04

Finalizando o código

Transcrição

Agora precisamos mostrar o valor total do carrinho, até aqui só calculamos o total de um produto específico. Para calcular o total do carrinho, só precisamos fazer um laço na classe `CarrinhoCompras` que a cada iteração chame o método `getTotal` criado anteriormente. Passando como parâmetro o produto atual e guardando o resultado de cada iteração em uma variável. Este método também se chamará `getTotal` , mas não receberá parâmetros. Vejamos no código como fica:

```
public BigDecimal getTotal(){
    BigDecimal total = BigDecimal.ZERO;
    for (CarrinhoItem item : itens.keySet()) {
        total = total.add(getTotal(item));
    }
    return total;
}
```

Para podermos testar, vamos ajustar outros dois detalhes que são: O `CarrinhoComprasController` não tem nenhum método mapeando a página de itens do carrinho e vamos fazer com que ao adicionar um produto, o usuário seja redirecionado para a página de itens do carrinho.

Para o primeiro ponto, precisaremos de um novo método. Chamaremos este método de `itens` e o mapeamento será o mesmo do controller que é `/carrinho` . Neste caso especificaremos apenas o método `HTTP` que será `GET` . Este método apenas cria um objeto `ModelAndView` direcionando a requisição para a página `itens.jsp` . Vejamos em código:

```
@RequestMapping(method=RequestMethod.GET)
public ModelAndView itens(){
    return new ModelAndView("/carrinho/itens");
}
```

Observação: Lembre-se que esta adição e a próxima modificação é para ser feita na classe `CarrinhoComprasController` .

E para o ajuste do segundo ponto, precisamos apenas alterar o método `add` para que em vez de `redirect:/produtos` este faça `redirect:/carrinho` .

```
@RequestMapping("/add")
public ModelAndView add(Integer produtoId, TipoPreco tipo){
    ModelAndView modelAndView = new ModelAndView("redirect:/carrinho");
    CarrinhoItem carrinhoItem = criaItem(produtoId, tipo);
    carrinho.add(carrinhoItem);
    return modelAndView;
}
```

Se iniciarmos o servidor agora, teremos um erro do tipo `NotSerializableException` informando que o objeto da classe `CarrinhoCompras` não é serializável. Isso acontece, porque o servidor ao verificar que um objeto está em escopo de

sessão, ele tenta salvar este objeto em arquivo. Para que ele possa sempre salvar a sessão e recuperar este objeto. Para resolvermos isto, basta fazer com que a classe `CarrinhoCompras` implemente a interface `Serializable` .

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

    private static final long serialVersionUID = 1L;

}
```

Perceba que a interface `Serializable` não adiciona nenhum método, mas sim um atributo estático. Agora podemos testar! Faça todos os passos de uma compra normal, visite a lista de produtos, adicione alguns ao carrinho de compras e clique em `Finalizar Compra` . Verifique se tudo está funcionando corretamente e prossiga para a próxima aula, onde continuaremos a desenvolver nosso carrinho de compras.