Desenvolvimento Web I

Aula 15 - Desenvolvendo uma livraria virtual – Parte 4

Apresentação

Olá, caro aluno, estamos quase finalizando nosso exemplo de aplicação em Servlet e JSP, a Livraria Virtual. Nesta aula, você estudará a implementação das demais funcionalidades que iremos implementar, assim como incorporar o uso de Javascript ao seu projeto. Desejo-lhe uma boa aula!

Vídeo 01 - Apresentação

Objetivos

- Descrever quando e como usar as tecnologias Servlet e JSP.
- Definir como implementar a funcionalidade de mostrar carrinho e de efetuar a compra dos livros.
- Reconhecer como a quantidade de itens de cada produto do carrinho de compras pode ser alterada para se tornar mais interativa.
- Validar, no próprio navegador web, os dados digitados pelo usuário no formulário de finalização de compras.

Funcionalidades relativas ao carrinho de compras

Muito bem, já estudamos quase toda a implementação do sistema de Livraria Virtual que estamos propondo. Para terminarmos, falta apenas estudarmos a implementação das funcionalidades relacionadas ao carrinho de compras, como as relacionadas a seguir.

- Mostrar o conteúdo do carrinho de compras.
- Realizar a compra e visualizar o recibo.

Lembrando que, na aula anterior, já vimos a configuração da anotação no servlet. Basicamente, o seguinte trecho de código foi colocado no arquivo, indicando as URLs que serão utilizadas para acionar essas funcionalidades:

```
1    @WebServlet(name="Controlador", urlPatterns = {"/livros/livraria", "/livros/catalogo", "/livros/d
```

Atividade 01

1. Verifique se a configuração da anotação do servlet controlador de seu projeto Livraria Virtual contém a configuração para as funcionalidades citadas.
2. Verifique se a configuração da anotação do servlet controlador de seu projeto de Locadora Virtual contém a configuração para as funcionalidades citadas.

Mostrar conteúdo do carrinho

Vamos começar com a implementação da funcionalidade de apresentação do conteúdo do carrinho de compras. Considere a tela apresentada pela **Figura 1**. Dado que foi colocado livros no carrinho de compras, podemos, agora, acessar o link “Ver

carrinho de compras" para visualizar o conteúdo do carrinho. A tela que apresenta o conteúdo do carrinho de compras é mostrada na **Figura 2**.

Figura 01 - Tela do sistema após adicionar um livro ao carrinho de compras

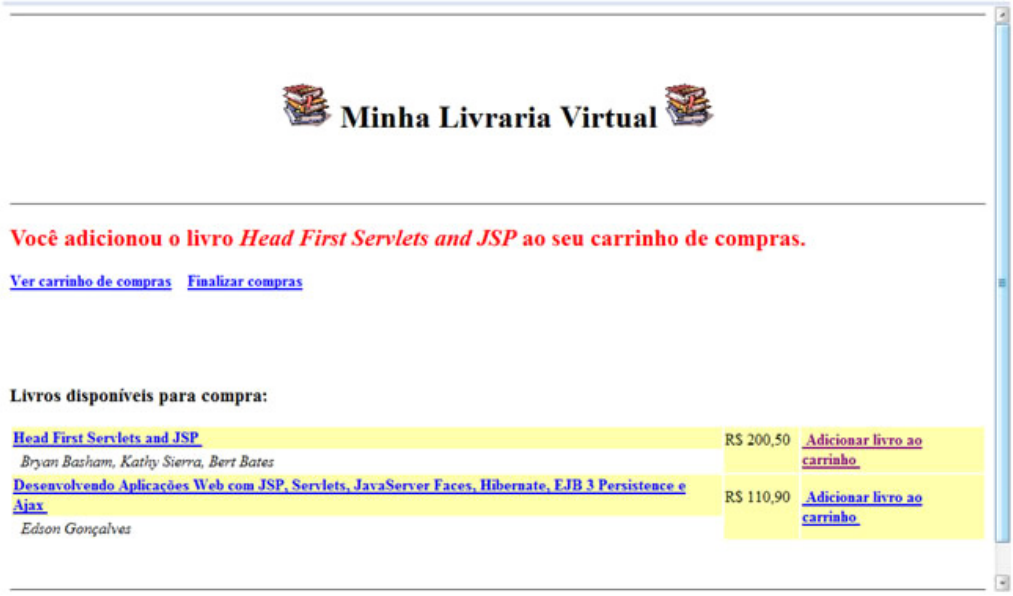


Figura 02 - Tela de visualização do conteúdo do carrinho de compras

Para implementar a tela que mostra o conteúdo do carrinho, precisamos criar o arquivo **mostrarCarrinho.jsp** na pasta **livros** (lembre de sempre adicionar o cabecalho.jsp e rodape.jsp on inicio e fim dos seus JSPs com a diretiva <%@ include >). Vamos estudar agora o conteúdo desse arquivo, começando pela declaração do LivrariaBean:

