Desenvolvimento Web I

Aula 13 - Desenvolvendo uma livraria virtual – Parte 2

Apresentação

Na aula anterior, você implementou a modelagem das classes de negócio do sistema de livraria online. Nesta aula, você continuará a desenvolver a modelagem desse sistema, adicionando mais recursos como por exemplo a sua parte Web, permitindo que o usuário visualize os dados do seu sistema. Você estudará o desenvolvimento de um Servlet controlador e das páginas de apresentação JSP da livraria virtual. Continuaremos com as atividades relacionadas à locadora virtual de filmes, então, você será o responsável por colocar a locadora funcionando na Web! Boa leitura!

Objetivos

- Entender o desenvolvimento da camada Web de sistemas, como o da livraria virtual, o que inclui o desenvolvimento de Servlets controladores e de arquivos JSP de apresentação do conteúdo.

Contexto dos Servlets

Na aula anterior, você estudou a modelagem de diversas classes, como a classe Livraria, responsável por implementar as funcionalidades da livraria: manutenção do estoque e venda dos livros. Para que essa classe e suas operações fiquem disponíveis para a camada de apresentação Web (Servlets e arquivos JSP), é necessário disponibilizar o sistema de alguma forma. Como já vimos antes, vamos disponibilizar o sistema por meio do contexto dos Servlets. Isso é feito pela classe LivrariaContextListener, mostrada na Listagem 1. Um ContextListener pode ser adicionado através do menu New... -> Listener ou diretamente através do menu New... -> Class adicionando a referência à interface ServletContextListener após a criação da classe.

```
1  package livraria.servlet;
2
3  import javax.servlet.ServletContext;
4  import javax.servlet.ServletContextEvent;
5  import javax.servlet.ServletContextListener;
6  import javax.servlet.annotation.WebListener;
7
8  import livraria.negocio.Livraria;
9
10 @WebListener
11 public final class LivrariaContextListener implements ServletContextListener {
12
13    public static final String SISTEMA_LIVRARIA = "sistemaLivraria";
14
15    public void contextInitialized(ServletContextEvent event) {
16       ServletContext context = event.getServletContext();
17
18       try {
19          Livraria livraria = new Livraria();
20          context.setAttribute(SISTEMA_LIVRARIA, livraria);
21       } catch (Exception ex) {
22          System.out.println(
23          "O sistema de livraria não pode ser publicado no contexto: "
24          + ex.getMessage());
25       }
26    }
27
28    public void contextDestroyed(ServletContextEvent event) {
29       ServletContext context = event.getServletContext();
30
31       Livraria livraria = (Livraria) context.getAttribute(SISTEMA_LIVRARIA);
32
33       if (livraria != null) {
34          livraria.fechar();
35       }
36
37       context.removeAttribute(SISTEMA_LIVRARIA);
38    }
39 }
```

Listagem 1 - Classe que publica o sistema da livraria online através do contexto dos Servlets

Como você pode notar, temos o método contextInitialized() (linha 13), que será executado no evento de inicialização do contexto dos Servlets. Esse método é executado apenas uma vez no servidor e ele instancia a classe Livraria, criando, por meio da linha 17, o núcleo da nossa Livraria virtual, o qual é publicado através de um atributo de contexto de nome “sistemaLivraria”, conforme instruções mostradas pelas linhas 11 e 18. O uso da constante SISTEMA_LIVRARIA evita que o texto “sistemaLivraria” seja escrito em vários lugares (sujeito a erros).

Dica

Utilize a opção New / Listener do Eclipse ao invés de New / Class para criar novos Listeners. Assim, a configuração deles no arquivo web.xml (ou, se for o caso, a anotação de @WebListener na classe criada) será feita automaticamente.

Observação: na versão 3.0 do Servlet, é adicionada a anotação @WebListener que remove a necessidade de criação do mapeamento do listener no web.xml. A anotação já define o weblistener e o mapeamento de acordo com o nome do arquivo java criado.

Atividade 01

1. Implemente o LivrariaContextListener conforme apresentado. Crie um Listener de exemplo que, ao ser executado, acessa o contexto e escreve uma página de resposta indicando se o atributo do contexto de nome sistemaLivraria existe (não é nulo).
2. Implemente um Listener de contexto para o sistema de locadora virtual desenvolvido como atividade na aula anterior. Faça um teste de acordo com o teste realizado para o LivrariaContextListener.

