Java na WEB – Componentes Bean

Objetivo:
Ao final da aula o aluno será capaz de:

- Criar site dinâmico utilizando componentes Bean
- Utilizar Tags de ação
- Empregar Coleções de Bean.

Sumário

Prof. Vida.
vida@utfpr.edu.br

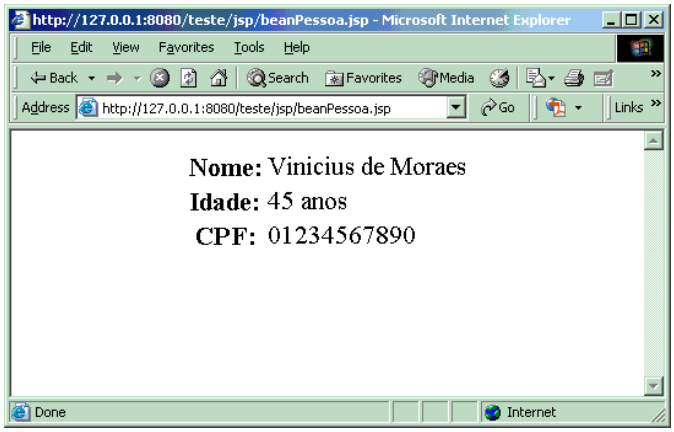
5. Definição de JavaBean

Introdução:	Um JavaBean é um componente de software que pode ser reutilizado e vinculado facilmente a outros componentes para formar novas aplicações. Ele deve ser implementado de acordo com algumas regras ou convenções.
Convenção de Criação:	Essa é a primeria convenção a ser considerada no desenvolvimento. A classe necessita implementar a interface `java.io.Serializable`. Exemplo: `public class MinhaClasse implements java.io.Serializable{ }` A interface Serializable não contém declaração de métodos e, portanto, o JavaBean não necessita implementar nenhum método específico. Essa interface têm a função marcadora para a classe e indica que instâncias podem ser serializadas.
Convenção de Modificadores de Acesso.	Em um Bean, todas as variáveis necessitam possuir o encapsulamento private e os seus identificaores devem ser escritos somente com letras minúsculas.
Convenção de Construtor	Um Bean necessita possuir um construtor sem parâmetro algum. Esse construtor é utilizado pelo contêiner para instanciá-lo. Um Bean também admite a definição de contrutores com parâmetros. Caso o programador opte por utilizar um construtor dessa forma, um construtor sem parâmetro necessita ser definido, mesmo que esse contrutor não execute operação alguma. No exemplo a seguir, essa exigência é atendida n linha 07 do código. Nessa linha, um contrutor sem parâmetro algum é implementado.

Exemplo de Implementação de Construtores:	`// arquivo Pessoa.java`
	
	`/*00*/ package javaNaWeb.beans;`
	`/*01*/ public class Pessoa implements`
	`/*02*/ java.io.Serializable`
	`/*03*/ {`
	`/*04*/ String nome="";`
	`/*05*/ int idade=0;`
	`/*06*/ String cpf="";`
	
	`/*07*/ public Pessoa() {}`

```
/*08*/    public Pessoa(String vNome, int vIdade
/*09*/                 , String vCpf)
/*10*/    {  setNome(vNome);
             setIdade(vIdade); setCpf(vCpf);}

// metodos de acesso: leitura
/*11*/   public String getNome() {return nome;}
/*12*/   public int getIdade() {return idade; }
/*13*/   public String getCpf() {return cpf; }

//metodos de acesso:escrita
/*14*/   public void setNome(String v ) {nome=v;}
/*15*/   public void setIdade(int v) {idade=v;}
/*16*/   public void setCpf(String v) {cpf=v;}

/*17*/   public String toString(){
/*18*/      String saida = "\n Nome: " + nome
/*19*/         +"\n Idade: " + idade +  " anos"
/*20*/         +"\n CPF: " + cpf;
/*21*/      return saida;
/*22*/ } }
```

