DataAccessLayer.EntitiyFramework;

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BusinessLayer.Concrete

{

public class CategoryManager : ICategoryService

{

ICategoryDal \_categoryDal;

public CategoryManager(ICategoryDal categoryDal)

{

\_categoryDal = categoryDal;

}

public void CategoryAdd(Category category)

{

\_categoryDal.Insert(category);

}

public void CategoryDelete(Category category)

{

\_categoryDal.Delete(category);

}

public void CategoryUpdate(Category category)

{

\_categoryDal.Update(category);

}

public Category GetById(int id)

{

return \_categoryDal.GetById(id);

}

public List<Category> GetList()

{

return \_categoryDal.GetListAll();

}

}

}

1. CoreKampi -> Controllers -> Category.cs içerisine aşağıdaki kodlar yazılır.

using BusinessLayer.Concrete;

using DataAccessLayer.EntitiyFramework;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace CoreKampi.Controllers

{

public class Category : Controller

{

CategoryManager cm = new CategoryManager(new EfCategoryRepository());

public IActionResult Index()

{

var values = cm.GetList();

return View(values);

}

}

}

1. Daha sonra yukarıdaki işaretli Index kısmına sağ tıklayıp Add View -> Razor View -> Add diyelim.

Category.cs e bağlı olarak oluşturulan Index.cshtml de kategorileri listeleyen kodları yazalım.

@using EntityLayer.Concrete

@model List<Category>

@{

ViewData["Title"] = "Index";

}

<h1>Index</h1>

<table class="table table-bordered">

<tr>

<th>ID</th>

<th>Kategori Adı</th>

<th>Durumu</th>

<th>Detaylar</th>

<th>Sil</th>

</tr>

@foreach (var item in Model)

{

<tr>

<td>@item.CategoryID.</td>

<td>@item.CategoryName.</td>

<td>@item.CategoryStatus.</td>

<td><a href="#" class="btn btn-warning">Detaylar</a></td>

<td><a href="#" class="btn btn-danger">Sil</a></td>

</tr>

}

</table>

#22 Asp.Net Core 5.0 Proje Kampı Temanın Projeye Eklenmesi

1. <https://github.com/MuratYucedag/CoreBlogTema> adresinden tema indilirip rar dan çıkarılır.
2. Rar dan çıkan klasörü CoreKampi katmanında wwwroot içerisine sürekleyelim.
3. CoreKampi -> Views -> Shared sağ tık -> Add Razor View -> UserLayout ismi verilir.
4. Controller sağ tık -> Add -> Controller -> MVC (Empty Controller) -> BlogController ismi verilir.
5. BlogControler içerisinde Index metoduna sağ tıklanıp Add Razor View denir. Ve Use Layout kısmı seçilir ve UserLayout a bağlanır.
6. İndirilen temda blog3.html deki tüm kodlar kopyalanıp UserLayout.cshtml ye yapıştırılır. Ayrıca hata vermemesi çin UserLayout un en altına @RenderBody() yazılır.Index.cshtml çalıştırılnca tema eklenmiş olur.
7. Css leri düzenlemek için css kısımlarının önüne tema klasör ismi eklenir.
8. Sayfada Blog Post olan section kodlarını şimdilik kesip bir metin belgesinde saklayalım ve aynı alana @RenderBody() ekleyelim. Ve sayfanın en altındaki @RenderBody() yi silelim.

#23 Asp.Net Core 5.0 Proje Kampı 23 Partial View Await Async

PartialView ile sayfanın bazı kısımlarını metotlar gibi farklı sayfalarda yönetebiliriz.

1. UserLayout.cshtml de aşağıdaki sarı işaretli kodu ilgili alana ekleyelim.

@{

Layout = null;

}

@await Html.PartialAsync("Partial1")

1. Shared sağ tık -> Add -> View -> Razor View -> Partial1 ismi verilir (Create Partial kısmı işaretlenir.)
2. UserLayout da 95.satırda başlayan navbar kısmını keselim Partial1.cshtml ye ekleyelim. Yukarıdai sarı kodu da üstten kesip kesilen yere yapıştıralım.