Servlet controlador da livraria virtual

Seguindo o padrão MVC (Model-View-Controler), utilizaremos um Servlet como componente controlador e arquivos JSP como os componentes responsáveis por montar as páginas de resposta ao usuário.

O Servlet controlador da livraria virtual, de nome ServletControladorLivraria, é mostrado na Listagem 2. Esse Servlet implementa o método doGet() (linha 16) e utiliza o método doPost() (linha 43) apenas para redirecionar a requisição ao método doGet().

Na execução do Servlet controlador, é feito o acesso ao sistema da livraria virtual (linha 20) por meio do atributo de contexto, conforme demonstrado anteriormente. Também é feito um acesso à sessão do usuário (linha 22). Lembre-se de que o método getSession() cria uma sessão, caso ela não exista. Em seguida, nas linhas 24 a 28, o carrinho de compras é acessado ou criado e armazenado como atributo da sessão de nome “cart”, caso ele ainda não exista.

Por fim, as linhas 30 a 40 são responsáveis por processar e montar a resposta para as requisições dos clientes. Para facilitar esse trabalho, utilizaremos a seguinte estratégia:

- As URLs de acesso ao sistema não possuirão extensão. Isso pode ser visto no seguinte exemplo http://localhost:8080/Livraria/livros/catalogo.

- O caminho da URL será utilizado para descobrir qual o serviço a ser executado. Isso é feito na linha 30, onde o método getServletPath() da variável request é utilizado para recuperar parte da URL de acesso. No caso, a parte relativa ao caminho do Servlet: /livros/catalogo. Essa informação é armazenada na variável acaoSelecionada.

- Dependendo da ação a ser executada (conteúdo da variável acaoSelecionada), uma determinada operação do sistema pode ser executada. Essas operações, no caso, serão apresentadas aos poucos e seu código será inserido logo após a linha 32.

O conteúdo da variável acaoSelecionada é utilizado, também, para definir o componente (arquivo JSP) responsável por montar a página de resposta. Isso é feito pela linha 34, onde usamos a convenção do nome e pasta onde se encontra o arquivo JSP de resposta igual ao conteúdo da variável acaoSelecionada (URL do Servlet) concatenada com a extensão .jsp.

Dessa forma, o acesso à URL http://localhost:8080/Livraria/livros/catalogo poderá resultar na execução de uma operação do sistema identificada pelo final da URL (/livros/catalogo) e sua resposta será montada de acordo com o arquivo catalogo.jsp encontrado na pasta de nome livros.

```
1  package livraria.servlet;
2
3  import javax.servlet.annotation.WebServlet;
4  import javax.servlet.http.HttpServlet;
5  import javax.servlet.http.HttpServletRequest;
6  import javax.servlet.http.HttpServletResponse;
7  import javax.servlet.http.HttpSession;
8
9  import livraria.negocio.CarrinhoCompras;
10 import livraria.negocio.Livraria;
11 import livraria.negocio.Livro;
12 import livraria.negocio.excecoes.CompraException;
13 import livraria.negocio.excecoes.LivroNaoEncontradoException;
14
15 @WebServlet(name="Controlador", urlPatterns={"/livros/livraria"})
16 public class ServletControladorLivraria extends HttpServlet {
17    private static final long serialVersionUID = 1L;
18    public void doGet(HttpServletRequest request, HttpServletResponse response) {
19       String idLivro = null;
20       String limpar = null;
21       Livro livro = null;
22       Livraria livraria = (Livraria) getServletContext().getAttribute(
23       LivrariaContextListener.SISTEMA_LIVRARIA);
24       HttpSession session = request.getSession();
25
26       CarrinhoCompras carrinho = (CarrinhoCompras) session.getAttribute("cart");
27       if (carrinho == null) {
28          carrinho = new CarrinhoCompras();
29          session.setAttribute("cart", carrinho);
30       }
31
32       String acaoSelecionada = request.getServletPath();
33
34       // Aqui entram as operações do sistema
35
36       String tela = acaoSelecionada + ".jsp";
37
38       try {
39          request.getRequestDispatcher(tela).forward(request, response);
40       } catch (Exception ex) {
41           ex.printStackTrace();
42       }
43    }
44
45    public void doPost(HttpServletRequest request, HttpServletResponse response) {
46       doGet(request, response);
47    }
48 }
```

