Ders 1 : Projenin Oluşturulması

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

Ders 2 – 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

Ders 3 – Sınıfların Oluşturulması

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; }

}

}

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; }

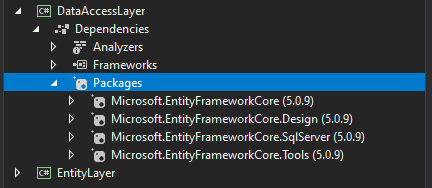
public bool AboutStatus { get; set; }

}

}

Ders 4 : Paketlerin İndirilmesi

1. DataAccessLayer (Sağ Tık ) 🡪 Add 🡪 Manage Nuget Packages 🡪Browse aşağıdaki paketler ilgili sürüme göre yüklenir.



1. Yukarıdaki paketler CoreKampi katmanına da kurulmalı
2. DataAccessLayer ( Sağ tık ) 🡪 Add 🡪 New Folder 🡪Concrete adında klasör oluştur.
3. DataAccessLayer ( Sağ tık ) 🡪 Add 🡪 New Folder 🡪Abstract adında klasör oluştur.
4. DataAccessLayer ( Sağ tık ) 🡪 Add 🡪 New Folder 🡪Repositories adında klasör oluştur.

Ders 5 : Connection String ve Context Sınıfı

1. DataAccessLayer 🡪 Concrete ( Sağ tık ) 🡪 Add 🡪Class 🡪Context.cs

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-BN2LEAS; database=CoreBlogDb;integrated security=true;");

}

}

}

Ders 6 : Referansların Tanımlanması:

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.

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-BN2LEAS;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; }

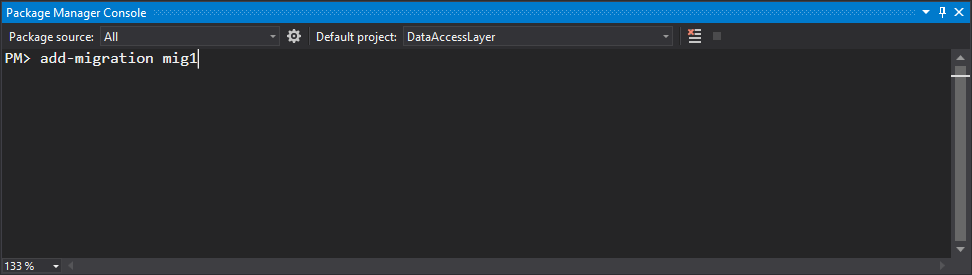
}

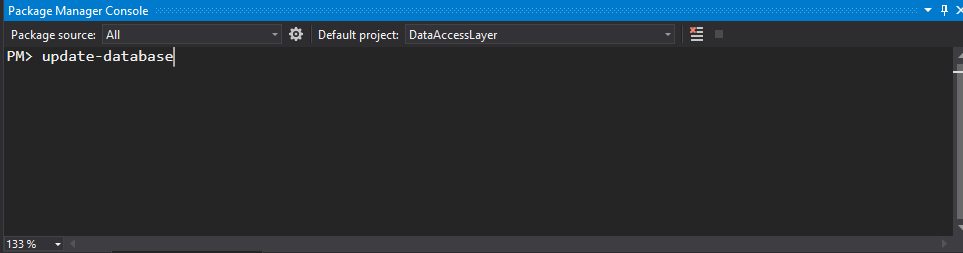
}

Ders 7 : Migration Tanımlanması

View -> Other Windows -> Package Manager Console seçilir.

Gelen alanda Default Project kısmında DataAccessLayer seçilir ve aşağıdaki kodlar yazılır.





