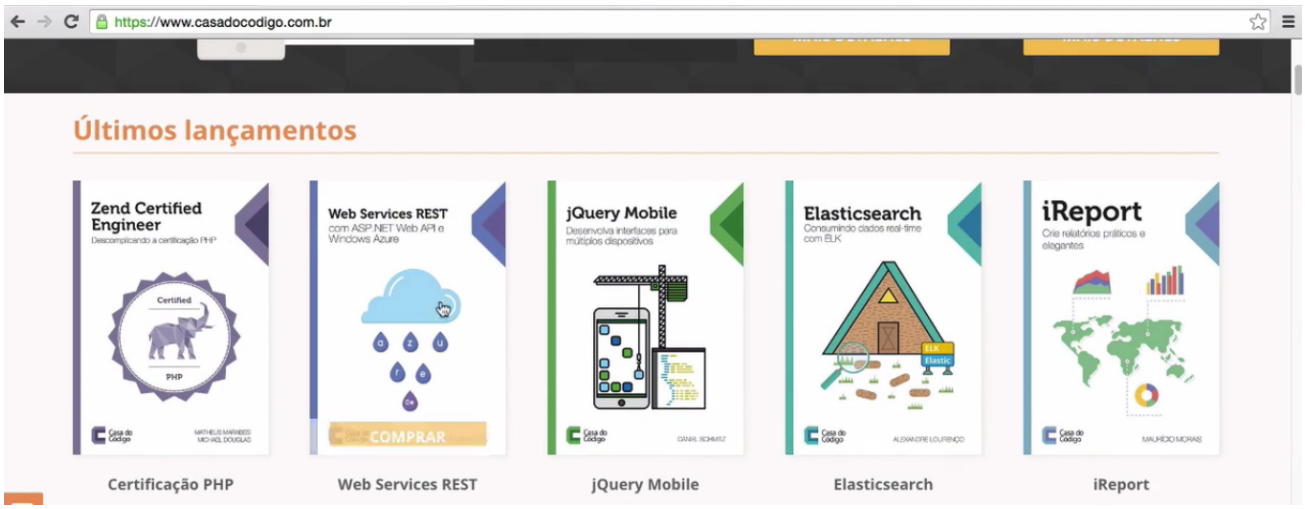
01

Adicionando a Home Page da CDC

Transcrição

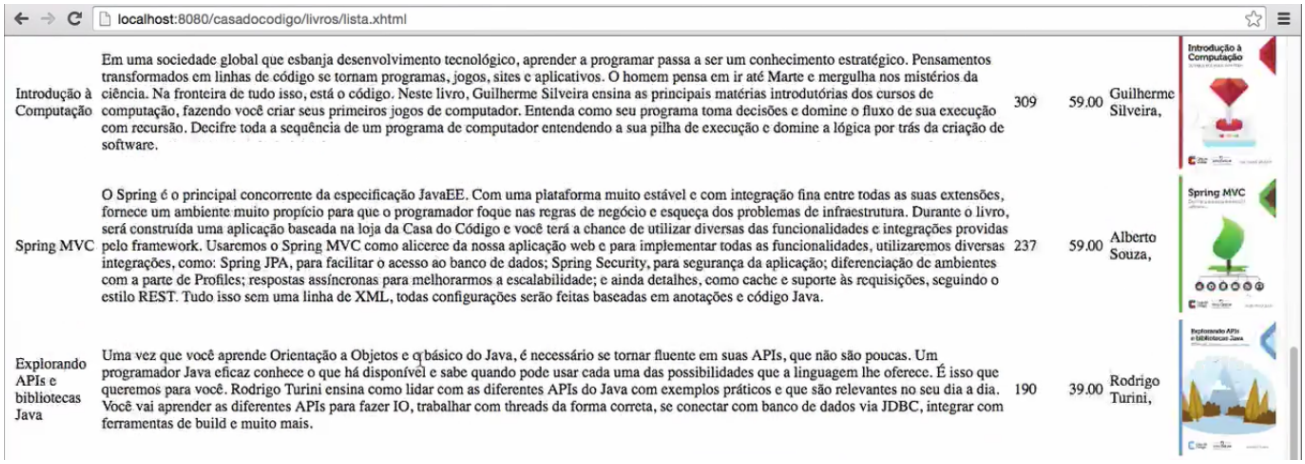
Nosso sistema administrativo e o formulário já estão muito bons. Então vamos tentar fazer com que nossa HomePage comece a ser exibida, e tenhamos uma página inicial bem atrativa.



