

Importando Bootstrap

Transcrição

O que faremos nesta aula é melhorar um pouco o estilo de algumas páginas da nossa aplicação. Perceba que o foco até aqui foi fazer com que as funcionalidades do sistema estivessem prontas e por isso, deixamos as páginas todas sem estilo. Veja por exemplo como estão as páginas de listagem de produtos e o formulário de cadastro.

Listagem de produtos:

Título	Descrição	Páginas
TDD no Mundo Real	Por que não testamos software? Porque é caro? Porque é demorado? Porque é chato?	230
Android	Utilize o framework do google	210
Java 8 Prático	Java 8 na prática	250
Java 8 Prático com sumário	Java 8 Prático com sumário	50
Java 8 Prático com sumário	Java 8 Prático com sumário	250
Java 8 Prático: Lambdas, Streams e os novos recursos da linguagem	Você é um programador com boa fluência em Java? Está na hora de dar mais um passo. Quase 20 anos após sua primeira versão, um novo Java surge com novidades importantes. Entre os principais recursos, a linguagem recebe default methods, method references e lambdas. São conceitos simples, mas que trazem importantes possibilidades. Durante o livro, exploraremos esses e outros avanços. Sempre com exemplos práticos e apresentando onde utilizá-los, migrando o código legado do dia a dia para o novo paradigma funcional do Java 8.	151
Novo Livro	livro de testes	215
Novo Livro 2	Testando o cache pela segunda vez	350

Formulário de cadastro:

Para melhorar estas páginas usaremos um **framework front end** bem conhecido, o **Bootstrap** (<http://getbootstrap.com/>). Você poderá baixá-lo [aqui](http://getbootstrap.com/getting-started/#download) (<http://getbootstrap.com/getting-started/#download>) e depois, descompactar o arquivo. Verifique a existência de **três** pastas dentro de outra, chamada **Bootstrap**, sendo elas: **css**, **js** e **fonts**.

Para adicionar o **Bootstrap** ao nosso projeto, seguiremos o seguinte padrão. Criaremos uma pasta dentro de `webapp` chamada de `resources`, na qual guardaremos as pastas `CSS`, `JS` e `Fonts` do **Bootstrap**. Agora podemos abrir a página `lista.jsp` dentro de `webapp/WEB-INF/views/produtos` e adicionar as linhas de importe dos arquivos de `css` e `javascript` do **Bootstrap** da seguinte forma:

```
<link rel="stylesheet" href="resources/css/bootstrap.min.css" />
<link rel="stylesheet" href="resources/css/bootstrap-theme.min.css" />
<script src="resources/js/bootstrap.min.js"></script>
```

Assim poderemos reiniciar o servidor para que a aplicação carregue estes arquivos e após o reinício, abriremos a página de listagem de produtos. Mas algo estranho acontece:

Título	Descrição	Páginas
TDD no Mundo Real	Por que não testamos software? Porque é caro? Porque é demorado? Porque é chato?	230
Android	Utilize o framework do google	210
Java 8 Prático	Java 8 na prática	250
Java 8 Prático com sumário	Java 8 Prático com sumário	50
Java 8 Prático com sumário	Java 8 Prático com sumário	250
Java 8 Prático: Lambdas, Streams e os novos recursos da linguagem	Você é um programador com boa fluência em Java? Está na hora de dar mais um passo. Quase 20 anos após sua primeira versão, um novo Java surge com novidades importantes. Entre os principais recursos, a linguagem recebe default methods, method references e lambdas. São conceitos simples, mas que trazem importantes possibilidades. Durante o livro, exploraremos esses e outros avanços. Sempre com exemplos práticos e apresentando onde utilizá-los, migrando o código legado do dia a dia para o novo paradigma funcional do Java 8.	151
Novo Livro	livro de testes	215
Novo Livro 2	Testando o cache pela segunda vez	350

Nada mudou na listagem, a página continua a mesma. Experimente abrir as ferramentas do desenvolvedor do seu navegador e verificar a aba console, verá algo semelhante a:



Note que apesar dos caminhos para os arquivos estarem corretos, o *Spring* não os carrega corretamente, porque não está conseguindo encontrar os arquivos. Isto acontece porque o **Servlet** do *Spring* está processando todas as requisições e procurando cada um dos recursos requisitados dentro dele. Desta forma o *Spring* está procurando arquivos *JPS*s com os mesmos nomes dos arquivos de estilo e script que adicionamos antes.