Listagem 2 - Código-fonte do Servlet controlador da livraria virtual

O arquivo livraria.jsp será o arquivo responsável por montar a tela inicial do sistema, a qual é mostrada na Figura 1. Ele deve ser criado dentro de uma pasta chamada livros. O código JSP desse arquivo é mostrado em seguida, na Listagem 3. Ele começa com a inclusão da taglib c, na linha 1, ou seja, você precisa adicionar a jstl-1.2.jar no WebContent/WEB-INF/lib, como visto em anteriormente). Seu código inclui, também, o uso de uma imagem chamada livro.gif, a qual é impressa antes e depois do texto Minha Livraria Virtual. Para implementar essa página, você deve escolher uma imagem que achar adequada. Faça uma busca na internet. Uma sugestão é usar o serviço de busca de imagens do Google (http://images.google.com.br).

Figura 01 - Tela inicial do sistema Livraria Virtual

```
1  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
2  <html>
3     <head>
4        <title>
5           Minha Livraria Virtual
6        </title>
7     </head>
8     <body bgcolor="#FFFFFF">
9        <center>
10          <hr>
11          <br>
12          <h1>
13             <img width="50" src="../imagem/livro.gif">
14             <font size="+3">Minha Livraria Virtual</font>
15             <img width="50" src="../imagem/livro.gif">
16          </h1>
17       </center>
18         <br>
19          <hr>
20            <p><b>Livraria Online</b></p>
21            <c:url var="url" value="/livros/catalogo" />
22            <p><b><a href="${url}?Add=">Iniciar compras</a></b>
23          <br>
24          <hr>
25     <center><em>Copyright © 2010 Livraria Virtual. </em></center>
26    </body>
27 </html>
```

Listagem 3 - Código JSP do arquivo responsável por montar a tela inicial mostrada na Figura 1

Já na linha 30, temos o uso do marcador , o qual é responsável por montar a URL correta para o caminho /livros/catalogo. Por exemplo, se seu contexto web tiver o nome Livraria, o acesso a essa página deve ser http://localhost:8080/Livraria/livros/catalogo. Sendo assim, o valor atribuído à variável URL, criada pelo marcador , terá valor igual a /Livraria/livros/catalogo. Isso torna o sistema independente do nome a ser utilizado no contexto web.

A URL armazenada na variável url é, então, utilizada na linha 32 com o objetivo de criar a referência para a página de compra de livros. Note o acesso à variável url, adicionado do texto "?Add=". Isso fará com que a URL gerada seja, na verdade, igual a /Livraria/livros/catalogo?Add=. O uso desse parâmetro Add passado com valor vazio será visto mais adiante, quando estudarmos o código do arquivo catalogo.jsp