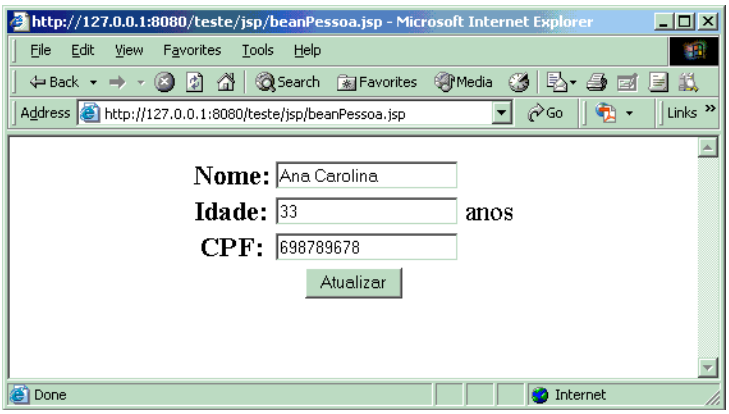
Implementação de Métodos em um Bean	Os métodos em um Bean são implemtentados da mesma forma que classes normais. Esses métodos podem ser implementados para realizar quaisquer tipos de operações.
O que torna a classe Pessoa um Bean?	A classe Pessoa foi criada de maneira similar à qualquer outra classe, mas com algumas pequenas diferenças que são oriundas das regras do bean, a saber: • **Implementa a interface java.io.Serializable.** • **Os atributos ou propriedades da classe são privados.** • **Construtor sem parâmetro algum.** • **Implementação de Métodos de acesso.**

Como utilizar um Bean em uma página JSP?	`<%@ page import="javaNaWeb.beans.*" %>`
	`<% Pessoa p = new Pessoa();`
	`p.setNome("Vinicius de Moraes");`
	`p.setIdade(45);`
	`p.setCpf("01234567890");`
	`%>`
	`<html><body>`
	`<center>`
	`<table border=0>`
	`<tr><th> Nome: </th> <td> <%= p.getNome()`
	`%> </td></tr>`
	`<tr><th> Idade: </th> <td> <%= p.getIdade()`
	`%> anos </td></tr>`
	`<tr><th> CPF: </th> <td> <%= p.getCpf() %>`
	`</td></tr>`
	`</table></body></html>`
	 

Como utilizar o Bean em uma sessão.

```
<%@ page import="javaNaWeb.beans.*" %>
<% Pessoa p = (Pessoa)
            session.getAttribute("pessoa");
   if (p==null) {p= new Pessoa();}
%>
<%
  String nome =
         (String)request.getParameter("nome");
  String idade =
         (String)request.getParameter("idade");
  String cpf =
         (String)request.getParameter("cpf");

  if (nome != null) {p.setNome(nome);}
  if (idade != null)
  {p.setIdade(Integer.parseInt(idade.trim()));}
  if (cpf != null) {p.setCpf(cpf.trim());}
  session.setAttribute("pessoa",p);
%>

<html>
<body>
<center>
<table border=0>
<form action="beanPessoa.jsp" method=”post”>
<tr><th> Nome: </th>
    <td> <input type="text" name = "nome"
     value="<%= p.getNome().trim() %>">
    </td></tr>
<tr><th> Idade: </th>
    <td> <input type="text" name = "idade"
     value="<%= p.getIdade() %>"> anos
    </td></tr>
<tr><th> CPF: </th>
     <td> <input type="text"name = "cpf"
      value="<%= p.getCpf().trim() %>">
      </td></tr>
<tr><td align=center colspan=2>
       <input type="submit" value="Atualizar">
```

```
</form></td></tr>
 </table>
 <br>
 </body></html>
```



6. Coleção de Bean (exemplo)

Coleções de Bean podem ser utilizadas em diversas situações. Talvez, a mais típica seja o carrinho de compras de uma loja virtual. Cada produto inserido no carrinho pode ser representado como um bean e o carrinho de compras como a coleção de Bean. Essa coleção pode ser armazenada na sessão.

Uma coleção de Bean pode ser definida utilizando-se arrays ou a classe java.util.Vector, por exemplo.

