E-Commerce

E-Ticaret projesi 6 projeden oluşan bir çözümlemedir. Katmanlı mimari ile tasarlanan projeler aşağıdadır.

**Ecommerce-API**

AspNet.Core 5.0 Web API restful servis projesidir.

**Ecommerce-BL**

Asp.Net.Core 5.0 Library projesidir. Ecommerse çözümlemesinin iş katmanıdır.

**Ecommerce-DAL**

Asp.Net.Core 5.0 Library projesidir. Ecommerse çözümlemesinin veri ulaşım katmanıdır.

**Ecommerce-DTO**

Asp.Net.Core 5.0 Library projesidir. Ecommerse çözümlemesinin data transfer objelerinin tanımlandığı katmanıdır. Arayüz katmanları bu objeler üzerinden sunum yaparlar.

**Ecommerce-Entity**

Asp.Net.Core 5.0 Library projesidir. Ecommerse çözümlemesinde veri tabanı nesnelerinin sınıflar ile tarif edildi alandır. Her bir sınıf veritabanında bir tablo desenini ifade eder.

**Ecommerce-Web**

Asp.Net.Core 5.0 Web MVC projesidir. E-Ticaret çözümünün arayüz uygulamasıdır.

Kullanılan Teknolojiler

Asp.Net Core 5.0, Asp.Net Core MVC, Asp.Net Core Web API, Entity Framework 5.0.17, SqlServer Express, Jquery, AdminLTE-3.2.0 Template, Bootsrap, Docker Desktop 4.13.1

appsettings.json

{

"ConnectionStrings": {

"connStr": "Server=localhost;Database=Ecommerce;User Id=sa;Password=Demo123456\*;"

},

"ApiUrl": {

"url": "https://localhost:44379"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*"

}

Yukarıdaki bağlantı noktası ile entity frame code first yaklaşımı ile tablolar oluşturulur.

20221109211029\_EcommerceInitilization.cs

public partial class EcommerceInitilization : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Categories",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Name = table.Column<string>(type: "nvarchar(50)", nullable: false),

Description = table.Column<string>(type: "nvarchar(MAX)", nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_Categories", x => x.Id);

});

migrationBuilder.CreateTable(

name: "Products",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

CategoryId = table.Column<int>(type: "int", nullable: false),

Name = table.Column<string>(type: "nvarchar(50)", nullable: false),

Price = table.Column<int>(type: "int", nullable: false),

Stock = table.Column<int>(type: "int", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Products", x => x.Id);

table.ForeignKey(

name: "FK\_Products\_Categories\_CategoryId",

column: x => x.CategoryId,

principalTable: "Categories",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateIndex(

name: "IX\_Products\_CategoryId",

table: "Products",

column: "CategoryId");

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "Products");

migrationBuilder.DropTable(

name: "Categories");

}

}

[DbContext(typeof(EcommerceContext))]

partial class EcommerceContextModelSnapshot : ModelSnapshot

{

protected override void BuildModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("Relational:MaxIdentifierLength", 128)

.HasAnnotation("ProductVersion", "5.0.17")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

modelBuilder.Entity("Ecommerce.Entity.Concrete.Category", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("Description")

.HasColumnType("nvarchar(MAX)");

b.Property<string>("Name")

.IsRequired()

.HasColumnType("nvarchar(50)");

b.HasKey("Id");

b.ToTable("Categories");

});

modelBuilder.Entity("Ecommerce.Entity.Concrete.Product", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("CategoryId")

.HasColumnType("int");

b.Property<string>("Name")

.IsRequired()

.HasColumnType("nvarchar(50)");

b.Property<int>("Price")

.HasColumnType("int");

b.Property<int>("Stock")

.HasColumnType("int");

b.HasKey("Id");

b.HasIndex("CategoryId");

b.ToTable("Products");

});

modelBuilder.Entity("Ecommerce.Entity.Concrete.Product", b =>

{

b.HasOne("Ecommerce.Entity.Concrete.Category", "Category")

.WithMany("Products")

.HasForeignKey("CategoryId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Category");

});

modelBuilder.Entity("Ecommerce.Entity.Concrete.Category", b =>

{

b.Navigation("Products");

});

#pragma warning restore 612, 618

}

}

Web API dokümantasyon dosyası Swager ile sağlanmıştır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

E-Commerce Uygulaması Geliştirme Yöntemi

1. Entity Katmanı oluşturuldu. Bu katmanda IEntity interface oluşturuldu. Veritabanı sınıflarının işaretlendiği interface dir. Concrete klasöründek her bir sınıf veritabanında tabloya bağlanmıştır.
2. DAL katmanı oluşturulur. IEntityRepository interface ile CRUD işlemlerinin imzaları oluşturulur. Bu katman içerisinde oluştulucak her bir intreface için base interface olarak tanımlanır. ICategoryDAL ve IProductDAL interface leri oluşturulur. CategoryDAL ve ProductDAL sınıfları ile veritabanı işlemleri yapılır.
3. BL katmanı oluşutulur. İş katmanında arayüzün ihtiyacı olan tüm metodlar tanımlanır. DAL katmanındaki tüm objeler bu katmanda kullanılır. Yapıcı metodlarda bu sınıflar inject edilir.
4. DTO katmanındaki sınıflar arayüz katmanında ortak bir sınıf olarak kullanılır.

metin, ekran görüntüsü, iç mekan içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu