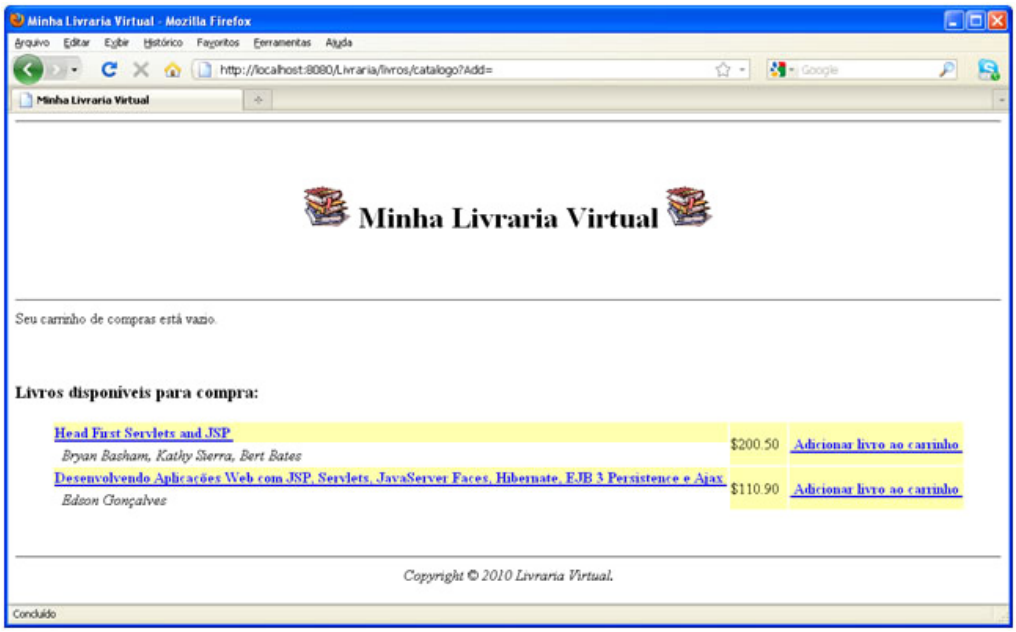
Atividade 02

1. Implemente o Servlet controlador da Livraria virtual, bem como sua página de entrada, conforme apresentado. Execute e verifique que o comportamento do código está correto.
2. Escreva um Servlet controlador e a página de entrada do sistema para o seu projeto de Locadora virtual. Execute o sistema e verifique se o seu Servlet e a página inicial da Locadora virtual estão funcionando corretamente.

Catálogo de livros

Agora que você já viu o código do Servlet controlador, vamos passar para o desenvolvimento da próxima tela do sistema: a do catálogo de livros. Essa tela é mostrada na Figura 2 e apresenta a lista de livros disponíveis para o usuário, além de informações sobre o carrinho de compras.

Figura 02 - Tela de catálogo de livros quando o carrinho de compras está vazio



Para implementar essa tela, criamos, na pasta livros, o arquivo catalogo.jsp. Note que a URL de acesso à pagina mostrada na Figura 1 termina com /livros/catalogo?Add=, por isso, o nome do nosso arquivo tem que ser catalogo.jsp e ele tem que estar na pasta livros. Lembrando que toda nova URL criada para o sistema tem que ser adicionada na anotação urlPatterns do nosso Servlet controlador, como mostraremos a seguir, para que ela seja processada adequadamente.

```
... @WebServlet(name="Controlador",
urlPatterns={"/livros/livraria","/livros/catalogo"})
public class ServletControladorLivraria extends HttpServlet{
...
```

Vamos agora estudar o conteúdo desse arquivo, o qual será apresentado aos poucos, iniciando-se pela Listagem 4.1. Nas primeiras duas linhas do arquivo, vemos a declaração das taglibs c e fmt. Em seguida, temos a montagem da parte superior da tela, a qual é a mesma da tela inicial do sistema e de todas as outras. Isso quer dizer que estamos duplicando código, não é mesmo? Duplicar código significa aumentar, desnecessariamente, o tamanho do código-fonte, além de dificultar a manutenção e entendimento do sistema. Mas, não se preocupe, na próxima aula, depois de entender bem o código da tela de catálogo, veremos como tratar esse problema!

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

<html>
   <head>
      <title>
         Minha Livraria Virtual
      </title>
   </head>
   <body bgcolor="#FFFFFF">
      <center>
         <hr>
         <br>
         <h1>
            <img width="50" src="../imagem/livro.gif">
            <font size="+3">Minha Livraria Virtual</font>
            <img width="50" src="../imagem/livro.gif">
         </h1>
      </center>
         <br>
         <hr>
```

Listagem 4.1 - Código-fonte do arquivo catalogo.jsp – Parte 1

O restante do conteúdo do arquivo catalogo.jsp irá acessar o sistema da livraria virtual para fazer consulta a livros disponíveis etc. Essas consultas exigirão o uso da classe LivrariaBean, mostrada na Listagem 5. Essa classe representa um bean que irá implementar as operações necessárias para a execução do arquivo JSP. Podemos ver que essa classe tem como atributo o próprio sistema da livraria (atributo sistema, do tipo Livraria) e um atributo idLivro do tipo String. O primeiro atributo receberá, como veremos adiante, o objeto Livraria armazenado no contexto dos Servlets. Já o segundo atributo representa o identificador do livro escolhido pelos usuários em operações como adicionar um livro ao carrinho.

Observe nessa classe como funciona os métodos getLivro() e getLivros(). O primeiro retorna o objeto do tipo Livro, que possui identificador igual ao do atributo idLivro. Já o segundo método retorna todos os livros em estoque na livraria. Por fim, temos o método comprarLivros(), que realiza a compra dos livros existentes no carrinho de compras recebido como parâmetro. Esses métodos fazem uma “ponte”, ou seja, uma interligação entre o código dos arquivos JSP e o sistema (classe Livraria). Dessa forma seu código fica mais organizado, isolando a camada de apresentação de HTML (no JSP), as ações a serem realizadas no LivrariaBean e mantém o processamento dos dados na classe Livraria. Veremos seu uso à medida que apresentamos os códigos JSP.

