09

# Injetando Modelo na View Carrossel

### Transcrição

Chegou a hora de injetarmos o nosso modelo na *view* para gerarmos uma página dinâmica. E quem faz esta injeção é o *controller*, cujo pedido acessará o banco de dados para passar o modelo à *view* por meio do repositório de produtos.

Assim, modificaremos nosso repositório para podermos criar um método para a obtenção de produtos. Na interface `IProdutoRepository`, criaremos o método `GetProdutos()`, a partir do qual retornaremos uma lista contendo todos os produtos do catálogo.

```
public interface IProdutoRepository
{
    void SaveProdutos(List<Livro> livros);
    IList<Produto> GetProdutos();
}
```

Feito isso, iremos à classe `ProdutoRepository` e forçaremos a implementação desta interface com o novo método. Com `IProdutoRepository` selecionado, usaremos "Ctrl + ." e optaremos por "Implement interface", e será criado o método `GetProdutos()`, que acessará o Dbset dos produtos, retornando-o em uma lista:

```
public IList<Produto> GetProdutos()
{
    return contexto.Set<Produto>().ToList();
}
```

Iremos ao *controller* para consumirmos este método do repositório e pegarmos os produtos. Para isso, acessaremos `PedidoController.cs`, e criaremos um campo privado que receberá o valor quando fizermos a injeção de dependência. Também criaremos um construtor a partir deste campo, selecionando `private readonly IProdutoRepository produtoRepository` e usando "Ctrl + .".

```
public class PedidoController : Controller
{
    private readonly IProdutoRepository produtoRepository;

    public PedidoController(IProdutoRepository produtoRepository)
    {
        this.produtoRepository = produtoRepository;
    }
}
```

Agora consumiremos um método de `produtoRepository` na *action* de `Carrossel()`, cujo retorno será injetado como parâmetro do método `View()`:

```
public IActionResult Carrossel()
{
```

```
        return View(produtoRepository.GetProdutos());
    }
```

Feito isso, modificaremos a *view* de `Carrolsel()` para que esta listagem seja exibida na tela. Abriremos "Views > Pedido > Carrossel.cshtml" no painel "Solution Explorer", que se encontra do lado direito. Logo após a linha com `<h3>Catálogo</h3>`, deixaremos um espaço para renderizar a listagem de *views* na tela.

Usaremos `@` para indicar que estamos colocando um código C# em um arquivo `.cshtml`. Em seguida, incluiremos uma instrução C#, que no caso será a varredura da nossa lista de produtos, o nosso modelo. Em `foreach`, poderemos colocar um HTML, dentro do qual incluiremos um trecho em C#.

```
@foreach (var produto in Model)
{
    <div>@produto.</div>
}
```

Porém, o IntelliSense não reconhece ao digitarmos `@produto.`, por não termos descrito o tipo ou modelo desta *view* de carrossel. Teremos que incluir a diretiva `@model`, que exige o tipo de modelo que iremos injetar na *view*, isto é, a lista de produtos.

```
@model List<Produto>;

<h3>Catálogo</h3>

@foreach (var produto in Model)
{
    <div>@produto.Nome</div>
}
```

Desta vez, o programa nos mostra as opções ao digitarmos, e então rodaremos a aplicação com a tecla "F5" para verificar a diferença na nossa tela. A parte dos produtos será carregada e exibida na *view*. Com isso, temos o catálogo, e cada um dos produtos contidos no banco de dados.

Adiante, passaremos os produtos para baixo, onde se localiza o carrossel com as informações tais como nome e preço.