

## Obtendo ou criando novo pedido

### Transcrição

Começando deste ponto? Você pode fazer o [DOWNLOAD completo do projeto \(https://github.com/alura-cursos/ASPNETCore20/archive/a93411b9c9eedf5b343d766547f329a3e43bde11.zip\)](https://github.com/alura-cursos/ASPNETCore20/archive/a93411b9c9eedf5b343d766547f329a3e43bde11.zip) para continuar seus estudos a partir deste capítulo.

Se estamos acessando a *action* do carrossel — o `produtoRepository` — para pegarmos os produtos do catálogo, de que maneira chegaremos às informações para exibirmos o carrinho na *action* correspondente?

Teremos que acessar o repositório referente aos pedidos, para o qual é necessário modificar nosso `PedidoController()` e criar um novo repositório como um campo privado. Faremos uma cópia deste repositório, declararemos um `IPedidoRepository`, passando-o em seguida por uma injeção de dependência nesta classe, adicionando um novo parâmetro no construtor.

E, então, o atribuiremos no corpo do construtor e acessaremos o repositório de pedido na *action* de carrinho, criando uma variável local chamada `Pedido`. Chamaremos o método para obtermos o pedido atual da sessão com `GetPedido()`, que ainda não existe, pois não o criamos.

O código estará assim:

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

    public PedidoController(IProdutoRepository produtoRepository,
        IPedidoRepository pedidoRepository)
    {
        this.produtoRepository = produtoRepository;
        this.pedidoRepository = pedidoRepository;
    }

    public IActionResult Carrossel()
    {
        return View(produtoRepository.GetProdutos());
    }

    public IActionResult Carrinho()
    {
        Pedido pedido = pedidoRepository.GetPedido();
        return View();
    }
}
```

Selecionaremos `GetPedido`, usaremos "Ctrl + ." e clicaremos na opção "Gerar método `IPedidoRepository.GetPedido`", com a qual o Visual Studio criará o método. Navegaremos à interface `IPedidoRepository`, e faremos a classe concreta

PedidoRepository implementar a interface com o novo método.

Com o mouse sobre IPedidoRepository na linha `public class PedidoRepository : BaseRepository<Pedido>`, `IPedidoRepository`, usaremos "Ctrl + ." e escolheremos "Implementar interface". Teremos a interface abaixo pronta para implementação:

```
public Pedido GetPedido()
{
    throw new NotImplementedException();
}
```

Como faremos para obtermos o pedido atual?

O primeiro passo é verificar qual `id` do pedido está gravado na sessão. Vamos declarar uma variável local com `GetPedidoId()`, o método que obtém a informação da sessão. Além disso, consultaremos a nossa tabela de pedidos para saber se este `pedidoId` existe — se não existir, criaremos um novo pedido a ser retornado.

Assim, declararemos a variável `pedido`, do `DbSet` de pedido. Criaremos um filtro com uma consulta LINQ para descobrirmos o pedido que possui o `id` correspondente. Na expressão lambda, usaremos `p => p.Id == pedidoId`, em que `pedidoId` se refere à sessão.

Com o método `SingleOrDefault()`, verificaremos se o elemento existe. Ele retornará o pedido, caso o elemento exista. Caso contrário o retorno será nulo, sem gerar erros. Na sequência, verificaremos se o pedido é nulo ou não. Se sim, será necessário criá-lo e, por este motivo, usaremos o `if`. Para a persistência no banco de dados, utilizaremos `contexto.SaveChanges()`.

No caso do pedido não ser nulo, isto é, se houver alguma informação, retornaremos `pedido` a partir de `GetPedido()`.

```
public Pedido GetPedido()
{
    var pedidoId = GetPedidoId();
    var pedido = dbSet.Where(p => p.Id == pedidoId)
                        .SingleOrDefault();

    if (pedido == null)
    {
        pedido = new Pedido();
        dbSet.Add(pedido);
        contexto.SaveChanges();
    }

    return pedido;
}
```

Voltando a `PedidoController.cs`, passaremos a informação dos itens do pedido que acabamos de obter do repositório para a `view`. E injetaremos o parâmetro `pedido.Itens` no método `View()`, isto é, os itens do pedido obtidos do repositório:

```
public IActionResult Carrinho()
{
    Pedido pedido = pedidoRepository.GetPedido();
```

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