```
1  package livraria.negocio;
2
3  import java.util.List;
4
5  import livraria.negocio.excecoes.CompraException;
6  import livraria.negocio.excecoes.LivroNaoEncontradoException;
7
8  public class LivrariaBean {
9     private Livraria sistema = null;
10    private String idLivro = "0";
11
12    public LivrariaBean() {
13    }
14
15    public void setIdLivro(String id) {
16       this.idLivro = id;
17    }
18
19    public void setSistema(Livraria livraria) {
20       this.sistema = livraria;
21    }
22
23    public Livro getLivro() throws LivroNaoEncontradoException {
24       return (Livro) sistema.getLivro(idLivro);
25    }
26
27    public List<Livro> getLivros() {
28       return sistema.getLivros();
29    }
30
31    public void comprarLivros(CarrinhoCompras cart) throws CompraException {
32       sistema.comprarLivros(cart);
33    }
34 }
```

Listagem 5 - Código da classe LivrariaBean

Voltando para o arquivo catalogo.jsp, a segunda parte de seu código é mostrada na Listagem 4.2. Observe a declaração de uso do LivrariaBean nas linhas 25 a 27. O marcador <jsp:useBean > indica a classe do bean a ser instanciado (já que o escopo é page, o bean será instanciado em cada acesso). O identificador do bean (atributo id) é livrariaBean, o qual é utilizado no restante do arquivo para referenciar o bean. Sua propriedade de nome sistema é configurada com o valor da expressão ${sistemaLivraria}, o qual irá retornar o atributo do contexto dos Servlets de nome “sistemaLivraria”. Relembrando, esse atributo é criado pela classe LivrariaContextListener, vista anteriormente, e referencia a instância criada para o sistema (objeto Livraria).

```
1  <jsp:useBean id="livrariaBean" class="livraria.negocio.LivrariaBean" scope="page" >
2    <jsp:setProperty name="livrariaBean" property="sistema" value="${sistemaLivraria}" />
3  </jsp:useBean>
```

Listagem 4.2 - Código-fonte do arquivo catalogo.jsp – Parte 2

Por questões didáticas, as linhas 29 a 36 desse arquivo, que são relacionadas ao parâmetro da requisição Add, serão apresentadas no final da aula. Com relação ao código mostrado na Listagem 4.3, ele é responsável por mostrar o conteúdo do carrinho de compras. Usamos um marcador para simular uma instrução if-then-else. Na linha 38, é testado se o atributo numeroItens do objeto carrinho de compras (propriedade cart da sessão) possui valor maior que zero, ou seja, testa se existe pelo menos um item no carrinho de compras. Caso esse teste seja satisfeito (verdadeiro), o conteúdo das linhas 39 a 45 serão executados. Um link de nome url será criado com valor /livros/mostrarCarrinho, passando os parâmetros limpar e remover com valor igual a 0. Esse link levará a página que apresenta o conteúdo do carrinho de compras (linha 43), a qual será vista na próxima aula. Já as linhas 44 e 45 criam um link para a página de compra dos itens encontrados no carrinho. Essa página também será vista em outra aula.

Caso não existam itens no carrinho de compras, ou seja, o atributo numeroItens do carrinho de compras seja igual a 0, a mensagem “Seu carrinho de compras está vazio.” será apresentada, conforme mostrado nas linhas 47 a 49.

