01

DataService, Injeção de dependência

Transcrição

Poderíamos aproveitar que estamos criando o banco de dados automaticamente para implementarmos um método para a carga destes dados, e alimentarmos nosso banco com o catálogo inicial de produtos. Porém, com isso estaríamos violando o princípio de **separação de responsabilidades**.

Sendo assim, criaremos uma classe específica (`DataService`) para não poluirmos a classe `Startup` , com um método público denominado `InicializaDB()` para inicializar o banco de dados, caso seja necessário. Para isso, ele precisará acessar o contexto, `ApplicationContext` .

Portanto, criaremos um campo privado e, para que o contexto seja inicializado, criaremos também uma instância. Poderíamos usar simplesmente `private readonly ApplicationContext contexto = new ApplicationContext` , mas não o faremos porque queremos usar o sistema de injeção de dependências do ASP.NET Core.

Selecionaremos o campo privado `contexto` usando o atalho "Ctrl + ." e gerando um construtor, que tomará o parâmetro `ApplicationContext contexto` como obrigatório na criação do objeto `DataService` , e então teremos a instância do contexto:

```
class DataService
{
    private readonly ApplicationContext contexto;

    public DataService(ApplicationContext contexto)
    {
        this.contexto = contexto;
    }

    public void InicializaDB()
    {
        contexto.Database.EnsureCreated();
    }
}
```

Por padrão, o `ApplicationContext` é gerado pela injeção de dependência, assim, é possível acessar o objeto `contexto` no método `InicializaDB()` . Para garantir que ele seja criado, usaremos `EnsureCreated()` , e com isso chamaremos o `DataService` , em vez de fazer a criação do banco de dados diretamente na classe `Startup` .

Queremos que se crie uma instância do serviço de dados, para o qual basta substituirmos `ApplicationContext` por `DataService` em `serviceProvider.GetService<ApplicationContext>().Database.EnsureCreated()` , mais acima no código.

Em seguida, chamaremos o novo método, `InicializaDB()` , deixando a linha de código da seguinte maneira: `serviceProvider.GetService<DataService>().InicializaDB()` . Porém, para o serviço de injeção de dependência do ASP.NET Core funcionar, é necessário registrar a nova classe, isto é, `DataService` , no contêiner de injeção de dependências.

Iremos ao método `ConfigureServices()` da classe `Startup` e faremos o registro chamando um método que adiciona uma instância, a qual queremos que exista somente enquanto os objetos que a utilizarem estiverem ativos. Este método é `AddTransient` , que significa adicionar uma instância temporária.

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();

    string connectionString = Configuration.GetConnectionString("Default");

    services.AddDbContext<ApplicationContext>(options =>

        options.UseSqlServer(connectionString)
    );

    services.AddTransient<DataService>();
}
```

Normalmente, trabalhamos com interfaces para a injeção de independência, então moveremos a classe `DataService` para um novo arquivo utilizando "Ctrl + ." e escolhendo a opção "Move type to DataService.cs". Vamos acessá-lo pelo painel de "Solution Explorer", do lado direito.

Criaremos uma interface a partir deste novo arquivo, com "Ctrl + ." e "Extract Interface...". Na janela que se abre, clicaremos em "OK", e a interface será extraída. Vamos voltar a `Startup.cs` e, no registro da injeção de dependência, incluiremos o tipo da interface: `services.AddTransient<IDataService, DataService>();` .