6.1 Definição do Bean

```
////////////////////////////////////// arquivo Produto.java
package javaNaWeb.mercado;

public class Produto implements java.io.Serializable{
 private int codigo;
 private String nome;
 private double valor;
 private int quantidade = 1;

 public Produto(){}

 public Produto(int _codigo, String _nome, double _valor)
 {  this.codigo = _codigo;  this.nome= _nome;
```

```
    this.valor = _valor;}

 public int getCodigo() {return codigo;}
 public String getNome() {return nome;}
 public double getValor() {return valor;}

 public void setCodigo(int v) {codigo=v;}
 public void setNome(String v){nome=v;}
 public void setValor(double v) {valor=v;}

 public String toString()
 {  return ( "Codigo:" + codigo
          +  "\tNome :" + nome
          +  "\t Valor:"   +
java.text.NumberFormat.getCurrencyInstance().format(valor));}


 public boolean isEquals(Produto p)
 { boolean saida=false;
   if ((codigo==p.getCodigo())
    && (nome==p.getNome())
    && (valor==p.getValor())) {saida=true;}
   return saida; }

 public int getQuantidade()      {return quantidade;}
 public void setQuantidade(int v){quantidade += v;}
 public void incrementaQuantidade() {quantidade++;}
}
```

6.2 Definição da Página que coletará os dados

Após a definição do Bean, implementa-se a página JSP que coletará os dados do produto. Por exemplo:

```
<html>
<body>
<h1> Cadastro de Produtos </h1>
<center>
```

```
<form action="registrar.jsp" method="post" border=1>
<table border=0 align="center">
  <tr bgcolor="gray"> <th> codigo: </th> <td> <input type="text"
                name="codigo" size=8 maxlength=8> </td></tr>
  <tr bgcolor="white"> <th> nome: </th> <td> <input type="text"
                name="nome" size=80 maxlength=250> </td></tr>
  <tr bgcolor="gray"> <th> valor: </th> <td> <input type="text"
                name="valor" size=8 maxlength=8> </td></tr>
  <tr bgcolor="White"> <td colspan="2" align="center">
                  <input type="submit" value="Gravar" > </td></tr>
</table>
</body></html>
```

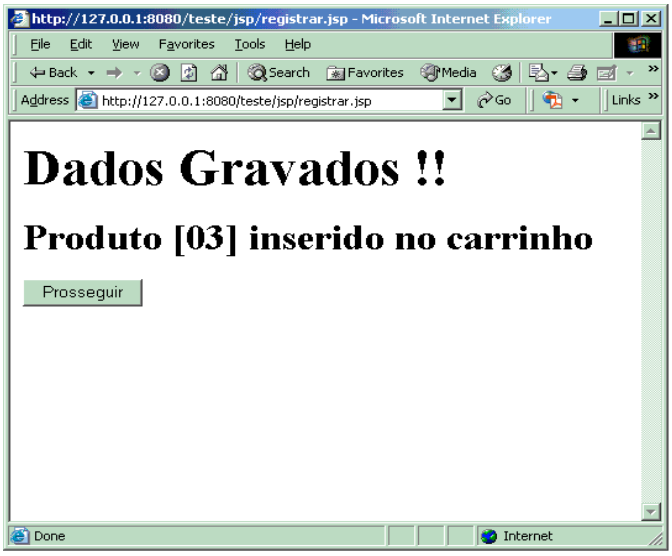
6.3 Definição da Página que processará as informações:

```
<!------------------------ Arquivo registrar.jsp >

<%@page import="java.util.Vector, javaNaWeb.mercado.*"  %>
<%@page errorPage="erro.jsp" %>

<% Vector dados= (Vector)session.getAttribute("registros");
   if (dados == null) {dados = new Vector();}
%>
<%!
  private int jahTem(Vector lista
                  , javaNaWeb.mercado.Produto novoProduto)
```

```
  {  int saida = -1;
     Produto aux;
     for (int i = 0; i < lista.size(); i++)
     { aux = ((Produto)lista.elementAt(i));
       if ( aux.isEquals(novoProduto) ==true) {saida=i; break;}
     }
     return saida;
  }
%>
<% Produto p = new Produto();
   String lidoCodigo = request.getParameter("codigo");
   String lidoNome = request.getParameter("nome");
   String lidoValor = request.getParameter("valor");
   boolean gravado=false;
   String feito="";
%>

<% if (lidoCodigo != null)
               p.setCodigo(Integer.parseInt(lidoCodigo.trim()));
   if (lidoNome != null) p.setNome(lidoNome);
   if (lidoValor != null)
               p.setValor(Double.parseDouble(lidoValor.trim()));
%>

<%
    if ((lidoCodigo != null) || (lidoNome != null)
                             || (lidoValor != null) )
    {  int i = jahTem(dados,p);

       if (i > -1)
           { ((Produto)dados.elementAt(i)).incrementaQuantidade();
             feito=
             "Quantidade incrementada do produto:"+lidoCodigo;}
      else { dados.add(p);
             feito="Produto [" + lidoCodigo
                       + "] inserido no carrinho";
           }
       session.setAttribute("registros", dados);
       gravado=true;}
```

Prof. Vida.
vida@utfpr.edu.br

```
%>

<html><body>

<%
   if (gravado == false)
     {out.println("<h1> Solicitação sem dados !! </h1>");
      out.println(
"<form action='..\\html\\cadastroDeProduto.html' method='post'>");
      out.println(
"<input type='submit' value='Fornecer Informações'></form>");
      out.println("</body></html>"); }
   else
     {out.println("<h1> Dados Gravados !! </h1>");
      out.println("<h2>" + feito + "</h2>");
      out.println("<form action='relatorio.jsp' method='post' >");
      out.println(
                "<input type='submit' value='Prosseguir'></form>");
     }
%>
</body></html>
```



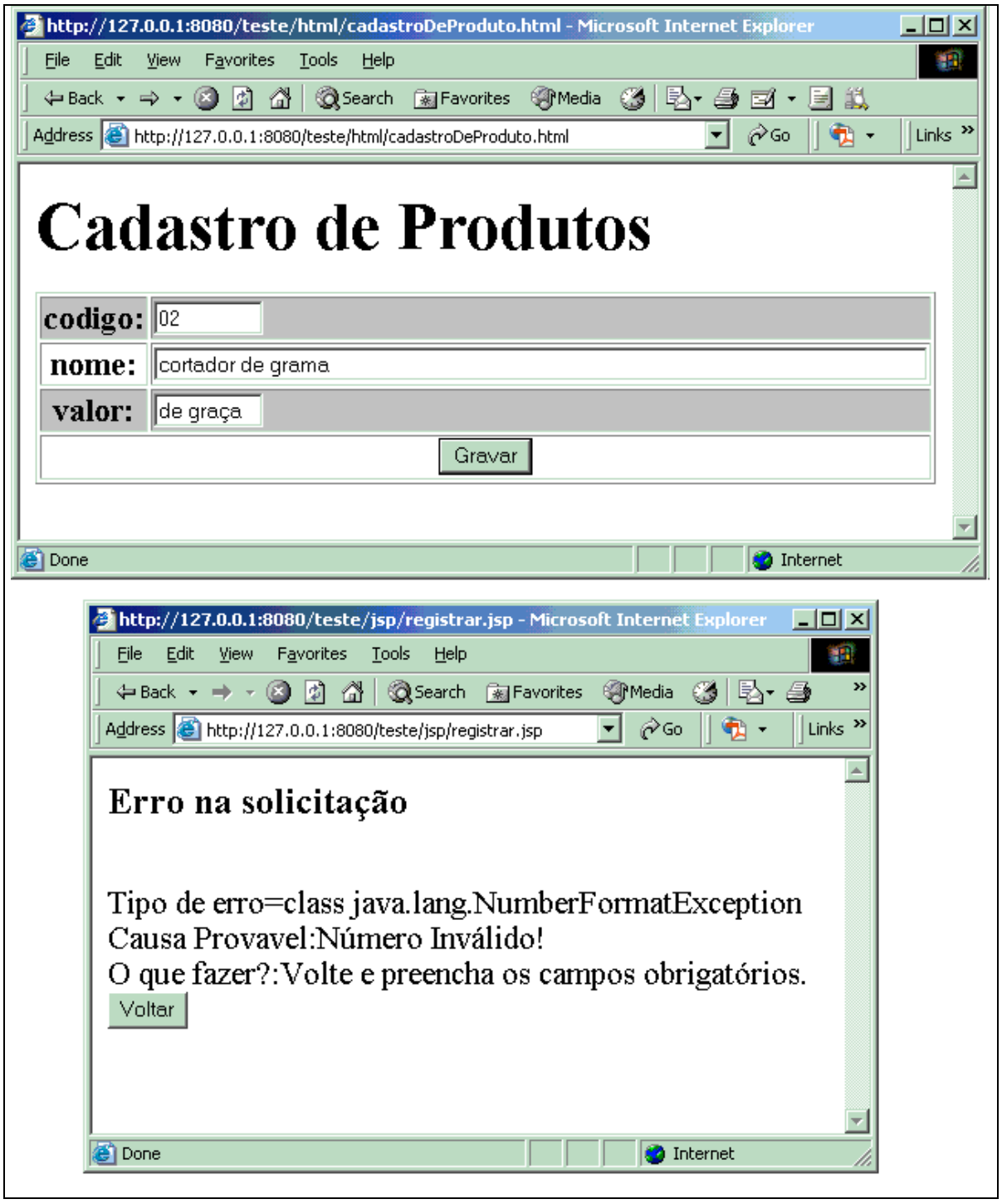
6.4 Definição da Página que fará o Tratamento de Erros.

Para o tratamento de erros, apenas para reduzir esforços, repete-se o código da página 27 da aula anterior e adiciona-se o botão voltar.

```
<%@page isErrorPage="true"%>
```

```
<html><body>
<h3> Erro na solicitação </h3>
<%
  String tipo ="";
  String causa="Desconhecida";
  String oQueFazer="Não definido !";
%>
<%
   if (exception != null)
    { tipo = exception.getClass().toString();

      if (tipo.indexOf("java.lang.NullPointerException")>=0)
      { causa = "Faltou informar um dado!";
        oQueFazer="Volte e preencha os campos obrigatórios.";
      }

      if (tipo.indexOf("java.lang.NumberFormatException")>=0)
      { causa = "Número Inválido!";
        oQueFazer="Volte e preencha os campos obrigatórios.";
      }

      out.println("<br>Tipo de erro="+tipo    );
      out.println("<br>Causa Provavel:" + causa);
      out.println("<br>O que fazer?:" + oQueFazer);
    }
%>
<br>
<input type="button" value="Voltar"
             onclick="javascript:history.back(1)"/>
</body>
</html>
```

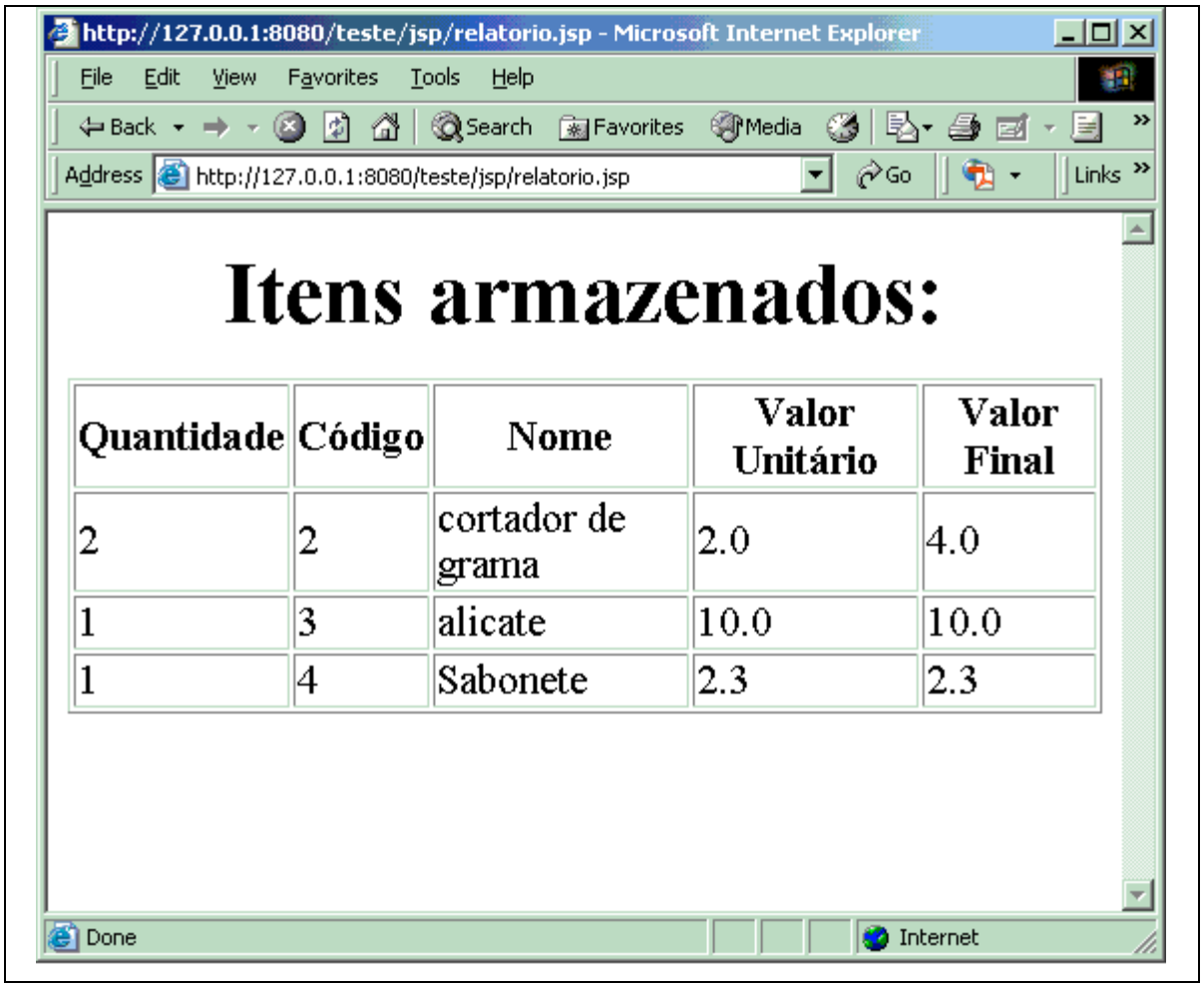
Prof. Vida.
vida@utfpr.edu.br



6.5 Definição da Página de Relatório

```
<!------------------------ Arquivo registrar.jsp >

<%@page import="java.util.Vector, javaNaWeb.mercado.*"  %>
```

```
<%@page errorPage="erro.jsp" %>

<% Vector dados= (Vector)session.getAttribute("registros");
   String lista="";
   if (dados != null)
   {
     Produto aux;
     for (int i = 0; i < dados.size(); i++)
     { aux = ((Produto)dados.elementAt(i));
       lista += "\n<tr><td>" + aux.getQuantidade() + "</td><td>"
                             + aux.getCodigo() + "</td><td>"
                             + aux.getNome() + "</td><td>"
                             + aux.getValor()*aux.getQuantidade()
                             + "</td></tr>";
     }
    }
%>

<html><body><center>
<h1> Itens armazenados:</h1>
<%
   if (lista.length() == 0)
     {out.println("<h2> Carrinho Vazio !! </h2>");}
   else
     {out.println("<table border = '1'>");
      out.println(
                  "<tr><th> Quantidade</th>"
                  +"<th>Código</th>"
                  +"<th>Nome</th>"
                  +"<th>Valor Unitário</th>"
                  +"<th>Valor Final</th></tr>");
      out.println(lista);
      out.println("</table>");
     }
%>
</body></html>
```