```
1  <c:choose>
2     <c:when test="sessionScope.cart.numeroItens > 0">
3        <c:url var="url" value="/livros/mostrarCarrinho">
4           <c:param name="limpar" value="0"/>
5           <c:param name="remover" value="0"/>
6        </c:url>
7        <p>
8          <a href="${url}">Ver carrinho de compras</a>
9          <c:url var="url" value="/livros/comprar" />
10         <a href="${url}">Finalizar compras</a></strong>
11       </p>
12    </c:when>
13    <c:otherwise>
14       Seu carrinho de compras está vazio.
15    </c:otherwise>
16 </c:choose>
```

Listagem 4.3 - Código-fonte do arquivo catalogo.jsp – Parte 3

A próxima parte do código, mostrado na Listagem 4.4, é responsável por mostrar a lista de livros disponíveis para compra. Para cada livro serão mostradas informações e links para se visualizar os detalhes do livro e para se adicionar o livro no carrinho de compras. Para isso, usamos o marcador <c:forEach >, visto na linha 58. Observe que a coleção de objetos utilizada no comando for é determinada pela expressão ${livrariaBean.livros}, ou seja, pelo retorno do método getLivros() do objeto livrariaBean. Cada objeto dessa coleção será armazenado na variável de nome livro e utilizado em comandos como o da linha 60, que pega o atributo idLivro e armazena seu valor em uma variável local também chamada de idLivro. A criação de um link para a página de apresentação dos detalhes do livro é mostrada nas linhas 62 a 64, adicionando como parâmetro a variável idLivro.

A linha 65 é responsável por apresentar o título do livro que está selecionado pelo marcador <c:forEach >. Já na linha 68, temos o uso do marcador <fmt:formatNumber > para apresentar o preço do livro como moeda. Isso fará a inclusão do símbolo $ antes do número e o uso de duas casas decimais para apresentação do valor.

```
1    <br>
2  <br>
3  <h3>Livros disponíveis para compra:</h3>
4  <div>
5   <table summary="layout">
6     <c:forEach var="livro" begin="0" items="${livrariaBean.livros}">
7     <tr>
8       <c:set var="idLivro" value="${livro.idLivro}" />
9       <td bgcolor="#ffffaa">
10        <c:url var="url" value="/livros/detalhesLivro" >
11          <c:param name="idLivro" value="${idLivro}"/>
12        </c:url>
13        <a href="${url}"><strong>${livro.titulo}</strong></a>
14      </td>
15         <td bgcolor="#ffffaa" rowspan=2>
16         <fmt:formatNumber value="${livro.preco}" type="currency"/>
17       </td>
18       <td bgcolor="#ffffaa" rowspan=2>
19         <c:url var="url" value="/livros/catalogo" >
20           <c:param name="Add" value="${idLivro}"/>
21           </c:url>
22           <a href="${url}">Adicionar livro ao carrinho</a>
23       </td>
24        </tr>
25    <tr>
26      <td bgcolor="#ffffff">
27          <em>${livro.autores} </em>
28      </td>
29    </tr>
30    </c:forEach>
31   </table>
32  </div>
33
```

Listagem 4.4 - Código-fonte do arquivo catalogo.jsp – Parte 4

Na linha 72, temos a criação de uma URL para a própria página de catálogo, passando o parâmetro Add com valor igual ao identificador do livro (idLivro) que o usuário tenha selecionado. Isso representa a adição desse item ao carrinho de compras. Muito bem! Finalmente chegamos ao uso do parâmetro Add. Ele é utilizado para indicar o livro que o usuário quer adicionar ao carrinho de compras. Essa operação é realizada adicionando-se o seguinte trecho de código ao Servlet controlador, logo após a linha 32 da classe ServletControladorLivraria mostrada na Listagem 2:

```
if (acaoSelecionada.equals("/livros/catalogo")) {
    idLivro = request.getParameter("Add");

