03

Injetando o Cadastro na View e Redirecionando se Necessário

Transcrição

Nesta aula, faremos com o que `PedidoController.cs` passe o modelo para a view `Cadastro.cshtml`. Em `PedidoController.cs`, trabalharemos no método de `Cadastro()`.

```
public IActionResult Cadastro()
{
    return View();
}
```

Primeiramente pegaremos o objeto `cadastro` que se encontra em `Pedido`, o acessaremos por meio do repositório. Escreveremos `pedidoRepository` e coletemos o pedido utilizando o método `getPedido()`. Armazenaremos o objeto dentro de uma variável local `pedido`.

```
public IActionResult Cadastro()
{
    var pedido = pedidoRepository.GetPedido();

    return View();
}
```

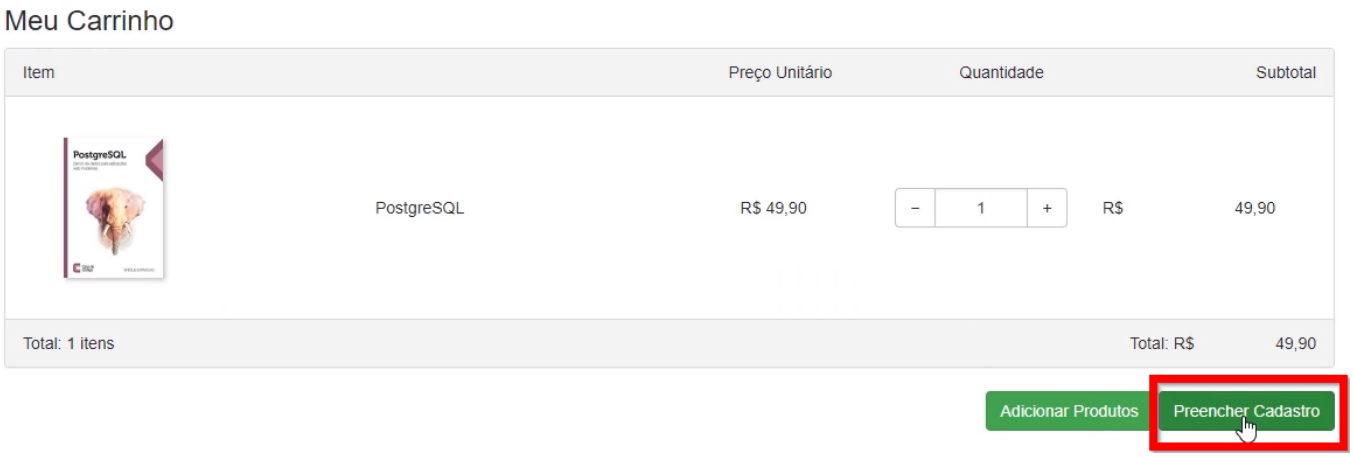
Em seguida, passaremos para a `View()` o cadastro acessado por meio de `pedido.Cadastro`. Contudo, existe uma situação em que `pedido` pode ser nulo, nesse caso não podemos permtir que o usuário prossiga na compra, portanto o direcionaremos para a página inicial do projeto, isto é, a `Carrossel`. Utilizaremos o método `RedirectToAction()` e passaremos como parâmetro a action de destino `Carrossel`.

```
public IActionResult Cadastro()
{
    var pedido = pedidoRepository.GetPedido();

    if (pedido == null)
    {
        return RedirectToAction("Carrossel");
    }

    return View(pedido.Cadastro);
}
```

Executaremos a nossa aplicação partido da página inicial e escolheremos um item qualquer para adicioná-lo ao carrinho. Assim feito, clicaremos sobre o botão "Preencher Cadastro" para acessarmos a tela de formulário.



Na página de cadastro teremos os formulários em branco a serem preenchidos

Clicaremos com o botão direito sobre a página e selecionaremos a opção "Inspecionar", desse modo conseguiremos visualizar o código html gerado pelo servidor.

Encontraremos a tag `<form>` gerada a partir da tag helper. Teremos a `action` marcada como `resumo`, mas o Asp.NET gerou a `action` corretamente no html como `/Pedido/resumo`.

```
<h3>Cadastro</h3>
<form method="post" action="/Pedido/resumo" novalidate="novalidate"> == $0
```

Mais abaixo veremos que para cada um dos campos foi criado um input html com uma série de informações. Veremos que o input de `Nome` possui uma série de validações que se encontram no `modelo.cs`, como, por exemplo, que o nome é obrigatório e precisa ser preenchido. (`"The Nome field is required."`)

```
<input type="text" class="form-control" id="nomeCliente" placeholder="Nome do Cliente" placeholder='
```

Ao observarmos o nosso arquivo `modelo.cs` veremos que em `Cadastro()` teremos os campos marcados como obrigatórios.

```
public Cadastro()
{
}

public virtual Pedido Pedido { get; set; }
[Required]
```

```
public string Nome { get; set; } = "";
[Required]
public string Email { get; set; } ="";
[Required]
public virtual Telefone { get; set; } = "";
[Required]

<****!****>
```

O Asp.NET Core facilita nosso trabalho ao inserir uma série de tags html para indicar que um campo é obrigatório.