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

Esse código é o mesmo utilizado na página de catálogo, servindo para criar e configurar o bean **LivrariaBean.** Já para mostrar o conteúdo do carrinho, primeiro, vamos confirmar que o carrinho está, realmente, com algum conteúdo. Esse teste é feito no código abaixo (**<c:if >**), mostrando, inclusive, a quantidade de itens existente no carrinho (**${sessionScope.cart.numeroItens}**). Note que a decisão sobre o uso do singular (1 livro) ou plural (2 livros, por exemplo) também é feita usando o comando **<c:if>**.

```
<c:if test="${sessionScope.cart.numeroItens > 0}">
   <font size="+2">Quantidade de itens do carrinho: ${sessionScope.cart.numeroItens}
      <c:if test="{sessionScope.cart.numeroItens == 1}">
         livro.
      </c:if>
      <c:if test="${sessionScope.cart.numeroItens > 1}">
         livros.
      </c:if>
```

Em seguida, montamos uma tabela com um cabeçalho indicando as informações a serem apresentadas:

```
   </font><br>
   <table summary="layout">
   <tr>
      <th align="left">Quantidade</th>
      <th align="left">Título</th>
      <th align="left">Preço</th>
   </tr>
```

Depois disso, devemos apresentar os itens do carrinho, certo? Isso é feito usando uma instrução **<c:forEach >** para navegar nos itens do carrinho (objetos do tipo **ItemCompra**). Para cada item de compra, o livro é pego e armazenado. Informações como título e preço são apresentadas. Um link para visualizar os detalhes de cada livro e outro para removê-lo do carrinho também estão sendo criados usando-se a instrução **<c:url >**:

```
  <c:forEach var="itemCompra" items="${sessionScope.cart.itens}">
  <c:set var="livro" value="${itemCompra.item}" />
  <tr>
    <td align="right" bgcolor="#ffffff"> ${itemCompra.quantidade} </td>
    <td bgcolor="#ffffaa">
      <c:url var="url" value="/livros/detalhesLivro" >
        <c:param name="idLivro" value="${livro.idLivro}" />
            <c:param name="Clear" value="0" />
      </c:url>
      <strong><a href="${url}">${livro.titulo}</a></strong>
    </td>
    <td bgcolor="#ffffaa" align="right">
      <fmt:formatNumber value="${livro.preco}" type="currency"/> </td>
    <td bgcolor="#ffffaa">
      <c:url var="url" value="/livros/mostrarCarrinho" >
       <c:param name="remover" value="${livro.idLivro}" />
      </c:url>
      <strong><a href="${url}">Remover</a></strong>
    </td>
  </tr>
</c:forEach>
```

Além disso, temos que mostrar o total do valor da compra. Lembre-se de que o carrinho de compras tem um método chamado **getTotal()** para isso (**${sessionScope.cart.total}**):

```
  <tr>
    <td colspan="5" bgcolor="#ffffff">
    <br></td>
  </tr>
  <tr>
    <td colspan="2" align="right" bgcolor="#ffffff">Subtotal</td>
    <td bgcolor="#ffffaa" align="right">
      <fmt:formatNumber value="${sessionScope.cart.total}" type="currency"/>
    </td>
    <td><br></td>
  </tr>
</table>
```

Por fim, devemos criar os links para Continuar comprando, Finalizar compra e de Esvaziar carrinho de compras:

```
<c:url var="url" value="/livros/catalogo" >
   <c:param name="Add" value="" />
</c:url>
<strong><a href="${url}">Continuar comprando</a>
<c:url var="url" value="/livros/comprar" />
<a href="${url}">Finalizar compra</a>
<c:url var="url" value="/livros/mostrarCarrinho" >
 <c:param name="limpar" value="limpar" />
 <c:param name="remover" value="0" />
</c:url>
<a href="${url}">Esvaziar carrinho</a></strong>
</c:if>
```

Para que as operações de remover e de limpar possam funcionar, você deve alterar o código do Servlet controlador para considerar essas ações por meio do código mostrado a seguir que deve ser adicionado logo após o bloco onde é executada a verificação se a ação selecionada é "/livros/catalogo".. Note que se a URL **/livros/mostrarCarrinho** receber o **id** de um livro como parâmetro, ele será removido do carrinho. Além disso, se o parâmetro **limpar** tiver como valor a String "**limpar**", o carrinho será esvaziado (remoção de todos os itens). Isso está consistente com o código JSP que acabamos de ver. Em caso de dúvida, olhe novamente o código JSP mostrado anteriormente para verificar isso.

```
else if (acaoSelecionada.equals("/livros/mostrarCarrinho")) {
      idLivro = request.getParameter("remover");

   if (idLivro != null) {
      carrinho.remover(idLivro);
   }

   limpar = request.getParameter("limpar");

   if ((limpar != null) && limpar.equals("limpar")) {
      carrinho.limpar();
   }
}
```