Para resolvermos o problema, precisamos dizer para o *Spring* que as requisições referentes a arquivos de script, estilo, fontes e imagens não devem ser processadas pelo *servlet* do *Spring*, mas sim pelo *servlet default* da aplicação.

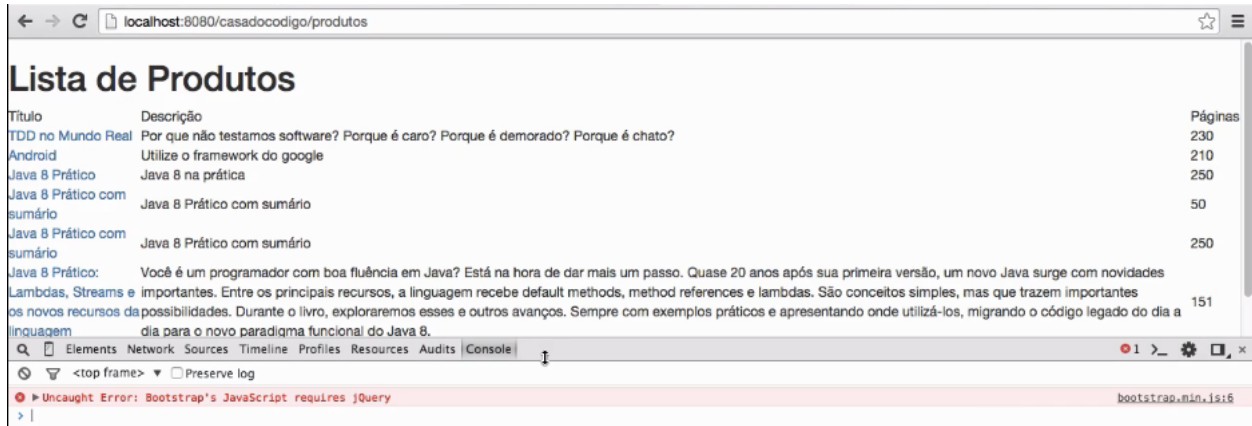
Vamos fazer esta configuração na classe `WebAppConfiguration`. O primeiro passo é fazer com que esta classe agora herde da classe `WebMvcConfigurationAdapter`. Pois dessa forma, herdaremos uma série de métodos que nós permitem fazer algumas configurações de forma mais prática.

```
public class AppWebConfiguration extends WebMvcConfigurerAdapter{
    [...]
}
```

Após isso, no final da classe, criaremos um novo método chamado `configureDefaultServletHandling` que irá configurar o *servlet* padrão para que este atenda as requisições de arquivos como *css* e *js*. Este método recebe um parâmetro do tipo `DefaultServletHandlerConfigurer` que chamaremos apenas de `configurer` e usaremos o método `enable()` deste objeto para habilitar o *servlet* padrão do servidor de aplicação.

