02

Criando o backend do Carrinho

Transcrição

Agora podemos obter o item que o usuário selecionou e adicionar a um carrinho de compras.

Abra o `livro-detalhe.xhtml` e onde temos o botão de **Comprar**, modifique o valor de `jsf:action` para `#{carrinhoComprasBean.add()}`. Nesse código modificamos o valor `adicionar` para `add`, nada demais, apenas para resumir. Porém perceba que agora temos os parêntesis, pois queremos invocar o método direto. Não apenas isso mas também passar o `ID` do `Livro` para que o processo de adição saiba qual livro colocar no carrinho.

Para passar o `ID`, vamos passar o código do ID da seguinte forma: `#{carrinhoComprasBean.add(livroDetalheBean.id)}`. O **JSF** consegue passar o parâmetro direto na chamada do método.

Como ainda não temos o *ManagedBean* relacionado, vamos criá-lo. Pressione `Ctrl + N` e escolhar `Class`. O nome da classe será `CarrinhoComprasBean`. Após ter a classe criada, adicione a *annotation* `@Model` na classe. Crie também um método novo, chamado `public String add(Integer id)`.

Nesse método, temos que pegar o `Livro` pelo `ID`. Vamos chamar o método `buscaPorId` do `LivroDao` para nos devolver o `Livro` que precisamos. Você deve perceber também que ainda não injetamos o próprio `LivroDao`, vamos fazer isso. Adicione um atributo de classe para o livroDao e utilize annotation de injeção de dependência.

Além disso temos também que pedir para injetar o carrinho de compras que usaremos de agora em diante. Crie também o atributo de classe para a classe `CarrinhoCompras` e chame o atributo de `carrinho` e faça a injeção da dependência. Esse objeto ainda não existe, então vamos pedir para o Eclipse criá-lo com `Ctrl + 1` e escolhendo a opção `Create class 'CarrinhoCompras'`. Escolha o pacote de `models` e pressione "Finish".

Nessa classe, poderemos ter uma lista de Livros para que possamos ir guardando os livros adicionados no carrinho. Porém, ainda temos que saber a quantidade que o usuário quer de um determinado item. Como o Livro não guarda a quantidade, vamos usar um objeto chamado `CarrinhoItem`. Mas também não queremos que os itens do carrinho se repitam, pois queremos a quantidade e o livro para conseguir montar o carrinho corretamente. Por isso, usaremos um `Set` de `CarrinhoItem`, para garantir que esse cara não se repetirá, pois o `Set` é uma estrutura de dados que não permite repetições.

Crie também na classe `CarrinhoCompras` o método `add` que usaremos para adicionar um `CarrinhoItem` ao carrinho. Dentro desse método apenas adicionamos o parâmetro recebido ao `Set`. Para que possamos acompanhar mais de perto, o código da classe `CarrinhoCompras` até o momento ficaria assim:

```
public class CarrinhoCompras {

    private Set<CarrinhoItem> itens = new HashSet<>();

    public void add(CarrinhoItem item) {
        itens.add(item);
    }

}
```

A classe `CarrinhoItem` ainda não foi criada, vamos criá-la. Utilize o mesmo atalho que usamos da outra vez. Com a classe criada, vamos adicionar algumas propriedades a essa classe.

Adicione o atributo `private Livro livro;` e também a quantidade `private Integer quantidade` . Gere os *getters and setters* desses atributos.

Quando utilizamos uma estrutura de dados baseada em **Hash** como estamos fazendo com o `HashSet` acima, precisamos gerar o método `hashCode` da classe que estamos usando. Como o `CarrinhoItem` é o elemento do `HashSet` , vamos pedir para o Eclipse gerar esse método bem como o seu par `equals` . Pressione `Ctrl + 3` e então digite `hashCode` . Uma opção chamada `Generate hashCode() and equals()` deve ter aparecido para vocês, dê **Enter** nessa opção, deixe selecionado apenas o `Livro` e logo em seguida **Finish**.

Você pode ter percebido que o Eclipse alertou sobre a classe `Livro` que ela não possui `hashCode` . Vamos gerar no Livro também pois o `hashCode` do `CarrinhoItem` usa o do `Livro` . Abra a classe `Livro` e ao final da classe, faça os mesmo passos: `Ctrl + 3` digite `hashCode` e pressione **Enter**. Selecione apenas o `ID` como item que queremos considerar no `hashCode` .

Agora temos o `hashCode` do Livro baseado no ID e o hashCode do CarrinhoItem que é baseado apenas no `Livro` . Temos praticamente tudo pronto para adicionar o Livro, porém ainda temos que transformar o `Livro` que pegamos no **MB** ao Carrinho. Vamos voltar para a classe `CarrinhoComprasBean` , e no método `add` , crie um novo `CarrinhoItem` e pelo construtor dele já adicione o `Livro` (criaremos o construtor em breve). Com a instância do `CarrinhoItem` , podemos chamar o `carrinho` que já temos para adicionar `carrinho.add(carrinhoItem)` .

Ao final do método, vamos enviar o usuário para a tela de carrinho. Coloque o `return` para `"carrinho?faces-redirect=true` . Nosso código do método `CarrinhoComprasBean` até o momento deve estar assim:

```
@Model
public class CarrinhoComprasBean {

    @Inject
    private LivroDao livroDao;

    @Inject
    private CarrinhoCompras carrinho;

    public String add(Integer id) {
        Livro livro = livroDao.buscarPorId(id);
        CarrinhoItem item = new CarrinhoItem(livro);
        carrinho.add(item);

        return "carrinho?faces-redirect=true";
    }
}
```

Ainda estamos com problemas no construtor que não criamos do `CarrinhoItem` . Vá para a linha com erro e pressione `Ctrl + 1` a opção `Create constructor ...` . Dentro do construtor, basta colocar o livro recebido como parâmetro para o atributo livro. Para termos uma ideia geral da classe `CarrinhoItem` com os atributos e construtor, nossa classe ficou assim:

```
public class CarrinhoItem {
```

```
    private Livro livro;
    private Integer quantidade;

    public CarrinhoItem(Livro livro) {
        this.livro = livro;
        this.quantidade = 1;
    }

    // Getters e Setters aqui...

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((livro == null) ? 0 : livro.hashCode());
        return result;
    }
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        CarrinhoItem other = (CarrinhoItem) obj;
        if (livro == null) {
            if (other.livro != null)
                return false;
        } else if (!livro.equals(other.livro))
            return false;
        return true;
    }

}
```

Vamos continuar com a construção da tela no próximo vídeo.