Faremos a exibição dos livros em destaque separando-os dos demais livros, assim como é feito na loja da Casa do Código. Queremos a mesma home, porém não iremos criar tudo novamente, pois seria muito trabalhoso. Você pode baixar o pacote com a home da *CDC* no link abaixo:

https://s3.amazonaws.com/caelum-online-public/java-ee-webapp/casadocodigo-javaee-home.zip (https://s3.amazonaws.com/caelum-online-public/java-ee-webapp/casadocodigo-javaee-home.zip)

Você já encontrará cerca de 15 livros cadastrados no pacote, com as capas, descrição e nome dos autores.

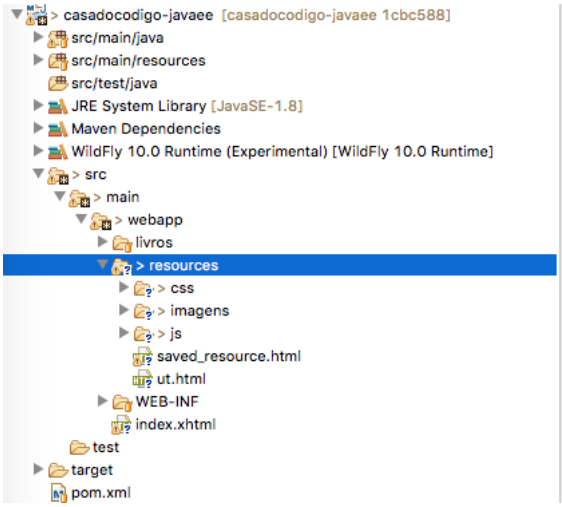


Após baixar o arquivo, delete o atual arquivo `index.html` que temos na nossa aplicação e vamos importar um novo arquivo com uma estrutura já preparada para o nosso projeto.

Ainda no Eclipse, clique com o botão direito sobre o projeto e selecione a opção `Import`. Vá em `General` e depois em `Archive File` e depois vá em `Next`. Nessa tela, cliquem em `Browse...` que fica logo a frente do campo `From archive file:` e navegue até o local onde você baixou o `zip` da home da CDC, clique em OK. Você deve estar vendo uma

estrutura de pastas no lado esquerdo e no lado direito o arquivo `index.xhtml` . Antes de finalizar, na caixa `Into folder` procure pelo caminho `src/main/webapp` e depois clique `OK` . Por fim, clique em `Finish` para finalizar o processo.

Você já deve estar vendo os arquivos importados.



Vamos testar se nosso `index.xhtml` já está funcionando. *Full Publish* do projeto e acesse: http://localhost:8080/casadocodigo/index.xhtml (http://localhost:8080/casadocodigo/index.xhtml). Navegue um pouco na aplicação para brincar, mas lembre-se que alguns links estão enviando para a loja oficial da Casa do Código.

Como todos os livros estão com a mesma capa, percebemos que ainda não está sendo considerada os livros cadastrados no banco de dados. Vamos fazer com que nossos livros cadastrados apareçam de fato na Home.

Abra o `index.xhtml` e vá para a linha 217. A partir da segunda tag `<li>` do arquivo, selecione todos os demais `<li>` até antes do fechamento da tag `</ul>` de modo que sobre ao menos 1 tag `<li>` para servir de repetição do nosso `for` .