    if (!idLivro.equals("")) {
        try {
            livro = livraria.getLivro(idLivro);
            carrinho.adicionar(livro);
        } catch (LivroNaoEncontradoException ex) {
            // isso não deve acontecer
        }
    }
}
```

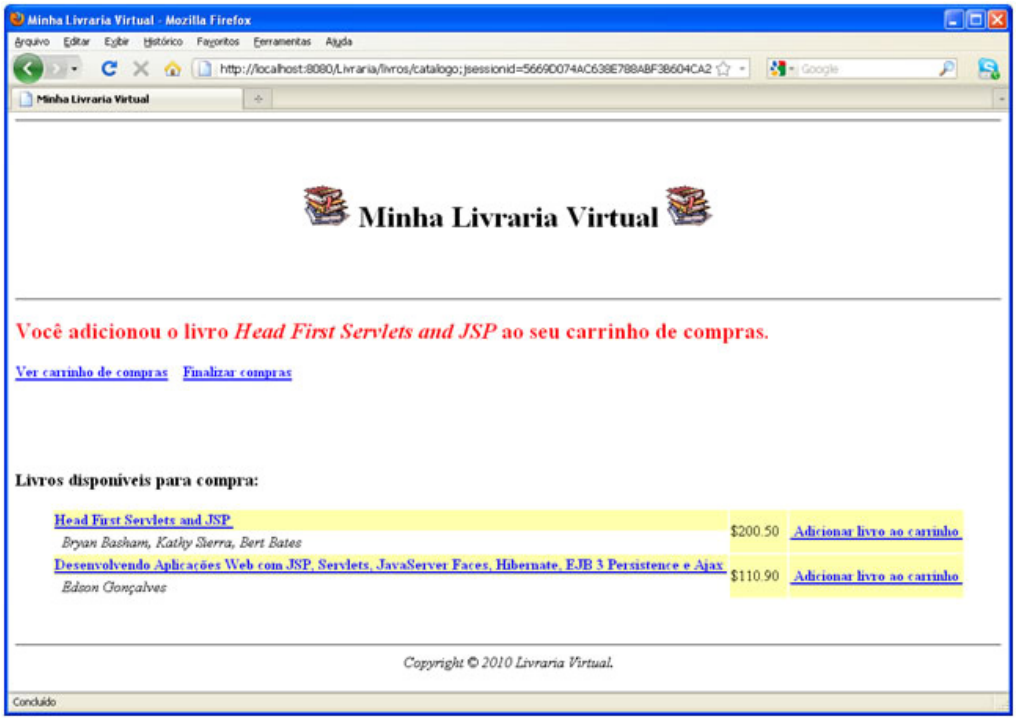
Esse código verifica se a ação selecionada (representada pela parte final da URL acessada) é igual a "/livros/catalogo" (parâmetros da requisição não entram aqui). Sendo o caso, pega-se o valor do parâmetro Add e, se ele não for vazio, busca-se o livro por meio do objeto livraria e o adiciona ao carrinho de compras. A busca do livro pode gerar uma exceção, o que não deve ocorrer normalmente, pois o código do livro é escrito no link de forma automática pelo sistema. Caso aconteça a exceção devido a um acesso indevido por um hacker, por exemplo, ela fará com que a operação de adição do livro no carrinho não ocorra, sem comprometer a geração da página de resposta.

Caso a operação de adição do livro ao carrinho de compras seja executada com sucesso, o arquivo catalogo.jsp será, novamente, carregado e, em especial, teremos também a execução do código mostrado na Listagem 4.5. Esse código representa as linhas do arquivo catalogo.jsp que não foram mostradas anteriormente e deve ser adicionado após o fechamento da Tag </jsp:useBean> para funcionar corretamente. Como podemos observar, testa-se o parâmetro Add para ver se ele está vazio. Se não estiver, é porque a operação de adicionar um livro ao carrinho foi executada. Se existe um código de livro no parâmetro Add, então, o ServletControladorLivraria, em sua execução, adicionou esse livro ao carrinho de compras. Esse trecho de código é, então, utilizado para apresentar uma mensagem confirmando a execução dessa operação, como mostrado pela Figura 3 (ver mensagem em vermelho).

```
1   <c:if test="${not empty param.Add}">
2   <c:set var="idL" value="${param.Add}"/>
3   <jsp:setProperty name="livrariaBean" property="idLivro" value="${idL}" />
4   <c:set var="livroAdicionado" value="${livrariaBean.livro}" />
5   <h3>
6     <font color="red" size="+2">Você adicionou o livro <em>${livroAdicionado.titulo}</em> ao seu
7   </h3>
8 </c:if>
```

Listagem 4.5 - Código-fonte do arquivo catalogo.jsp responsável por mostrar o livro que acabou de ser adicionado ao carrinho de compras

Figura 03 - Tela de catálogo recarregada após usuário clicar no link de adicionar livro ao carrinho



Por fim, temos o final do código do arquivo catalogo.jsp, mostrado na Listagem 4.6. Esse trecho de código é responsável por escrever um rodapé com uma mensagem de direitos autorais, a mesma que é mostrada na página inicial do sistema e que também será mostrada nas demais páginas do sistema. Na próxima aula, veremos como evitar que esse código seja duplicado em todas as páginas. Imagine ter que alterar o copyright para 2015 em todas as páginas do sistema! Seria bem melhor ter que fazer isso em um só lugar, não acha?