Vamos aproveitar e adicionar algumas mensagens ao sistema? Logo depois da declaração de uso do bean **LivrariaBean**, podemos colocar o seguinte código:

```
    <c:if test="${param.limpar == 'limpar'}">
   <font color="red" size="+2">
      <strong>O carrinho de compras foi esvaziado!</strong><br> <br>
   </font>
</c:if>
<c:if test="${param.remover != '0'}">
   <c:set var="id" value="${param.remover}"/>
   <jsp:setProperty name="livrariaBean" property="idLivro" value="${id}" />
   <c:set var="livroRemovido" value="${livrariaBean.livro}" />
   <font color="red" size="+2">O seguinte livro foi removido do carrinho:
      <em>${livroRemovido.titulo}</em>.
      <br> <br>
   </font>
</c:if>
<c:if test="${sessionScope.cart.numeroItens <= 0}">
   <font size="+2">Carrinho vazio</font>
   <br> <br>
   <c:url var="url" value="/livros/catalogo" >
      <c:param name="Add" value="" />
   </c:url>
   <strong><a href="${url}">Ver catálogo</a></strong>
</c:if>
```

Isso fará com que uma mensagem de confirmação seja apresentada nos casos de esvaziar o carrinho ou de remover um livro dele. Além disso, incluímos uma mensagem de que o carrinho está vazio com um link para a página do catálogo para facilitar a navegação do usuário após ter esvaziado o carrinho. Olhe atentamente o código JSP mostrado e observe o uso das instruções JSP responsáveis pela apresentação dessas mensagens. As **Figuras 3, 4** e **5** ilustram isso.