Logo acima do `<li>` vamos colocar uma tag `<ui:repeat value="" var="livro">` . Já colocamos o nome do `var` como sendo **livro** e vamos realizar algumas modificações no meio do código.

Dentro da tag `<a>` procure pelo atributo `title` e modifique para `title="#{livro.titulo}"` . No atributo `<img>` procure pelos atributos `alt` e `title` e coloque os valores `alt="#{livro.titulo}" title="#{livro.titulo}` e dentro da tag `<span>` também modifique o título do livro atual para `#{livro.titulo}` .

Como estamos trabalhando na parte de últimos lançamentos da Home, vamos criar um `ManagedBean` que será responsável por trazer os 5 livros que fazem parte dos ultimos lançamentos. Chamando esse `Bean` de `HomeBean` e podemos ter um método chamado `ultimosLancamentos` . Antes de criar efetivamente nosso `Bean` , podemos já adicionar essa informação ao nosso `ui:repeat` . Nosso código de repetição dos últimos lançamentos ficará assim:

```
<ui:repeat value="#{homeBean.ultimosLancamentos()}" var="livro">
    <li class="livroNaVitrine vitrineDestaquinho-produto">
        <a href="https://www.casadocodigo.com.br/products/livro-certificacao" class="livroNaVitı
                title="#{livro.titulo}">
            <div class="livroNaVitrine-imagemContainer" role="presentation">
                <img class="livroNaVitrine-imagem"
                    src="#{request.contextPath}/resources/imagens/certificacao-java-featured_la
                        alt="#{livro.titulo}" title="#{livro.titulo}" />
            </div>
            <span class="livroNaVitrine-nome">#{livro.titulo}</span>
        </a>
```

```
    </li>
</ui:repeat>
```

Já referenciamos o `HomeBean`, então vamos criá-lo. Após usarmos o comando `Ctrl + N`, selecionaremos `Class` e daremos o nome da classe de `HomeBean`. Depois, colocaremos a classe no pacote de `beans`. Anote a classe com `@Model` como já fizemos anteriormente e também crie um método chamado `ultimosLancamentos` que retorna um `List<Livro>`. Dentro do nosso método vamos precisar de um `LivroDao`, então já vamos *injetar* o `LivroDao` no nosso Bean e chamamos dentro do nosso método `dao.ultimosLancamentos()`. Nosso `HomeBean` até o momento ficará assim:

```
package br.com.casadocodigo.loja.beans;

import javax.enterprise.inject.Model;

@Model
public class HomeBean {

    @Inject
    private LivroDao dao;

    public List<Livro> ultimosLancamentos() {
        return dao.ultimosLancamentos();
    }

}
```

Como o `LivroDao` ainda não possui o método `ultimosLancamentos`, vamos criá-lo pressionando `Ctrl + 1` e escolhendo a opção `"Create method ..."`. O Eclipse deve ter criado para você o método, já com a assinatura tudo certinho.

Dentro do método criado, vamos colocar um **JPQL** para obter os livros do banco de dados. Além disso, como queremos apenas os últimos 5 resultados, vamos pedir para o `EntityManager` realizar essa limitação, utilizando o `setMaxResults`. O método `ultimosLancamentos()` completo ficará assim:

```
public List<Livro> ultimosLancamentos() {
    String jpql = "select l from Livro l order by l.id desc";
    return manager.createQuery(jpql, Livro.class)
            .setMaxResults(5)
            .getResultList();
}
```

Observe que o `order by l.id desc`, com o qual estamos buscando os livros ordenados inversamente pelo `ID` do `Livro`. Assim garantiremos que os últimos cadastros serão retornados pela *query*. Poderíamos usar a data de lançamento para isso, mas como no nosso caso não fará diferença, sinta-se à vontade pra usar qualquer um dos dois.

Vamos testar nossa aplicação. Faremos um *Full Publish* e veremos se nova *Home* está aparecendo corretamente e listando os últimos cinco livros.

