01

Criando o Modelo

Transcrição

> Começando deste ponto? No link a seguir, você pode fazer o DOWNLOAD completo do projeto (https://github.com/alura-cursos/ASPNETCore20/archive/a5ff7c49f58fae9c75f1d37c0ea5b8ade358156d.zip) para continuar seus estudos a partir deste capítulo.

Já temos a aplicação ASP.NET Core MVC rodando, porém, por enquanto temos somente as views, ou seja, as páginas com conteúdo estático em HTML. O próximo passo será passar os dados para estas views, para podermos formar uma página de verdade.

Teremos que criar um modelo, que fornecerá estes dados para o MVC. Para isso, serão necessárias determinadas classes. Normalmente, é necessária a criação de um banco de dados, que nos permita acessar estes dados. Contudo, faremos uma abordagem diferente, chamada *code first*, ou código primeiro. Isso significa que criaremos as classes e, em seguida, criaremos o banco de dados com base nestas.

As classes são representadas por quatro entidades, que aqui são: `Pedido`, `Cadastro`, `ItemPedido` e `Produto`. No `Pedido`, temos os itens do carrinho, e associado a ele temos o `Cadastro`, com os dados do usuário. Temos também o `ItemPedido`, com as peculiaridades do pedido, como quantidade e preço e, por fim, temos o `Produto`, que são os itens presentes no catálogo, no carrossel.

Cada uma das classes possui um atributo, uma propriedade, chamada `id`, que serve como chave primária para esta entidade. Temos ainda, uma propriedade de navegação, responsáveis por realizar a associação entre uma classe e outra. Por exemplo, temos o `Pedido` relacionado a `Cadastro`. Ou seja, é possível acessar o `Cadastro` a partir de uma classe `Pedido`, com os dados do usuário.

Retornaremos ao `Visual Studio`, e abriremos a pasta "_Recursos > dados".

Nela, encontramos um arquivo chamado `modelo.cs`. Nele, há diversos exemplos de classes, que são justamente aqueles abordados acima. Copiaremos este arquivo e colaremos na pasta "CasaDoCodigo > Models".

Contudo, o modelo, sozinho, é incapaz de fazer algo. Precisamos fazer ainda uma associação com o banco de dados. Para que possamos gerar tabelas a partir do nosso código, precisamos de um intermediário, que chamamos de **Entity Framework Core**. Ele faz o mapeamento do objeto relacional, e gerará a partir das classes as tabelas, campos e chaves necessários no banco de dados. Posteriormente, ele gerenciará esta troca de informações entre o banco e o sistema. Ele é fundamental para fazermos o acesso aos dados.

Assim, temos as classes do modelo, e precisamos agora criar um contexto para o Entity Framework Core saber como lidar com as entidades para fazer o mapeamento. Para isso, clicaremos com o botão direito do mouse sobre a pasta "CasaDoCodigo" e selecionaremos a opção "Add > Class...". A chamaremos de `ApplicationContext`.

Esta nova classe precisa herdar de `DbContext`, lembrando de importar o `using`. Precisamos inserir ainda um construtor, com o parâmetro `options`. Clicaremos com o botão direito do mouse sobre `Generate constructor 'ApplicationContext(options)'`. Removeremos o atributo `NotNullAtribute` pois ele não será necessário. Criaremos também um `override` para substituirmos um método, e assim criarmos o modelo.

```
namespace CasaDoCodigo
{
    public class ApplicationContext : DbContext
    {
        public ApplicationContext(DbContextOptions options) : base(options)
        {
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);
        }
    }
}
```

Quando estiver criando o modelo, ele acessará este método para poder fazer o mapeamento. Dando continuidade, faremos o mapeamento a partir do parâmetro `modelBuilder` e chamaremos o método `Entity`, que registrará uma classe do nosso modelo, no caso `Produto`, e o mapearemos para que possa ser adicionado ao modelo. Caso contrário, o Entity Framework não saberá que classes utilizará para gerar o banco de dados.

Para registrar que o método contém uma chave primária, chamaremos o método `HasKey` e o nome da propriedade que representa a referida chave, ou seja, `(t => t.Id)`:

```
namespace CasaDoCodigo
{
    public class ApplicationContext : DbContext
    {
        public ApplicationContext(DbContextOptions options) : base(options)
        {
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            modelBuilder.Entity<Produto>().HasKey(t => t.Id);
        }
    }
}
```