03

Mãos na massa: Expondo atributos na JSP

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

1) Ao abrir a página **detalhe.jsp**, todas as informações são exibidas corretamente, com exceção da data. Isso acontece porque não foi definido como ela deve ser apresentada. Use a *tag* **`formatDate`** para corrigir esse problema. Ela só aceita o tipo `Date`, então utilize o atributo `time` da data de lançamento:

```
<p>
    Data de publicação:
        <fmt:formatDate pattern="dd/MM/yyyy"
            value="${produto.dataLancamento.time}"/>
</p>
```

Não esqueça também de adicionar a `taglib`:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
```

2) Comece a criar o carrinho de compras! Todo carrinho de compras possui vários itens, e cada item é composto, além do próprio produto, do tipo que está sendo comprando (Ebook, Impresso, Combo). Portanto, crie a classe **`CarrinhoItem`** no pacote **`br.com.casadocodigo.loja.models`**, com os atributos **`produto`** e **`tipoPreco`**, além dos seus *getters* e *setters*. E já que para um `CarrinhoItem` existir, é preciso tanto do `produto` quanto o `tipoPreco`, peça-os por parâmetro no construtor:

```
public class CarrinhoItem {

    private Produto produto;
    private TipoPreco tipoPreco;

    public CarrinhoItem(Produto produto, TipoPreco tipoPreco) {
        this.produto = produto;
        this.tipoPreco = tipoPreco;
    }

    public Produto getProduto() {
        return produto;
    }

    public void setProduto(Produto produto) {
        this.produto = produto;
    }

    public TipoPreco getTipoPreco() {
        return tipoPreco;
    }

    public void setTipoPreco(TipoPreco tipoPreco) {
        this.tipoPreco = tipoPreco;
    }
```

```
}
```

3) Com a representação de um item do carrinho, crie o carrinho de fato. Para isso, crie a classe `CarrinhoCompras` no pacote `br.com.casadocodigo.loja.models`, com um atributo que possua um mapa que armazena o item e a quantidade no carrinho. Crie os métodos `add`, que adiciona o item no mapa, e `getQuantidade`, que retorna a quantidade de determinado item:

```
@Component
public class CarrinhoCompras {

    private Map<CarrinhoItem, Integer> itens = new LinkedHashMap<>();

    public void add(CarrinhoItem item) {
        itens.put(item, getQuantidade(item) + 1);
    }

    public Integer getQuantidade(CarrinhoItem item) {
        if(!itens.containsKey(item)) {
            itens.put(item, 0);
        }
        return itens.get(item);
    }

}
```

4) Para tudo funcionar corretamente, reescreva os métodos `hashCode` e `equals`, com os atributos `produto` e `tipoPreco`, da classe `CarrinhoItem`:

```
public class CarrinhoItem {

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((produto == null) ? 0 : produto.hashCode());
        result = prime * result + ((tipoPreco == null) ? 0 : tipoPreco.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 (produto == null) {
            if (other.produto != null)
                return false;
        } else if (!produto.equals(other.produto))
            return false;
```

```
            if (tipoPreco != other.tipoPreco)
                return false;
            return true;
        }

        // restante do código omitido
    }
```

5) Faça o mesmo para a classe `Produto`, mas somente com o atributo `id`:

```
    @Entity
    public class Produto {

        @Override
        public int hashCode() {
            final int prime = 31;
            int result = 1;
            result = prime * result + id;
            return result;
        }

        @Override
        public boolean equals(Object obj) {
            if (this == obj)
                return true;
            if (obj == null)
                return false;
            if (getClass() != obj.getClass())
                return false;
            Produto other = (Produto) obj;
            if (id != other.id)
                return false;
            return true;
        }

        // restante do código omitido
    }
```

6) Com as classes do carrinho prontas, é preciso acessá-lo. Crie o *controller* `CarrinhoComprasController` para o carrinho, no pacote `br.com.casadocodigo.loja.controllers`, e implemente os métodos `add`, que irá adicionar um item no carrinho, e `crieItem`, que irá criar um item para ser adicionado no carrinho. Não esqueça de utilizar as classes que foram criadas anteriormente. E para não se preocupar com a criação dos objetos, peça para o Spring injetá-los (`@Autowired`):

```
    @Controller
    @RequestMapping("/carrinho")
    public class CarrinhoComprasController {

        @Autowired
        private ProdutoDao produtoDao;

        @Autowired
        private CarrinhoCompras carrinho;
```

```
    @RequestMapping("/add")
    public ModelAndView add(Integer produtoId, TipoPreco tipoPreco) {

        ModelAndView modelAndView = new ModelAndView("redirect:/produtos");
        CarrinhoItem carrinhoItem = criaItem(produtoId,tipoPreco);

        carrinho.add(carrinhoItem);

        return modelAndView;
    }

    private CarrinhoItem criaItem(Integer produtoId, TipoPreco tipoPreco) {

        Produto produto = produtoDao.find(produtoId);
        CarrinhoItem carrinhoItem = new CarrinhoItem(produto, tipoPreco);

        return carrinhoItem;
    }
}
```

7) Para ver o carrinho funcionando, mostre a quantidade de itens de do lado do link **Carrinho**. Na classe `CarrinhoCompras`, crie o método `getQuantidade`, que conta os itens do carrinho:

```
public int getQuantidade() {
    return itens.values().stream()
        .reduce(0, (proximo, acumulador) -> proximo + acumulador);
}
```

E chame-o na JSP **detalhe.jsp**:

```
<ul class="clearfix">
    <li>
        <a href="#">
            Seu carrinho (${carrinhoCompras.quantidade})
        </a>
    </li>
    <li>
        <a href="/pages/sobre-a-casa-do-codigo" rel="nofollow">
            Sobre Nós
        </a>
    </li>
</ul>
```

8) Para enviar o carrinho de compras para a JSP, utilize o método `setExposedContextBeanNames`, da classe `InternalResourceViewResolver`, para deixar o carrinho disponível para todas as *views*. Além disso, para o Spring achar a classe **`CarrinhoCompras`**, adicione-a na anotação **`ComponentScan`**:

```
@EnableWebMvc
@ComponentScan(basePackageClasses={HomeController.class, ProdutoDao.class,
    FileSaver.class, CarrinhoCompras.class})
public class AppWebConfiguration {
```

```
    @Bean
    public InternalResourceViewResolver internalResourceViewResolver() {

        InternalResourceViewResolver resolver =
            new InternalResourceViewResolver();
        resolver.setPrefix("/WEB-INF/views/");
        resolver.setSuffix(".jsp");

        resolver.setExposedContextBeanNames("carrinhoCompras");

        return resolver;
    }

    // restante do código omitido
}
```