Figura 03 - Carrinho de compras com dois livros

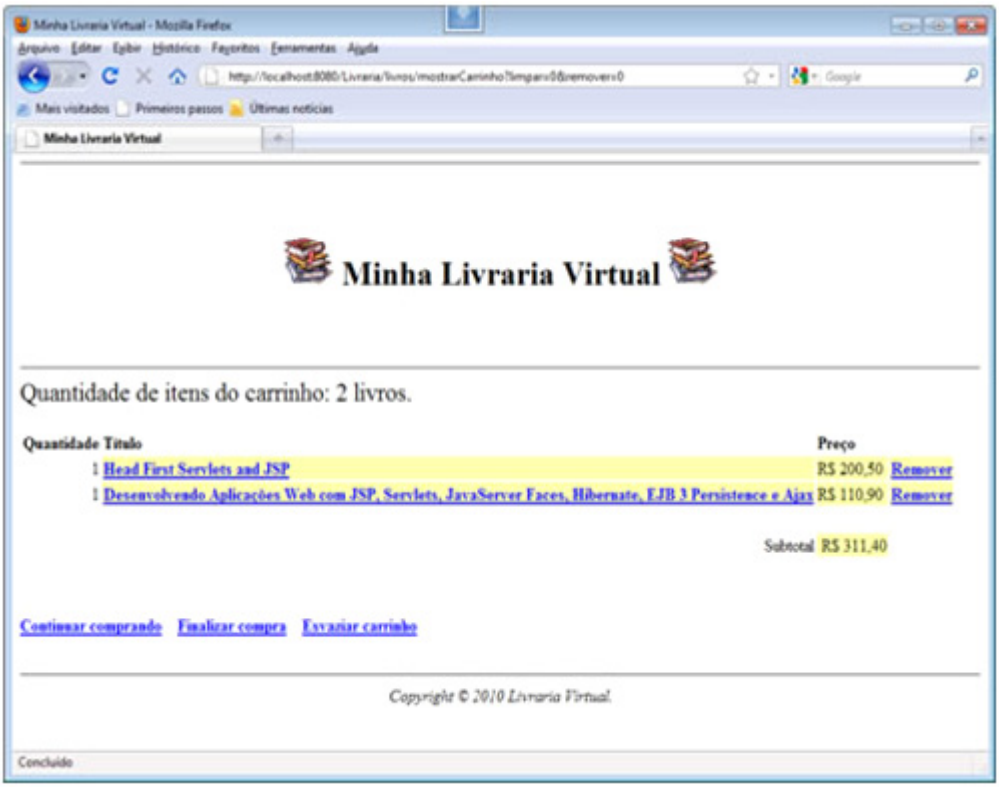


Figura 04 - Carrinho de compras com um dos livros removidos

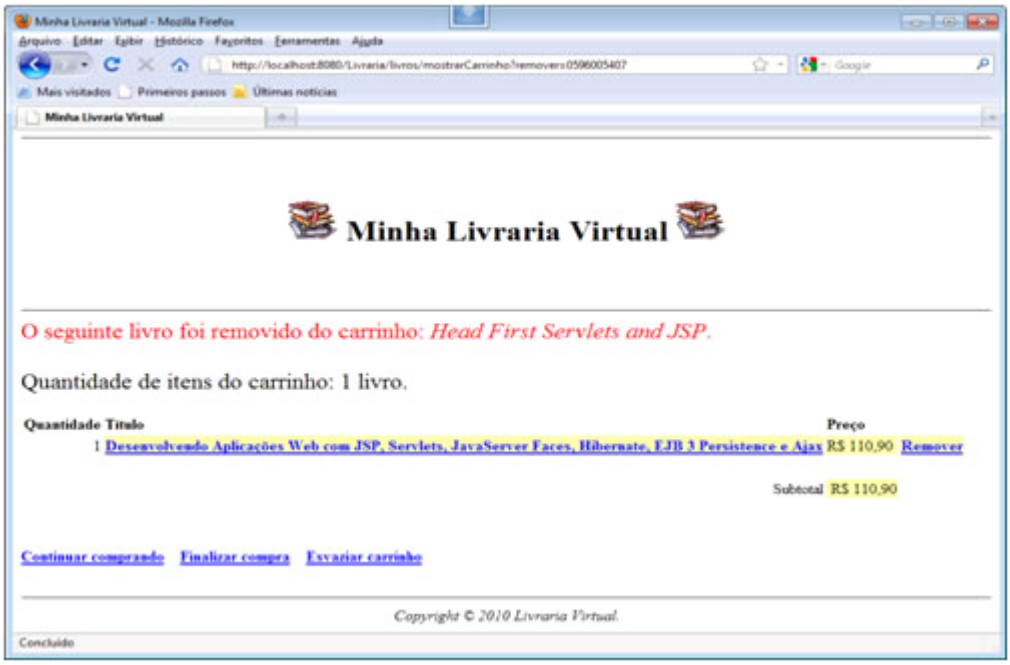
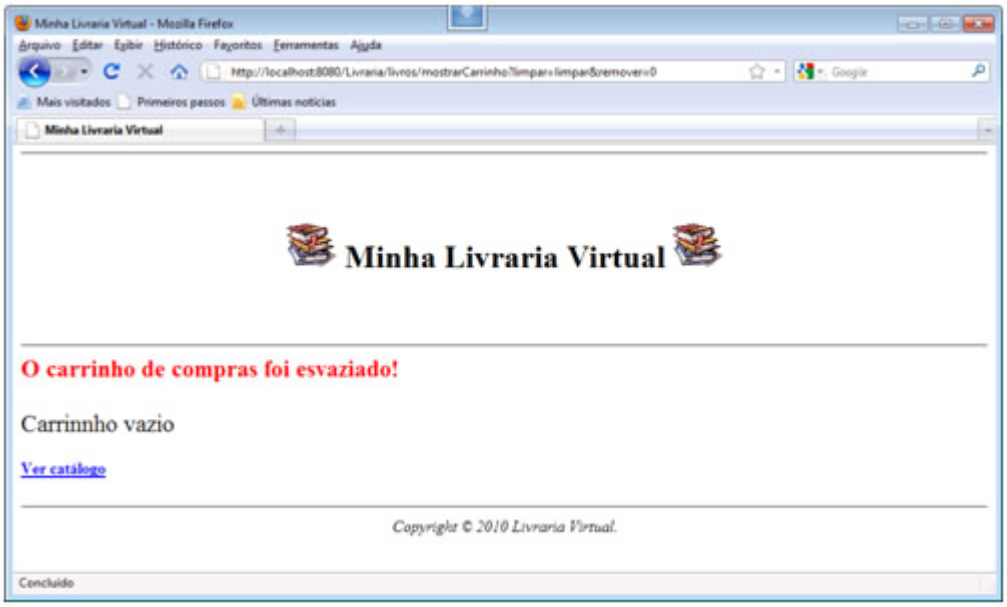


Figura 05 - Carrinho de compras esvaziado



Atividade 02

1. No sistema de Livraria Virtual, implemente a funcionalidade de mostrar carrinho de compras, conforme apresentado nesta aula. Execute e reporte se sua implementação está correta

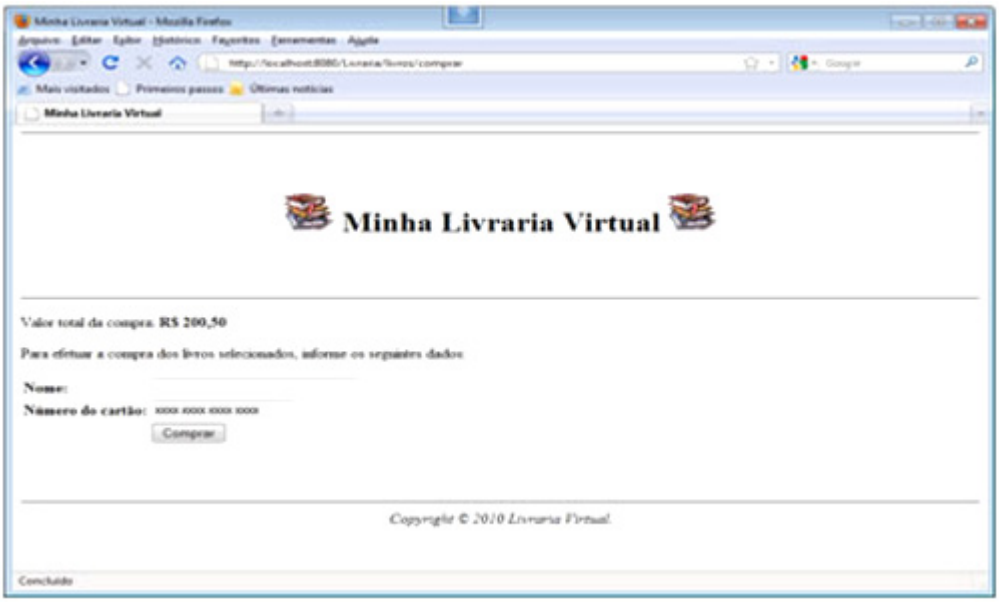
2. No seu projeto de Locadora Virtual, implemente a funcionalidade de mostrar carrinho de compras de forma similar àquela usada na Livraria Virtual. Execute e reporte se sua implementação está correta.