Voltando para a `index.xhtml`, vamos realizar uma alteração para os demais livros. Dentro do `vitrineDaColecao` (próximo da linha `238`), removeremos todos os `<li>` que estiverem dentro de `<ul>` mantendo apenas 1 que servirá para repetirmos. Logo acima do `<li>` que sobrou, adicione a tag `<ui:repeat>` da mesma forma que acima, mudando apenas o `value` para:

```
value="#{homeBean.demaisLivros()}"
```

Nosso código final ficará assim:

```
<ul class="vitrineDaColecao-lista">
    <ui:repeat value="#{homeBean.demaisLivros()}" var="livro">
        <li class="livroNaVitrine vitrineDaColecao-produto">
            <a href="https://www.casadocodigo.com.br/products/livro-certificacao" class="livroNa
                title="#{livro.titulo}">
                <div class="livroNaVitrine-imagemContainer" role="presentation">
                    <img class="livroNaVitrine-imagem"
                        src="#{request.contextPath}/resources/imagens/certificacao-java-feature
                        alt="#{livro.titulo}" title="#{livro.titulo}" />
                </div>
                <span class="livroNaVitrine-nome">#{livro.titulo}</span>
            </a>
        </li>
    </ui:repeat>
</ul>
```

Esse novo método no `Bean` precisa ser criado. Vamos criar um novo método na classe `HomeBean` chamado `demaisLivros()` retornando `List<Livro>` também. Dentro dele, apenas a chamada para `return dao.demaisLivros();` que ainda não existe no `LivroDao`. Seguiremos o mesmo esquema de antes, `Ctrl + 1` e escolha a opção `Create method ...`. Já dentro do método na classe `LivroDao`, criaremos a **JPQL** que cuidará do carregamento dos demais livros, e será exatamente igual a anterior. Mas se ela é igual a anterior, o que mudará dos demais livros para o últimos livros?

Nós limitamos os últimos lançamentos aos cinco primeiros resultados. Se já temos os cinco, não queremos que eles se repitam nos demais livros, certo? Então, vamos mudar os parâmetros do `EntityManager`. Vamos retirar a limitação dos últimos resultados, pois agora queremos todos. Mas queremos todos **sem os cinco primeiros**, e para isso vamos informar a propriedade `setFirstResult(5)`, que é onde dizemos que os resultados considerados precisam começar a partir do sexto item encontrado, apesar de informarmos o número `5`, pois a indexação começa do `0` (zero). E como já temos os cinco primeiros, realmente queremos apenas do sexto até o último.

```
public class LivroDao {

    // Demais métodos acima

    public List<Livro> demaisLivros() {
        String jpql = "select l from Livro l order by l.id desc";
        return manager.createQuery(jpql, Livro.class)
                .getResultList();
    }

}
```

Faça novamente um *Full Publish* e vamos entrar na nossa aplicação pelo navegador. Tudo deve estar funcionando corretamente.

Nossa aplicação está bem elegante, mas ainda queremos que as capas dos livros exibidas sejam de fato as capas que cadastramos. Para essa simples alteração, basta abrir o `index.xhtml` e dentro do `<li>` procurar pela tag `<img>`. Perceba que essa tag está apontando para algum caminho interno como `resources/...`. O que temos que fazer é alterar para *invocar* o nosso `FileServlet` que já criamos na aula anterior. Para isso, altere o atributo `src` para `src="#{request.contextPath}/file/#{livro.capaPath}"`. A tag `<img>` completa ficará assim:

```
<img class="livroNaVitrine-imagem"
        src="#{request.contextPath}/file/#{livro.capaPath}"
        alt="#{livro.titulo}" title="#{livro.titulo}" />
```

Lembre-se de alterar para os **Últimos Lançamentos** bem como para os **Demais Livros**.

Faça novamente um *Full Publish* do projeto e ao subir o servidor, navegue pela **Home** da aplicação. Todas as capas devem estar corretas e os títulos também correspondentes. Assim, nossa `Home` está praticamente idêntica a da **CDC** e agora temos uma página bem elegante.