Prof. Vida.
vida@utfpr.edu.br

7. Tags de Bean.

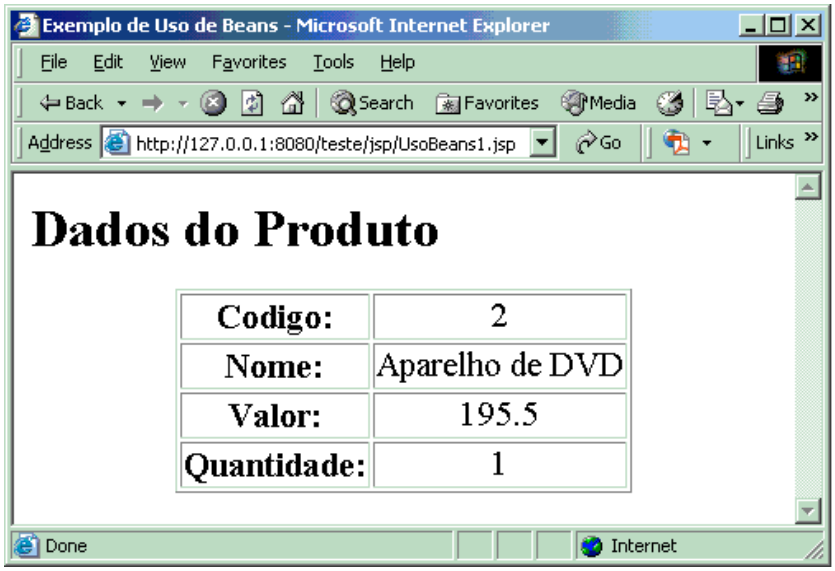
Tags:	• `<jsp:useBean>` : Utilizada para instanciar um JavaBean ou recuperar uma instância já existente. • `<jsp:setProperty>` : Utilizada para alterar o valor de uma propriedade. • `<jsp:getProperty>` : Utilizada para recuperar o valor de uma propriedade
Tag `<jsp:useBean>`	A instância de um bean é feita conforme o exemplo a seguir: `<jsp:useBean id="nome" class="nomeClasse" scope="escopo" />` Um bean pode conter cinco atributos, a saber: • `beanName` : instância de um bean • `class` : define a classe • `id` : especifica um identificador para o bean • `scope`: especifica o escopo onde o bean será gravado {page, request, session ou application} • `type` : força a conversão do bean para um tipo diferente da classe à qual pertence o bean.
Tag `<jsp:setProperty>`	Sempre que for necessário aletar a propriedade de um Bean já instanciado, deve-se utilizar a tag `<jsp:setProperty>`. São atributos da Tag: • `name`: o nome da instância do JavaBean criado. • `param`: define um parâmetro de requisição cujo conteúdo deve ser gravado em uma propriedade JavaBean • `property`: nome da propriedade a ser alterada. • `value`: O novo valor a ser gravado na propriedade.

Prof. Vida.
vida@utfpr.edu.br

Como usar a <jsp:setProperty>	• **<jsp:setProperty name="produto" property="quantidade" value="2"/> :** altera o valor da propriedade quantidade para o valor 2. • **<jsp:setProperty name="produto" property="quantidade" param="quant"/>** :altera o valor da propriedade quantidade para o valor do parâmetro quant. • **<jsp:setProperty name="produto" property="quantidade"/>** : Se o campo do formulário onde o usuário informou o valor a ser gravado tiver exatamente o mesmo nome que a propriedade, não é necessário o uso do atributo `param`. • **<jsp:setProperty name="produto" property="*"/>** : Se todos os parâmetros do formulário tiverem o mesmo nome das propriedades, não é necessário especificação de qual propriedade que será alterada nem de qual parâmetro.
A Tag <jsp:getProperty>	A Tag `<jsp:getProperty>` contém apenas dois atributos: `name` e `property`. Ela deve ser utilizada sempre que desejar recuperar o valor de uma propriedade. Exemplo de sintaxe de uso: **<jsp:getProperty name="produto" property="quantidade" />**