Realizar a compra do conteúdo do carrinho e mostrar recibo

Para efetuar a compra dos produtos encontrados no carrinho, o usuário deve clicar no link *Finalizar* compra. Esse link, segundo o código JSP do arquivo mostrarCarrinho.jsp, nos leva a URL /livros/comprar. O arquivo JSP relativo a essa URL (comprar.jsp) é mostrado a seguir. É apresentado o valor total da compra e um formulário contendo dois campos, o nome do usuário e o número de seu cartão de crédito (ver **Figura 5**). Depois de preencher essas informações, o usuário deve clicar no botão comprar para confirmar a compra.

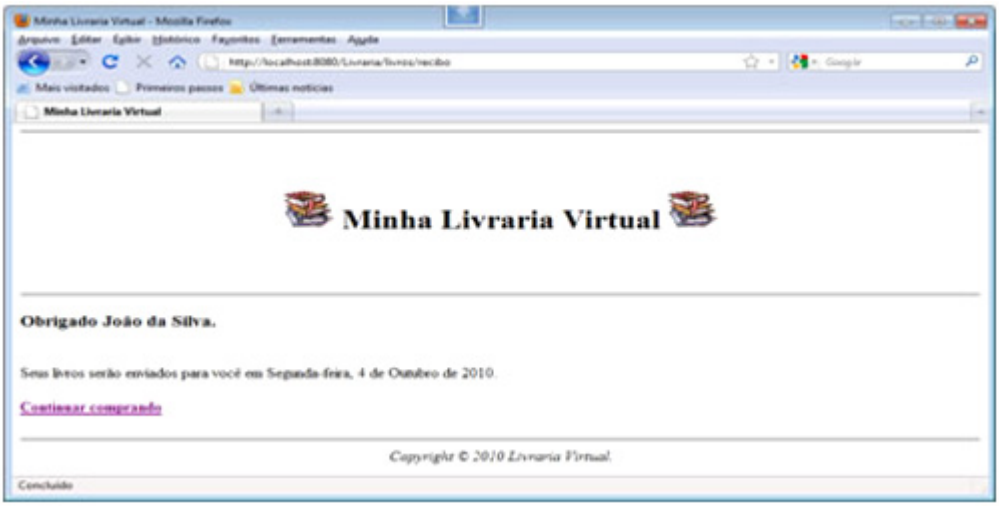
```
<%@ include file="/cabecalho.jsp" %>
<p>Valor total da compra:
<strong>
  <fmt:formatNumber value="${sessionScope.cart.total}" type="currency"/>
</strong>
<p>Para efetuar a compra dos livros selecionados, informe os seguintes dados:
<c:url var="url" value="/livros/recibo" />
<form action="${url}" method="post">
  <table summary="layout">
    <tr>
      <td><strong>Nome:</strong></td>
      <td><input type="text" name="nome" value="" size="30"></td>
    </tr>
    <tr>
      <td><strong>Número do cartão:</strong></td>
      <td><input type="text" name="cardnum"
            value="xxxx xxxx xxxx xxxx" size="19"></td>
    </tr>
    <tr>
      <td></td>
    <td><input type="submit" value="Comprar"></td>
    </tr>
  </table>
</form>
<%@ include file="/rodape.jsp" %>
```

Figura 06 - Tela de compra dos livros



Ao clicar no botão comprar, o usuário é levado a URL /livros/recibo. Essa URL irá gerar um recibo de compra para o usuário. Isso é mostrado pela **Figura 7.** É apresentada uma mensagem de agradecimento com o nome do usuário e indica-se que a data de envio dos livros é de cinco dias depois da data de compra (data atual), visando confirmação do pagamento do cartão.

Figura 07 - Tela de recibo de compra



Vídeo 02 - Carrinho de Compras, Finalizar Compras e Recibo

Para gerar o código da tela de recibos, primeiro temos que efetuar a compra dos livros. Isso é feito pelo método comprarLivros() da livraria, o qual é acionado pelo Servlet controlador, quando adicionamos a ele o seguinte código:

```
1  else if (acaoSelecionada.equals("/livros/recibo")) {
2     try {
3        livraria.comprarLivros(carrinho);
4     } catch (CompraException ex) {
5        ex.printStackTrace();
6     }
7  }
```

Já para montar a página JSP de recibo/confirmação de compra, temos o seguinte código para o arquivo recibo.jsp:

```
<%@ include file="/cabecalho.jsp" %>
<h3>Obrigado ${param.nome}.</h3><br>
<jsp:useBean id="now" class="java.util.Date" />
<jsp:setProperty name="now" property="time" value="${now.time+432000000}"/>

Seus livros serão enviados paravocê em
<fmt:formatDate value="${now}" type="date" dateStyle="full"/>.<br><br>
<c:remove var="cart" scope="session" />
<c:url var="url" value="/livraria.jsp" />
<strong>
   <a href="${url}">Continuar comprando</a>
</strong>
<%@ include file="/rodape.jsp" %>
```

Note o uso da classe Date e da configuração da data representada pelo objeto como igual à data atual (now.time) adicionada de 5 dias (equivalente a 432000000 milissegundos).

