04

Mãos na massa: Utilizando o Bootstrap

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

1) No curso, é utilizada a versão **3.3.x** do Bootstrap. Você pode fazer o download dela aqui (https://s3.amazonaws.com/caelum-online-public/springmvc-2-integracao-cache-seguranca-e-templates/bootstrap-3.3.7-dist.zip). Mas caso queira uma versão mais atual, você pode baixá-la pelo site do Bootstrap (http://getbootstrap.com/getting-started/#download).

2) Extraia o conteúdo do Bootstrap na pasta **src/main/webapp/resources**. Diferentemente do video você já possui essa pasta. Não se esqueça que, para o Bootstrap funcionar, é preciso *linkar* os arquivos no `head` das suas páginas. Use o `c:url`, visto no curso anterior (Spring MVC I: Criando aplicações web), por exemplo:

```
<c:url value="/resources/css" var="cssPath" />
<link rel="stylesheet" href="${cssPath}/bootstrap.min.css">
<link rel="stylesheet" href="${cssPath}/bootstrap-theme.min.css">
```

3) Por padrão, o Spring pega qualquer requisição, inclusive o link de arquivos CSS e JavaScript. Para corrigir isso, faça a classe de configuração `AppWebConfiguration` herdar de `WebMvcConfigurerAdapter` e sobrescrever o método `configureDefaultServletHandling`:

```
@EnableWebMvc
@EnableCaching
@ComponentScan(basePackageClasses={HomeController.class, ProdutoDAO.class,
    FileSaver.class, CarrinhoCompras.class})
public class AppWebConfiguration extends WebMvcConfigurerAdapter {

    @Override
    public void configureDefaultServletHandling(
            DefaultServletHandlerConfigurer configurer) {
        configurer.enable();
    }

    // restante do código omitido
}
```

4) Com o **Bootstrap** configurado no projeto, comece a usar as classes do mesmo para estilizar as páginas da aplicação. Comece pela listagem dos produtos (**lista.jsp**), adicionando classes na tabela e trocando as *tags* de título do cabeçalho da tabela de `td` para `th`:

```
<table class="table table-bordered table-striped table-hover">
    <tr>
        <th>Título</th>
        <th>Descrição</th>
        <th>Páginas</th>
    </tr>
    <c:forEach items="${produtos}" var="produto">
        <!-- código omitido -->
```

```
    </c:forEach>
</table>
```

5) Envolver a listagem dos produtos com uma `div` que usa a classe `container` do *Bootstrap*. Ela centraliza os elementos na página, não deixando que tudo fique muito nos cantos da tela:

```
<body>
    <div class="container">
        <h1>Lista de Produtos</h1>
        <div>${sucesso}</div>
        <div>${falha}</div>
        <table class="table table-bordered table-striped table-hover">
            <tr>
                <th>Título</th>
                <th>Descrição</th>
                <th>Páginas</th>
            </tr>
            <c:forEach items="${produtos}" var="produto">
                <!-- código omitido -->
            </c:forEach>
        </table>
    </div>
<body>
```

6) Toda vez que você quiser ir de uma página para outra, tem que digitar a URL da página na barra de endereços do navegador. Crie um menu que facilite essa navegação entre as páginas, usando as classes do próprio *Bootstrap* (o código deve ficar acima da listagem, logo depois da *tag* `<body>`):

```
<nav class="navbar navbar-inverse">
    <div class="container">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed"
                data-toggle="collapse"
                    data-target="#bs-example-navbar-collapse-1"
                        aria-expanded="false">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="${s:mvcUrl('HC#index').build()}">
                Casa do Código
            </a>
        </div>
        <div class="collapse navbar-collapse"
                id="bs-example-navbar-collapse-1">
            <ul class="nav navbar-nav">
                <li>
                    <a href="${s:mvcUrl('PC#listar').build()}">
                        Lista de Produtos
                    </a>
                </li>
                <li>
                    <a href="${s:mvcUrl('PC#form').build()}">
```

```
                    Cadastro de Produtos
                </a>
            </li>
        </ul>
    </div><!-- /.navbar-collapse -->
  </div>
</nav>
```

7) Para o menu não ficar sobrepondo o título da página, adicione a *tag* `style` dentro da *tag* `head`, modificando a propriedade `padding-top` da *tag* `body`, adicionando uma margem de `60px`:

```
<style type="text/css">
    body {
        padding-top: 60px;
    }
</style>
```

8) Estilize também a página de cadastro de produtos. Copie todo o código do menu junto com a `div` que tem a classe `container` e cole no **form.jsp**, assim como o código CSS do `head` também, alterando o `padding` para não espaçar somente o topo da página, mas também sua parte inferior: :

```
<head>
    <meta charset="UTF-8">
    <title>Livros de Java, Android, iPhone, Ruby, PHP e muito mais - Casa do Código</title>
    <c:url value="/resources/css" var="cssPath" />
    <link rel="stylesheet" href="${cssPath}/bootstrap.min.css" />
    <link rel="stylesheet" href="${cssPath}/bootstrap-theme.min.css" />
    <style type="text/css">
        body {
            padding: 60px 0px;
        }
    </style>
</head>
<body>
    <nav class="navbar navbar-inverse">
        <!-- código omitido -->
    </nav>
    <div class="container">
        <form:form action="${s:mvcUrl('PC#gravar').build()}"
            method="POST" commandName="produto"
                enctype="multipart/form-data">
            <!-- código omitido -->
        </form:form>
    </div>
</body>
```

9) Todos os campos estão envolvidos por uma `div` e a maioria deles são criados com as tags do próprio Spring, como por exemplo `<form:input>`. Nestes casos, adicione a classe `form-group` em cada uma das `div`s e o atributo `cssClass` com o valor `form-control` em cada um dos `<form:input>`. Além disso, adicione a classe `form-control` no `input` do sumário e as classes `btn btn-primary` no botão de submissão do formulário. E como o Bootstrap já estiliza bem os campos do formulário, remova os atributos `cols` e `rows` do campo `<form:textarea>`, adicionando também um título antes do formulário:

```
<h1>Cadastro de Produto</h1>
<form:form action="${s:mvcUrl('PC#gravar').build()}" method="POST"
        commandName="produto" enctype="multipart/form-data">
    <div class="form-group">
        <label>Título</label>
        <form:input path="titulo" cssClass="form-control" />
        <form:errors path="titulo" />
    </div>
    <div class="form-group">
        <label>Descrição</label>
        <form:textarea path="descricao"    cssClass="form-control" />
        <form:errors path="descricao" />
    </div>
    <div class="form-group">
        <label>Páginas</label>
        <form:input path="paginas" />
        <form:errors path="paginas" />
    </div>
    <div class="form-group">
        <label>Data de Lançamento</label>
        <form:input path="dataLancamento" cssClass="form-control" />
        <form:errors path="dataLancamento" />
    </div>
    <c:forEach items="${tipos}" var="tipoPreco" varStatus="status">
        <div class="form-group">
            <label>${tipoPreco}</label>
            <form:input path="precos[${status.index}].valor"
                cssClass="form-control" />
            <form:hidden path="precos[${status.index}].tipo"
                value="${tipoPreco}" />
        </div>
    </c:forEach>
    <div class="form-group">
        <label>Sumário</label>
        <input name="sumario" type="file" class="form-control" />
    </div>
    <button type="submit" class="btn btn-primary">Cadastrar</button>
</form:form>
```