05

Mãos na massa: Finalizando o carrinho

Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

1) Crie a seção de pagamentos, começando pelo `PagamentoController`, em `br.com.casadocodigo.loja.controllers`, com o método `finalizar`, que deve imprimir o total do carrinho e adicionar uma mensagem de *flash* com o conteúdo **Pagamento realizado com sucesso**:

```
@RequestMapping("/pagamento")
@Controller
@Scope(value = WebApplicationContext.SCOPE_REQUEST)
public class PagamentoController {

    @Autowired
    private CarrinhoCompras carrinho;

    @RequestMapping(value="/finalizar", method=RequestMethod.POST)
    public ModelAndView finalizar(RedirectAttributes model) {

        System.out.println(carrinho.getTotal());
        model.addFlashAttribute("sucesso", "Pagamento realizado com sucesso");

        return new ModelAndView("redirect:/produtos");
    }
}
```

2) Em **itens.jsp**, coloque o botão de finalização de compra dentro de um `form` e altere a sua `action`:

```
<tfoot>
    <tr>
        <td colspan="3">
            <form action="${s:mvcUrl('PC#finalizar').build()}" method="post">
                <input type="submit" class="checkout" name="checkout"
                    value="Finalizar compra" />
            </form>
        </td>
        <td class="numeric-cell">${carrinhoCompras.total}</td>
        <td></td>
    </tr>
</tfoot>
```

3) Todo *controller* que precisar acessar o carrinho, você deve lembrar de mudar o seu escopo. Existe uma alternativa para resolver esse problema, você pode mudar o escopo do *controller* ou adicionar um novo atributo na classe **`CarrinhoCompras`**, que é o **`proxyMode`**, com o valor **`TARGET_CLASS`**:

```
@Component
@Scope(value=WebApplicationContext.SCOPE_SESSION,
    proxyMode=ScopedProxyMode.TARGET_CLASS)
public class CarrinhoCompras implements Serializable {
```

```
    // código da classe omitido

}
```

4) Ainda falta conseguir remover os produtos do carrinho. Na classe **`CarrinhoCompras`**, crie o método **`remover`**, que recebe o `id` do produto e o seu tipo:

```
public void remover(Integer produtoId, TipoPreco tipoPreco) {
    Produto produto = new Produto();
    produto.setId(produtoId);
    itens.remove(new CarrinhoItem(produto, tipoPreco));
}
```

5) Crie o método da classe `CarrinhoComprasController` que irá invocar o método `remover` da classe `Carrinho` e anote-o com `@RequestMapping`:

```
@RequestMapping("/remover")
public ModelAndView remover(Integer produtoId, TipoPreco tipoPreco) {
    carrinho.remover(produtoId, tipoPreco);
    return new ModelAndView("redirect:/carrinho");
}
```

6) Não esqueça também de alterar a `action` do `form` da JSP **item.jsp**:

```
<td class="remove-item">
    <form action="${s:mvcUrl('CCC#remover').arg(0, item.produto.id).arg(1, item.tipoPreco).build
        method="POST">
        <input type="image" src="${contextPath}/resources/imagens/excluir.png"
            alt="Excluir" title="Excluir" />
    </form>
</td>
```

7) Agora, envie o pagamento para um serviço. Para isso, será necessário converter algumas classes para JSON, e para facilitar esse processo, use o **Jackson**. Adicione o seguinte XML dentro do seu arquivo **pom.xml**:

```
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>2.5.1</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.5.1</version>
</dependency>
```

8) Com o Jackson adicionado ao projeto, consuma um serviço. Na classe **`PagamentoController`**, no método **`finalizar`**, envie os dados do pagamento para a URL **`http://book-payment.herokuapp.com/payment`**, que recebe um JSON com o

formato abaixo:

```
{"value": 500}
```

Use a classe **`RestTemplate`** para acessar o serviço:

```
@RequestMapping("/pagamento")
@Controller
@Scope(value = WebApplicationContext.SCOPE_REQUEST)
public class PagamentoController {

    @Autowired
    private CarrinhoCompras carrinho;

    @Autowired
    private RestTemplate restTemplate;

    @RequestMapping(value="/finalizar", method=RequestMethod.POST)
    public Callable<ModelAndView> finalizar(RedirectAttributes model){
        return () -> {
            String uri = "http://book-payment.herokuapp.com/payment";

            try {
                String response = restTemplate.postForObject(uri,
                    new DadosPagamento(carrinho.getTotal()), String.class);
                model.addFlashAttribute("sucesso", response);
                System.out.println(response);
                return new ModelAndView("redirect:/produtos");
            } catch (HttpClientErrorException e) {
                e.printStackTrace();
                model.addFlashAttribute("falha", "Valor maior que o permitido");
                return new ModelAndView("redirect:/produtos");
            }
        };
    }
}
```

9) No pacote `br.com.casadocodigo.loja.models` , crie a classe **`DadosPagamento`** :

```
public class DadosPagamento {

    private BigDecimal value;

    public DadosPagamento(BigDecimal value) {
        this.value = value;
    }

    public DadosPagamento() {
    }

    public BigDecimal getValue() {
        return value;
    }
}
```

10) Na classe `AppWebConfiguration`, crie um novo método que retorne um `RestTemplate`:

```
@Bean
public RestTemplate restTemplate() {
    return new RestTemplate();
}
```

11) Na classe **`CarrinhoItem`**, implemente **`Serializable`**:

```
public class CarrinhoItem implements Serializable {

    private static final long serialVersionUID = 1L;

    // restante do código da classe omitido

}
```

12) Em **lista.jsp**, exiba a falha, se houver:

```
<body>
    <h1></h1>

    <div>${sucesso}</div>
    <div>${falha}</div>

<!-- restante do código omitido-->
```