Atividade 03

1. No sistema de Livraria Virtual, implemente a funcionalidade de efetuar a compra dos livros adicionados ao carrinho de compras, conforme apresentado nesta aula. Execute e reporte se sua implementação está correta.

2. No seu projeto de Locadora Virtual, implemente a funcionalidade de efetuar a compra de forma similar àquela usada na Livraria Virtual. Execute e reporte se sua implementação está correta.

Implementar a funcionalidade de “Alterar carrinho de compras”

Iremos agora implementar a funcionalidade de alterar o carrinho de compras para permitir que o usuário tenha a possibilidade de modificar a quantidade de cada item do seu carrinho de compras diretamente na tela de “Visualizar Carrinho”. Iremos implementar esse recurso adicionando duas figuras (com os símbolos “+” e “-”), ao lado da quantidade de cada livro, que quando acionados irão ser mapeados para as funções de incrementar e decrementar a quantidade, respectivamente. O

primeiro código a ser modificado será o do arquivo mostrarCarrinho.jsp. O código abaixo contém todas as alterações que foram feitas no arquivo (em seguida, as modificações serão explicadas).

```
    <%@ include file="/cabecalho.jsp" %>
<jsp:useBean id="livrariaBean" class="livraria.negocio.LivrariaBean" scope="page" >
  <jsp:setProperty name="livrariaBean" property="sistema" value="${sistemaLivraria}" />
</jsp:useBean>
<c:if test="${param.limpar == 'limpar'}">
  <font color="red" size="+2">
    <strong>O carrinho de compras foi esvaziado!</strong><br> <br>
  </font>
</c:if>
<c:if test="${param.remover != '0' && param.remover != null}">
  <c:set var="id" value="${param.remover}"/>
  <jsp:setProperty name="livrariaBean" property="idLivro" value="${id}" />
  <c:set var="livroRemovido" value="${livrariaBean.livro}" />
  <font color="red" size="+2">O seguinte livro foi removido do carrinho:
    <em>${livroRemovido.titulo}</em>.
    <br> <br>
  </font>
</c:if>
<c:if test="${sessionScope.cart.numeroItens > 0}">
  <font size="+2">Quantidade de itens do carrinho: ${sessionScope.cart.numeroItens}
  <c:if test="${sessionScope.cart.numeroItens == 1}">
    livro.
  </c:if>
  <c:if test="${sessionScope.cart.numeroItens > 1}">
    livros.
  </c:if>
  </font><br>
<table summary="layout">
  <tr>
    <th align=left colspan="3">Quantidade</th>
    <th align=left>Título</th>
    <th align=left>Preço</th>
  </tr>
  <c:forEach var="itemCompra" items="${sessionScope.cart.itens}">
    <c:set var="livro" value="${itemCompra.item}" />
    <tr>
      <td bgcolor="#ffffff">
        <c:url var="url" value="/livros/mostrarCarrinho" >
          <c:param name="alterar" value="${livro.idLivro}" />
          <c:param name="quantidade" value="1" />
          <c:param name="remover" value="0" />
        </c:url>
        <a href="${url}" style="text-decoration: none;">[+]</a>
      </td>
      <td bgcolor="#ffffff">
        <c:url var="url" value="/livros/mostrarCarrinho" >
          <c:param name="alterar" value="${livro.idLivro}" />
          <c:param name="quantidade" value="-1" />
          <c:param name="remover" value="0" />
        </c:url>
        <a href="${url}" style="text-decoration: none;">[-]</a>
```

```
        </td>
        <td align="left" bgcolor="#ffffff">${itemCompra.quantidade}</td>
        <td bgcolor="#ffffaa">
          <c:url var="url" value="/livros/detalhesLivro" >
          <c:param name="idLivro" value="${livro.idLivro}"/>
          <c:param name="Clear" value="0"/></c:url>
          <strong><a href="${url}">${livro.titulo}</a></strong>
        </td>
        <td bgcolor="#ffffaa" align="right">
          <fmt:formatNumber value="${livro.preco}" type="currency"/>
        </td>
        <td bgcolor="#ffffaa">
          <c:url var="url" value="/livros/mostrarCarrinho" >
          <c:param name="remover" value="${livro.idLivro}"/></c:url><strong><a href="${url}">Ren
        </td>
      </tr>
    </c:forEach>
    <tr>
      <td colspan="5" bgcolor="#ffffff"><br></td>
    </tr>
    <tr>
      <td colspan="2" align="right" bgcolor="#ffffff">Subtotal</td>
      <td bgcolor="#ffffaa" align="right">
        <fmt:formatNumber value="${sessionScope.cart.total}" type="currency"/>
      </td>
      <td><br></td>
    </tr>
</table>
<p> <p>
<c:url var="url" value="/livros/catalogo" >
    <c:param name="Add" value="" />
</c:url>
<strong>
    <a href="${url}">Continuar comprando</a>
      <c:url var="url" value="/livros/comprar" />
        <a href="${url}">Finalizar compra</a>
        <c:url var="url" value="/livros/mostrarCarrinho">
        <c:param name="limpar" value="limpar"/>
        <c:param name="remover" value="0"/>
      </c:url>
    <a href="${url}">Esvaziar carrinho</a>
</strong>
</c:if>
<c:if test="${sessionScope.cart.numeroItens <= 0}">
<font size="+2">Carrinho vazio</font>
<br> <br>
<c:url var="url" value="/livros/catalogo" >
    <c:param name="Add" value="" />
</c:url>
<strong><a href="${url}">Ver catálogo</a></strong>
</c:if>
```

```
103 <%@ include file="/rodape.jsp" %>
```

