Analiza kodu StoreContext.cs

1. Wprowadzenie

Plik `StoreContext.cs` definiuje kontekst bazy danych w aplikacji ASP.NET Core przy użyciu **Entity Framework Core**. Klasa `StoreContext` zarządza dostępem do tabel w bazie danych oraz stosuje konfiguracje encji.

2. Importowanie przestrzeni nazw

```
using Core.Entities;
using Infrastructure.Config;
using Microsoft.EntityFrameworkCore;
```

- **Core.Entities** – zawiera definicję encji `Product`.
- **Infrastructure.Config** – zawiera konfigurację encji, np. `ProductConfiguration`.
- **Microsoft.EntityFrameworkCore** – zapewnia funkcjonalność ORM do pracy z bazą danych.

3. Definicja klasy StoreContext

```
namespace Infrastructure.Data;
```

- **namespace Infrastructure.Data** – określa przestrzeń nazw dla kontekstu bazy danych.

```
public class StoreContext(DbContextOptions options) : DbContext(options)
```

- **StoreContext dziedziczy po DbContext** – klasa zarządza połączeniem z bazą danych i dostępem do tabel.
- **DbContextOptions options** – konfiguracja bazy danych przekazywana do konstruktora.

Definicja DbSet:

public DbSet<Product> Products {get; set;}

- **DbSet<Product> Products** – reprezentuje tabelę `Products` w bazie danych.

4. Metoda OnModelCreating

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

    base.OnModelCreating(modelBuilder);

    modelBuilder.ApplyConfigurationsFromAssembly(typeof(ProductConfiguration).Assembly);

}

- **base.OnModelCreating(modelBuilder)** – wywołuje domyślne zachowanie `OnModelCreating`.

- **ApplyConfigurationsFromAssembly(typeof(ProductConfiguration).Assembly)** – automatycznie stosuje konfiguracje encji (`ProductConfiguration`) z danego zestawu.

5. Zastosowanie klasy StoreContext

Klasa `StoreContext` jest używana do zarządzania bazą danych i może być rejestrowana w kontenerze DI w `Program.cs`:

builder.Services.AddDbContext<StoreContext>(options =>

{

    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection"));

});

- **AddDbContext<StoreContext>** – rejestruje `StoreContext` jako usługę.

- **UseSqlServer** – określa użycie SQL Server jako silnika bazy danych.

- **GetConnectionString("DefaultConnection")** – pobiera konfigurację połączenia z pliku `appsettings.json`.

6. Podsumowanie

- `StoreContext` zarządza połączeniem z bazą danych i dostępem do encji.

- Stosuje konfiguracje encji przy użyciu `ApplyConfigurationsFromAssembly`.

- Może być rejestrowany w kontenerze DI, umożliwiając łatwe zarządzanie dostępem do danych.

Jest to kluczowy element architektury aplikacji ASP.NET Core, zapewniający centralne zarządzanie operacjami na bazie danych.