```
    <br>
    <hr>
        <center><em>Copyright © 2010 Livraria Virtual. </em></center>
    </body>
</html>
```

Listagem 4.6 - Código-fonte do arquivo catalogo.jsp – Parte 5

Atividade 03

1. Implemente as classes e o arquivo JSP apresentado para o sistema da Livraria virtual. Execute e verifique se o seu comportamento está de acordo com o esperado. Lembre-se de que os links utilizados para realizar as operações de ver detalhes de um livro, ver carrinho de compras e finalizar compras ainda não estarão funcionando!

2. Realize uma implementação para o seu projeto de Locadora virtual, para apresentação de um catálogo de livros, de forma similar ao realizado para a Livraria virtual.

3. Veja o código-fonte das páginas HTML geradas e identifique que trechos de código são gerados dinamicamente e por quais comandos JSP.

Leitura Complementar

Para complementar a fixação do conhecimento, faça uma breve leitura das aulas anteriores sobre JSP, sua linguagem de expressões e JSTL. Verifique, também, artigos na internet sobre esse assunto, como, por exemplo:

JAVASERVER Pages - Syntax Reference. Disponível em: <http://www.oracle.com/technetwork/java/syntaxref12-149806.pdf>. Acesso em: 03 mar. 2015.

Resumo

Nesta aula, você estudou o papel do Servlet controlador no desenvolvimento do sistema da Livraria virtual. Uma estratégia foi montada para que o nome das URLs do sistema tenha relação com o nome dos arquivos JSP, facilitando, assim, o desenvolvimento do sistema. Você também estudou a aplicação dos marcadores JSP e do bean LivrariaBean no desenvolvimento da página de catálogo de produtos.

Na próxima aula, continuaremos a adicionar funcionalidades ao sistema e apresentar o uso das tecnologias aprendidas até agora neste curso. Até lá!

Autoavaliação

1. Descreva como você implementou o modelo Model-View-Controller, em especial:
    a. O papel do Servlet controlador no desenvolvimento dos sistemas como o da Locadora virtual.
    b. O papel dos arquivos JSP.
    c. O papel de beans como o de nome LivrariaBean.

Referências

AHMED, K. Z.; UMRYSH, C. E. **Desenvolvendo aplicações comerciais em Java com Java J2EE e UML**. Rio de Janeiro: Ciência Moderna, 2003. 324 p.

BASHAM, Bryan; SIERRA, Kathy; BATES, Bert. **Head First Servlets and JSP**: passing the Sun Certified Web Component Developer Exam (SCWCD). 2. ed.: O'Reilly Media, 2008.

CATTELL, Rick; INSCORE, Jim. **J2EE:** criando aplicações comerciais. Rio de Janeiro: Campus, 2001.

HALL, Marty. **More Servlets and JavaServer Pages (JSP)**. New Jersey: Prentice Hall PTR (InformIT). 2001. 752 p Sun Microsystems Core Series. Disponível em: <http://pdf.moreservlets.com/>. Acesso em: 20 ago. 2012.

HALL, Marty; BROWN, Larry. **Core Servlets and JavaServer Pages (JSP).** 2. ed. New Jersey: Prentice Hall PTR (InformIT). 2003. 736 p. Sun Microsystems Core Series, (Core Technologies, 1). Disponível em: <http://pdf.coreservlets.com/>. Acesso em: 20 ago. 2012.

HYPERTEXT Transfer Protocol -- HTTP/1.1: methods definition. Disponível em: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>. Acesso em: 24 jul. 2010.

J2EE Tutorial. Disponível em: <http://docs.oracle.com/javaee/7/tutorial/>. Acesso em: 03 mar. 2015.