Listagem 1 - Código do arquivo livros/mostrarCarrinho.jsp, localizado no diretório WebContent/livros

A primeira modificação é relativa à tabela HTML. Na coluna “Quantidade”, faremos com que ela ocupe três colunas, pois iremos adicionar mais duas células em cada linha (além da que já guarda a informação de quantidade), uma para o símbolo do “+” e outra para o “-”.

Dentro do ForEach, necessitamos adicionar essas duas novas células (tags) . Basicamente, incluímos uma imagem com um link para a própria URL de mostrarCarrinho (pois o carrinho vai continuar sendo exibido). No entanto, dessa vez, iremos passar dois parâmetros chamados “alterar” e “quantidade”. O primeiro guarda o código do livro a ter sua quantidade modificada e o outro guarda a quantidade desse livro a ser incrementada no carrinho.

A única diferença entre a célula que contém o incremento e a que contém o decremento são a imagem e o valor do parâmetro quantidade (que pode ser 1 ou -1, para incrementar e decrementar, respectivamente).

Outra modificação que precisa ser feita é no código do servlet. Como uma nova operação que não existia antes precisa ser executada (incremento e decremento de quantidade de itens do carrinho), precisamos também modificar o código do **ServletControladorLivraria**. Neste arquivo iremos melhorar a execução da ação de mostrarCarrinho, conforme o trecho abaixo (que representa somente a parte do código do servlet que foi afetada).

```
} else if (acaoSelecionada.equals("/livros/mostrarCarrinho")) {
    idLivro = request.getParameter("remover");

      if (idLivro != null) {
       carrinho.remover(idLivro);
      }

      limpar = request.getParameter("limpar");

      if ((limpar != null) && limpar.equals("limpar")) {
       carrinho.limpar();
      }

      idLivro = request.getParameter("alterar");

      if (idLivro != null) {
       int quantidade = Integer.parseInt(request.getParameter("quantidade"));

       if(quantidade == 1){
        carrinho.aumentarQuantidade(idLivro);
       }
       else if(quantidade == -1){
        carrinho.diminuirQuantidade(idLivro);

       }
    }
}
```

Listagem 2 - Código parcial do ServletControladorLivraria

Vídeo 03 - Adicionando Controle de Quantidade ao Carrinho

Como podemos ver, nesse incremento da funcionalidade, verificamos se foi solicitada uma alteração (isso é feito verificando se o parâmetro alterar é diferente de null). Caso isso seja verdade, recuperamos a quantidade e avaliamos se é um decremento ou incremento, chamando em seguida o método correto do carrinho de compras. Esses dois novos métodos são aumentarQuantidade() e diminuirQuantidade().

Por fim, iremos modificar a classe CarrinhoCompras para implementar esses dois novos métodos, conforme mostrado no trecho de código abaixo.

```
public void aumentarQuantidade(String idLivro) {
    if (itens.containsKey(idLivro)) {
        ItemCompra item = itens.get(idLivro);
        item.incrementaQuantidade();
    }
}
public void diminuirQuantidade(String idLivro) {
    if (itens.containsKey(idLivro)) {
        ItemCompra item = itens.get(idLivro);
        item.decrementaQuantidade();
    }
}
```

Listagem 3 - Novos métodos da classe CarrinhoCompras

Atividade 04

1. Melhore o sistema de Locadora Virtual de filmes, desenvolvida como atividade nas aulas anteriores, implementando a funcionalidade Incrementar e Decrementar itens do carrinho.

Melhorando a funcionalidade de “Finalizar Compras”

A última mudança que faremos na Livraria Virtual é relacionada ao processo de finalização de compra.