@await Html.PartialAsync("Partial1")

1. Artık navbar kısmını Partial1 üzerinde düzenlemeye başlayalım.
2. Yukarıdaki işlemler gibi sayfamızın üst tarafını ayrı bir partial a ekleyelim. ismi partial2 olsun
3. UserLayout da 50 ile 61 arasında olan Welcome Back kısmını Partial2 ye aktaralım.

Kesilen yere alttaki komutu ekleyelim.

@await Html.PartialAsync("Partial2")

1. Yukarıdaki işlemler gibi sayfamızın alt (footer) tarafını ayrı bir partial a ekleyelim. ismi partial3 olsun
2. UserLayout da footer alanını Partial3 ye aktaralım.

Kesilen yere alttaki komutu ekleyelim.

@await Html.PartialAsync("Partial3")

#24 Asp.Net Core 5.0 Proje Kampı Blogların Listelenmesi

1. BusinessLayer -> Abstract sağ tık -> Add -> Interface -> IBlogService.cs eklenir.

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BusinessLayer.Abstract

{

public interface IBlogService

{

void BlogAdd(Blog blog);

void BlogDelete(Blog blog);

void BlogUpdate(Blog blog);

List<Blog> GetList();

Blog GetById(int id);

}

}

1. BusinessLayer -> Concrete sağ tık -> Add -> Class -> BlogManager eklenir.

using BusinessLayer.Abstract;

using DataAccessLayer.Abstract;

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BusinessLayer.Concrete

{

public class BlogManager : IBlogService

{

IBlogDal \_blogDal;

public BlogManager(IBlogDal blogDal)

{

\_blogDal = blogDal;

}

public void BlogAdd(Blog blog)

{

throw new NotImplementedException();

}

public void BlogDelete(Blog blog)

{

throw new NotImplementedException();

}

public void BlogUpdate(Blog blog)

{

throw new NotImplementedException();

}

public Blog GetById(int id)

{

throw new NotImplementedException();

}

public List<Blog> GetList()

{

return \_blogDal.GetListAll();

}

}

}

1. CoreKampi -> Controllers -> BlogController sayfası açılır.

using BusinessLayer.Concrete;

using DataAccessLayer.EntitiyFramework;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace CoreKampi.Controllers

{

public class BlogController : Controller

{

BlogManager bm = new BlogManager(new EfBlogRepository());

public IActionResult Index()

{

var values = bm.GetList();

return View(values);

}

}

}

1. Index.cshtml de table oluşturalım.Verileri burda listeleyelim

@using EntityLayer.Concrete

@model List<Blog>

@{

ViewData["Title"] = "Index";

Layout = "~/Views/Shared/UserLayout.cshtml";

}

<h1>Index Sayfası</h1>

<br />

<table>

<tr>

<th>ID</th>

<th>Başlık</th>

</tr>

@foreach(var item in Model)

{

<tr>

<td>@item.BlogID</td>

<td>@item.BlogTitle</td>

</tr>

}

</table>

<br />

<br />

1. Veritabanında Blogs tablosuna kayıtlar girelim.
2. Daha önceden not defterine eklediğimiz kodu index sayfasına ekleyelim ve üstteki kodları bu yapıya göre yeniden yazalım

@using EntityLayer.Concrete

@model List<Blog>

@{

ViewData["Title"] = "Index";

Layout = "~/Views/Shared/UserLayout.cshtml";

}

<section class="main-content-w3layouts-agileits">

<div class="container">

<h3 class="tittle">Blog Posts</h3>

<div class="inner-sec">

<!--left-->

<div class="left-blog-info-w3layouts-agileits text-left">

<div class="row">

@foreach (var item in Model)

{

<div class="col-lg-4 card">

<a href="single.html">

<img src="@item.BlogImage" class="card-img-top img-fluid" alt="">

</a>

<div class="card-body">

<ul class="blog-icons my-4">

<li>

<a href="#">

<i class="far fa-calendar-alt"></i> @item.BlogCreateDate

</a>

</li>

<li class="mx-2">

<a href="#">

<i class="far fa-comment"></i> 21

</a>

</li>

<li>

<a href="#">

<i class="fas fa-eye"></i> @item.CategoryID

</a>

</li>

</ul>

<h5 class="card-title">

<a href="single.html">@item.BlogTitle</a>

</h5>

<p class="card-text mb-3">Lorem ipsum dolor sit amet consectetur adipisicing elit sedc dnmo eiusmod tempor incididunt.. </p>

