03

Ajustando a home

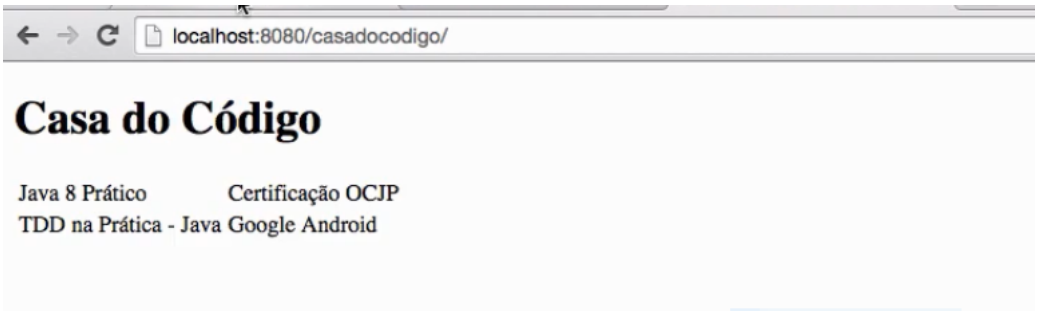
Transcrição

Melhorando a home

Bem-vindo ao segundo módulo do curso de **Spring MVC**. Neste curso continuaremos a desenvolver a aplicação da **Casa do Código** que iniciamos no primeiro módulo.

Se abrirmos a página inicial da nossa aplicação, veremos que ela não tem o nome do projeto e uma lista de livros em texto puro, sem nenhum estilo. Queremos estilizar esta página, até que ela fique mais próxima do estilo do site da Casa do Código [Casa do Código (http://www.casadocodigo.com.br)](http://www.casadocodigo.com.br).

Veja como está nossa página inicial, atualmente:



Queremos deixá-la assim:

O primeiro passo será copiar o HTML da página, que poderá ser baixado em: [https://s3.amazonaws.com/caelum-online-public/springmvc-2-integracao-cache-seguranca-e-templates/home-casadocodigo-antiga.zip (https://s3.amazonaws.com/caelum-online-public/springmvc-2-integracao-cache-seguranca-e-templates/home-casadocodigo-antiga.zip)](https://s3.amazonaws.com/caelum-online-public/springmvc-2-integracao-cache-seguranca-e-templates/home-casadocodigo-antiga.zip)

Baixe o arquivo e descompacte, verifique se o arquivo `home.jsp` se encontra no *zip*. Depois, apague o arquivo `home.jsp` que está no seu projeto atualmente. Este arquivo se encontra em: `webapp/WEB-INF/views/` . Após apagar o arquivo, copie o novo arquivo que acabou de baixar para esta mesma pasta. Feito isso, podemos continuar.

Dentro do arquivo `home.jsp` , precisamos encontrar o trecho de código responsável por exibir cada um dos livros e fazer com que esse trecho exiba os livros que temos em nosso banco de dados. Para isto, procure o seguinte trecho de código:

```
<li>
    <a href="URL para o livro" class="block clearfix">
        <h2 class="product-title">Java 8 Prático</h2>
        <img width="143"
            height="202"
            src="https://cdn.shopify.com/s/files/1/0155/7645/products/java8-featured_large.png?
            alt="Java 8 Prático"
            title="Java 8 Prático"/>
        <small class="buy-button">Compre</small>
    </a>
</li>
```

Note que a exibição dos livros é feita dentro de uma simples lista, e cada item `<li>` corresponde a um livro. Para que exibamos os livros então, precisaremos fazer um laço que percorra toda a nossa coleção de produtos e mostre suas informações.