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

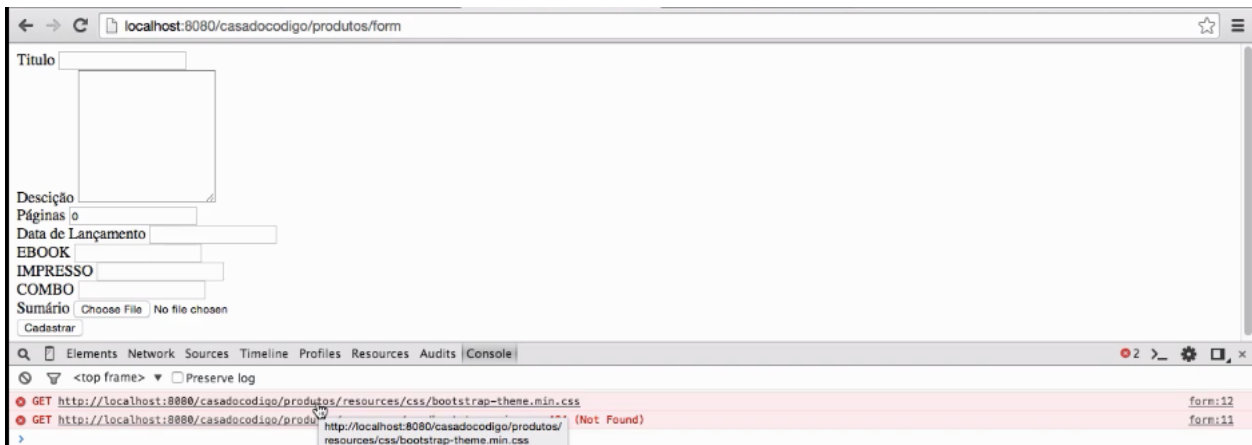
Se reiniciarmos o servidor neste momento, já veremos diferenças na página de listagem dos produtos. Perceba que a cor dos links mudaram assim como a fonte usada na página. Note que os erros no console do navegador também sumiram e um novo aparece desta vez:



Os scripts do *Bootstrap* requerem o **jQuery** para funcionar, mas como não os estamos usando, não há problema nenhum. Podemos inclusive, remover a *tag* de *import* do script do *Bootstrap*. No arquivo `lista.jsp` teremos o seguinte:

```
[...]
<head>
<meta charset="UTF-8">
<title>Livros de Java, Android, iPhone, Ruby, PHP e muito mais - Casa do Código</title>
<link rel="stylesheet" href="resources/css/bootstrap.min.css" />
<link rel="stylesheet" href="resources/css/bootstrap-theme.min.css" />
</head>
[...]
```

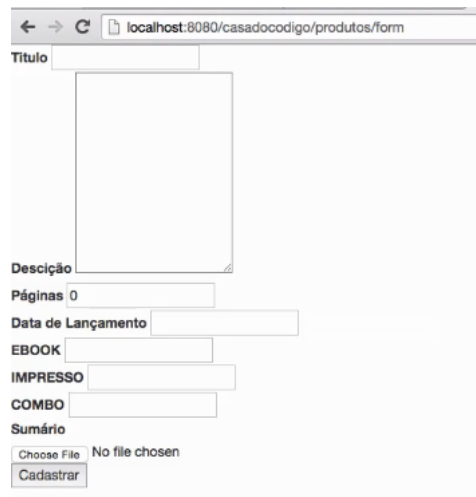
Vamos copiar estas linhas que importam os estilos do *Bootstrap* para a página `form.jsp` e ver o que acontecerá. No console do navegador, teremos um erro. Perceba que o caminho para os arquivos está estranho.



O caminho dos arquivos deveriam estar dessa forma: `casadocodigo/resources/`. No entanto, Os mesmos estão desta forma `casadocodigo/produtos/resources`. O problema é que estamos passando um caminho relativo à página. Para resolvermos isso, podemos usar a tag `c:url` para definir uma variável apontando para o caminho da pasta dos arquivos de css e usar esta variável na tag `link` do HTML da seguinte forma:

```
<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" />
</head>
```

Assim que atualizarmos a página `form.jsp`, já veremos que a solução funcionou e o formulário parece estar um pouco diferente. Ainda vamos estilizá-lo um pouco mais, mas já é notável a diferença.



The screenshot shows a web browser window with the address bar displaying `localhost:8080/casadocodigo/produtos/form`. The form contains the following elements:

- Titulo**: A text input field.
- Descrição**: A large text area.
- Páginas**: A text input field with the value `0`.
- Data de Lançamento**: A date input field.
- EBOOK**: A text input field.
- IMPRESSO**: A text input field.
- COMBO**: A text input field.
- Sumário**: A text input field.
- Choose File**: A button next to the text `No file chosen`.
- Cadastrar**: A button at the bottom of the form.

Lembre-se de copiar essas linhas de importe do css com uso da variável do arquivo `form.jsp` para o arquivo `lista.jsp`. Assim manteremos um padrão que sempre funciona em todas as páginas.