<a href="single.html" class="btn btn-primary read-m">Daha Fazla</a>

</div>

</div>

<br />

}

</div>

<!--//left-->

</div>

</div>

</div>

</section>

#25 Asp.Net Core 5.0 Proje Kampı Include Metodu Ve Eager Load

1. @item.Category.CategoryName kısmı hata veriyordu bu problemi çözmek için include metodu kullanmak gerekir.
2. DataAccessLayer -> Abstract -> IBlogDal classına gidilir

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.Abstract

{

public interface IBlogDal:IGenericDal<Blog>

{

List<Blog> GetListWithCategory();//Category ile birlikte gel

}

}

1. DataAccessLayer -> EntityFramework -> EfBlogRepository

using DataAccessLayer.Abstract;

using DataAccessLayer.Concrete;

using DataAccessLayer.Repositories;

using EntityLayer.Concrete;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataAccessLayer.EntitiyFramework

{

public class EfBlogRepository : GenericRepository<Blog>, IBlogDal

{

public List<Blog> GetListWithCategory()

{

using (var c = new Context())

{

return c.Blogs.Include(x => x.Category).ToList();

}

}

}

}

1. DataAccessLayer -> Repositories -> BlogRepository, CategoryRepository ve CommentRepositoy class ları ile artık işimiz kalmadı o sayfaları silelim. Sadece GenericRepository kalacak.
2. BusinessLayer -> Abstract -> IBlogService te metodu çağıralım.

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BusinessLayer.Abstract

{

public interface IBlogService

{

void BlogAdd(Blog blog);

void BlogDelete(Blog blog);

void BlogUpdate(Blog blog);

List<Blog> GetList();

Blog GetById(int id);

List<Blog> GetBlogListWithCategory();

}

}

1. BusinessLayer -> Concrete -> BlogManager a gidelim.

using BusinessLayer.Abstract;

using DataAccessLayer.Abstract;

using EntityLayer.Concrete;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BusinessLayer.Concrete

{

public class BlogManager : IBlogService

{

IBlogDal \_blogDal;

public BlogManager(IBlogDal blogDal)

{

\_blogDal = blogDal;

}

public void BlogAdd(Blog blog)

{

throw new NotImplementedException();

}

public void BlogDelete(Blog blog)

{

throw new NotImplementedException();

}

public void BlogUpdate(Blog blog)

{

throw new NotImplementedException();

}

public List<Blog> GetBlogListWithCategory()

{

return \_blogDal.GetListWithCategory();

}

public Blog GetById(int id)

{

throw new NotImplementedException();

}

public List<Blog> GetList()

{

return \_blogDal.GetListAll();

}

}

}

1. CoreKampi -> Controllers -> BlogController a gidelim.

using BusinessLayer.Concrete;

using DataAccessLayer.EntitiyFramework;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace CoreKampi.Controllers

{

public class BlogController : Controller

{

BlogManager bm = new BlogManager(new EfBlogRepository());

public IActionResult Index()

{

var values = bm.GetBlogListWithCategory();

return View(values);

}

}

}

1. Index.csHtml de artık @item.Category.CategoryName yazabiliriz.

#26 Asp.Net Core 5.0 Proje Kampı Read More Ve Tarih Formatı

1. @item.BlogContent ile içerikleri de ekleyelim. Uzun bir içerik olduğu zaman içerikte kayma oluşuyor ama Daha Fazla butonuna basmadan sadece 4 satır görüntülensin istiyorsak aşağıdaki işlemi yapalım.
2. Öncelikle Veritabanında BlogContent alanlarına içerikler ekleyelim.
3. Tarih formatını değiştirelim.

@(((DateTime)item.BlogCreateDate).ToString("dd.MMM.yyyy"))

1. İçeriklere karakter sınırı getirelim.ilk 100 karakteri gösterelim.

@item.BlogContent.Substring(0,item.BlogContent.Substring(0,100).LastIndexOf(" "))...

#27 Asp.Net Core 5.0 Proje Kampı Blog Detayları

1. Devamını Oku kısmına basınca ilgili içeriğe gitmesi için buttonlara link verelim.

<a href="/Blog/BlogDetails/@item.BlogID" class="btn btn-primary read-m">Devamını Oku</a>