Ders 8 : İlişkilerin Tanımlanması

1. Category ile Blog arasındaki ilişki ve Comment ile Blog arasındaki ilişki

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; }

}

}

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; }

}

}

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; }

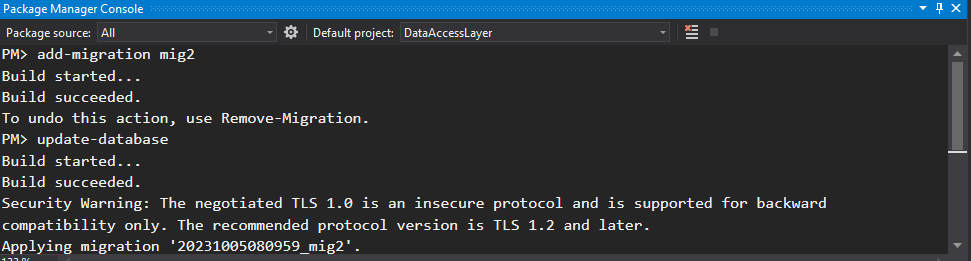
public int BlogID { get; set; }

public Blog Blog { get; set; }

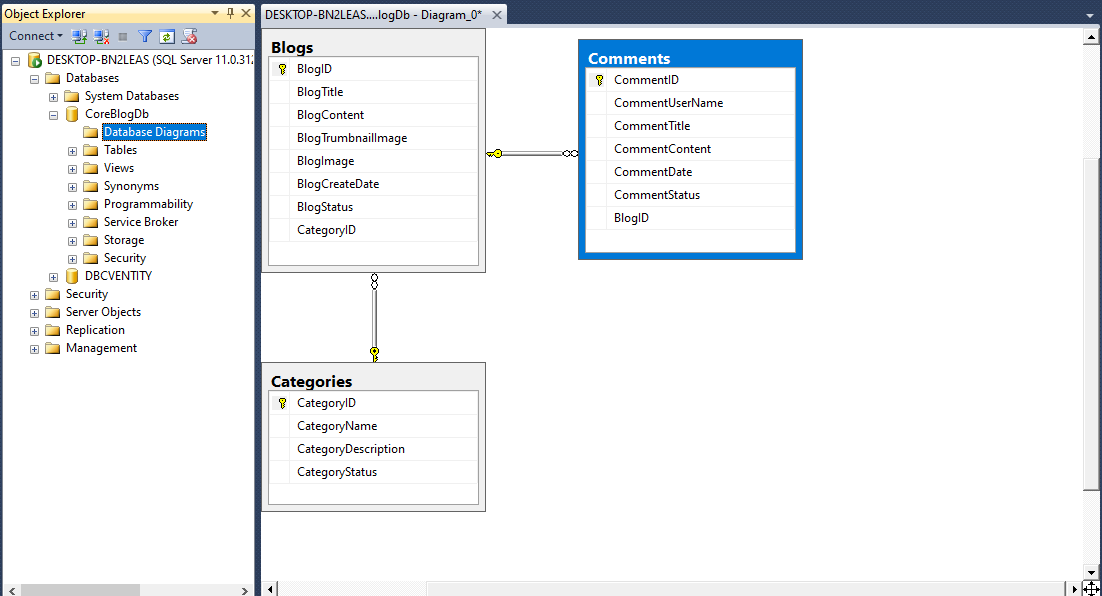
}

}

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

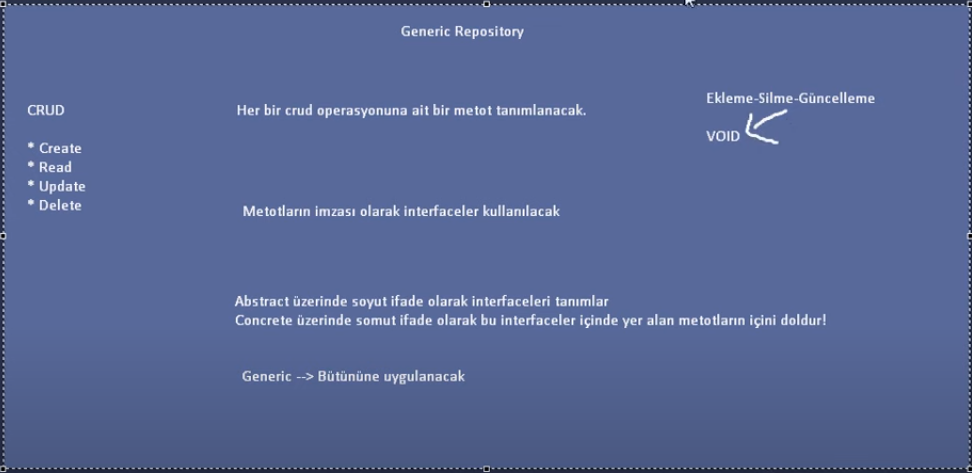


1. Veritabanında ilişki diyagramı aşağıdaki gibi olmuştur.



Ders 9 : 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

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

{

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();

}

}

}

Asp.core Ders 16 da kaldık…