Nessa modificação, iremos obrigar o usuário a digitar nome e número do cartão de crédito para finalizar a compra. Essa verificação será feita através do uso de JavaScript, aplicado aos campos do formulário HTML. A solução completa pode ser vista no trecho de código abaixo.

```
<%@ include file="/cabecalho.jsp" %>
<script>
   function comprar(){

      var form = document.getElementById("form");
      var nome = document.getElementById("nome");
      var cartao = document.getElementById("cardnum");

      if(nome.value == null || nome.value == ""){

         alert("Digite seu nome");
         return false;

      } else if(cartao.value == null || cartao.value == "" || cartao.value=="xxxx.xxxx.xxxx.xxxx") {
         alert("Digite o número do cartão");
         return false;
      } else {
         form.submit();
      }
   }

   function limparCartao(){
      var cartao = document.getElementById("cardnum");

      if(cartao.value == "xxxx.xxxx.xxxx.xxxx"){
         cartao.value = "";
      }
   }

</script>

<p>Valor total da compra:

<strong><fmt:formatNumber value="${sessionScope.cart.total}" type="currency"/></strong>

<p>Para efetuar a compra dos livros selecionados, informe os seguintes dados:<c:url var="url"

<form id="form"action="${url}" method="post">
   <table summary="layout">
      <tr>
         <td><strong>Nome:</strong></td>
         <td><input type="text" name="nome" id="nome" value="" size="30"></td>
      </tr>

      <tr>
         <td><strong>Número do cartão:</strong></td>
         <td><input type="text" name="cardnum" id="cardnum" onfocus="limparCartao()" value=
      </tr>

      <tr>
         <td></td>
```

```
52              <td><input type="button" onClick="comprar()" value="Comprar"></td>
53            </tr>
54
55          </table>
56        </form>
57  <%@ include file="/rodape.jsp" %>
```

Listagem 4 - Script de finalização de compra

Vídeo 04 - Adicionando JavaScript a Finalização de Compras

Como pode ser visto, as mudanças realizadas estão destacadas. Inicialmente, foram definidas duas funções JavaScript. A primeira delas, de nome **comprar()**, será chamada quando o usuário clicar no botão “Comprar”, sendo responsável por validar se os dois campos foram preenchidos pelo usuário antes de permitir que o formulário seja enviado ao servidor. A segunda função, **limparCartao()**, tem como objetivo limpar o campo “Número do Cartão”, assim que o mesmo for selecionado.

Atividade 05

1. Melhore o sistema de Locadora Virtual de filmes, desenvolvido como atividade das aulas anteriores, da seguinte forma:
    - Acrescente campos de endereço e demais dados de cartão de crédito.
    - Faça a validação dos campos antes de permitir que os dados sejam enviados ao servidor. Não é necessário fazer nenhum processamento ou armazenamento desses dados no servidor.

Resumo

Nesta aula, você estudou a implementação das funcionalidades de mostrar o conteúdo do carrinho de compras e de efetuar a compra dos livros, mostrando, inclusive, um prazo de entrega de cinco dias. Foi possível perceber o uso constante de instruções JSP, como as taglib JSTL e marcadores **<jsp:useBean>** e **<jsp:setProperty>** . Além disso, você praticou o uso de JavaScript, melhorando o sistema que você já vinha desenvolvendo em aulas anteriores. Como você deve ter notado, o uso do JavaScript acrescenta mais possibilidades para a sua página Web, tornando-a mais poderosa e atrativa. A partir de agora, use JavaScript sempre que for desenvolver algum sistema Web, principalmente em Java, pois ele é uma ferramenta muito importante para esses tipos de aplicações.

Autoavaliação

1. Descreva como as tecnologias dos Servlets e JSP foram usadas para o desenvolvimento da Livraria Virtual.
2. Descreva, de forma geral, como foi implementada a funcionalidade de mostrar carrinho e de efetuar a compra dos livros.
3. Identifique outros pontos em que você poderia usar JavaScript para melhorar a interatividade do sistema e implemente essas melhorias.

Leitura Complementar

Para complementar a fixação do conhecimento, faça uma releitura dos links mostrados a seguir.

- <http://www.oracle.com/technetwork/java/syntaxref12-149806.pdf > Acesso em: 10 mar. 2015.
- <http://cs.roosevelt.edu/eric/books/JSP/jstl-quick-reference.pdf>. Acesso em: 20 ago. 2012.
- <http://www.java2s.com/JSTL/UsingtheJSTLfunctions.htm>. Acesso em: 20 ago. 2012.
- <http://download.oracle.com/tutorial/bnakc.html>. Acesso em: 20 ago. 2012.

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), Sun Microsystems Core Series, 2001. 752 p. 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), Sun Microsystems Core Series, 2003. 736 p. (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: 20 ago. 2012

J2EE Tutorial. Disponível em: <http://docs.oracle.com/javaee/7/tutorial/>.Acesso em: 10 mar. 2015.

JAVASCRIPT and HTML DOM reference. Disponível em: <http://www.w3schools.com/jsref/>. Acesso em: 27 ago. 2012.

W3C. Disponível em: <http://www.w3.org/>. Acesso em: 27 ago. 2012.

W3SCHOOL. Disponível em: <http://www.w3schools.com/>. Acesso em: 27 ago. 2012.