Para isto, faremos um `forEach` na variável `produtos`. Como está é uma variável temporária no atributo `var` do `forEach`, usaremos outra variável: `produto`. Lembre-se que este laço deve envolver o item de lista inteiro da seguinte forma:

```
<c:forEach items="${produtos}" var="produto">

<li>
  <a href="URL para o livro" class="block clearfix">
          <h2 class="product-title">Java 8 Prático</h2>
          <img width="143"
              height="202"
              src="https://cdn.shopify.com/s/files/1/0155/7645/products/java8-featured_large.pn
              alt="Java 8 Prático"
              title="Java 8 Prático"/>
          <small class="buy-button">Compre</small>
  </a>
</li>

</c:forEach>
```

Agora precisamos exibir o nome do livro e construir a *URL* que quando clicada deverá levar o usuário à página de detalhes do livro. Neste mesmo bloco, montaremos a *URL* para a página de detalhes do livro onde há:

```
<a href="URL para o livro" class="block clearfix">
```

Faremos da seguinte forma:

```
<a href="${s:mvcUrl('PC#detalhe').arg(0, produto.id).build()}" class="block clearfix">
```

Estamos usando o `mvcUrl` do *Spring* para criarmos uma *URL* que será atendida pelo método `detalhe` da classe `ProdutosController` e passando para este método o `id` do livro como parâametro e no fim, pedimos para o *Spring* construir a *URL* através do método `build()`.

A próxima informação a ser exibida será o título do livro, localizada na seguinte linha do código:

```
<h2 class="product-title">Java 8 Prático</h2>
```

Este passo é bem simples, basta trocar o título que está no texto por um trecho de código **Java**:

```
<h2 class="product-title">${produto.titulo}</h2>
```

O código responsável por exibir a listagem dos livros após todas as modificações ficará da seguinte forma:

```
<c:forEach items="${produtos}" var="produto">

    <li>
    <a href="<s:mvcUrl('PC#detalhe').arg(0, produto.id).build()" class="block clearfix">
            <h2 class="product-title">${produto.titulo}</h2>
            <img width="143"
                height="202"
                src="https://cdn.shopify.com/s/files/1/0155/7645/products/java8-featured_large.
                alt="Java 8 Prático"
                title="Java 8 Prático"/>
            <small class="buy-button">Compre</small>
    </a>
    </li>

</c:forEach>
```

Nosso trabalho com a *view* termina aqui. Agora precisamos atualizar o `HomeController` para que ele passe para o `home.jsp`, a lista de livros que estamos querendo exibir. Veja como está o `HomeController` atualmente:

```
@Controller
public class HomeController {

    @RequestMapping("/")
    public String index(){
        System.out.println("Entrando na home da CDC");
        return "home";
    }

}
```

Basicamente, sua função é imprimir uma mensagem no console e retornar a *view*. Agora, o que precisamos fazer é usar o `ProdutoDao` para buscar os produtos no banco de dados e mandar a coleção de itens para a *view* através do `ModelAndView`.

Crie o atributo `produtoDao` do tipo `ProdutoDao`, em seguida, marque-o com a anotação `@Autowired`. Após isto, dentro do método `index` use o método `listar` do `produtoDao` e atribua o resultado a uma lista que chamaremos de `produtos`. Crie um objeto `modelAndView` do tipo `ModelAndView` que se direcionará para a *view* `home.jsp` e use o método `addObject` para anexar a listagem dos produtos. Como último passo, retorne o objeto `modelAndView`. Desta forma teremos:

```
@Controller
public class HomeController {

    @Autowired
    ProdutoDAO produtoDao;

    @RequestMapping("/")
    public ModelAndView index(){
        List<Produto> produtos = produtoDao.listar();
        ModelAndView modelAndView = new ModelAndView("home");
        modelAndView.addObject("produtos", produtos);
        return modelAndView;
    }

}
```

Perceba que o retorno do método `index` mudou em sua assinatura, fizemos a alteração porque não estamos mais retornando o nome da *view*, mas sim, um outro objeto que além de saber qual **view** carregar, também leva dados importantes para serem exibidos.

Observe também que as imagens de miniatura dos livros serão as mesmas para todos, infelizmente não criamos uma lógica para permitir que cada produto tenha sua miniatura de foto. Fica o desafio para que você faça...

Com a atualização, já podemos verificar como ficou a página inicial da aplicação remodelada.