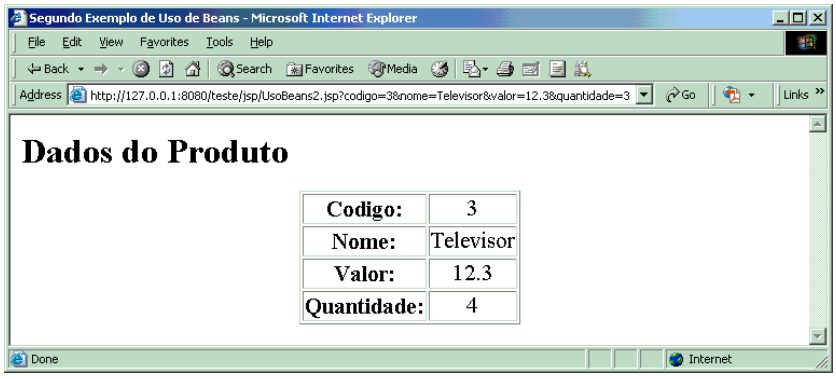
Arquivo UsoBean.jsp – exemplo de Uso de Bean:	`<jsp:useBean id="produto"`
	`  class="javaNaWeb.mercado.Produto" />`
	`<jsp:setProperty name="produto"`
	`  property="codigo" value="2" />`
	`<jsp:setProperty name="produto"`
	`  property="nome" value="Aparelho de DVD" />`
	`<jsp:setProperty name="produto"`
	`  property="valor"  value="195.5" />`
	
	`<html>`
	`<head><title> Exemplo de Uso de Bean`
	`</title></head>`
	`<body>`
	`<h2> Dados do Produto </h2>`
	`<center>`
	`<table border=1>`
	`<tr align="center"><th>Codigo: </th>`

```
     <td><jsp:getProperty name="produto"
              property="codigo"/></td></tr>
<tr align="center"><th>Nome: </th>
      <td><jsp:getProperty name="produto"
               property="nome"/></td></tr>
<tr align="center"><th>Valor: </th>
        <td><jsp:getProperty name="produto"
              property="valor"/></td></tr>
<tr align="center"><th>Quantidade: </th>
        <td><jsp:getProperty name="produto"
              property="quantidade"/></td></tr>

</table>
</body></html>
```



Arquivo UsoBeans2.jsp – exemplo de Uso de Bean:

```
<jsp:useBean id="produto"
class="javaNaWeb.mercado.Produto" />
<jsp:setProperty name="produto" property="*" />

<html>
<head><title> Segundo Exemplo de Uso de Bean
</title></head>
<body>
<h2> Dados do Produto </h2>
<center>
<table border=1>
<tr align="center"><th>Codigo: </th>
     <td><jsp:getProperty name="produto"
          property="codigo"/></td></tr>
<tr align="center"><th>Nome: </th>
     <td><jsp:getProperty name="produto"
          property="nome"/></td></tr>
<tr align="center"><th>Valor: </th>
      <td><jsp:getProperty name="produto"
          property="valor"/></td></tr>
<tr align="center"><th>Quantidade: </th>
      <td><jsp:getProperty name="produto"
          property="quantidade"/></td></tr>

</table>
</body></html>
```

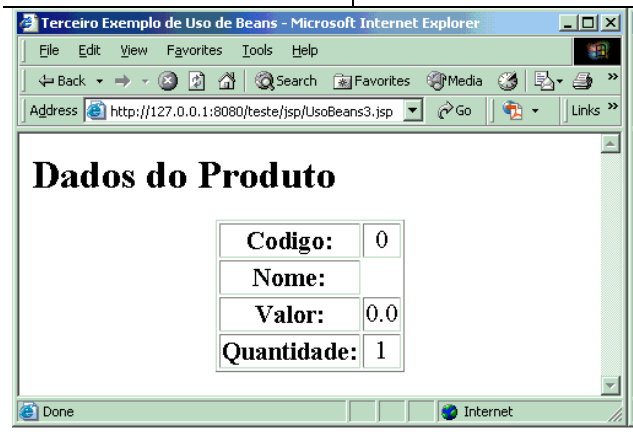


Armazenamento na Sessão

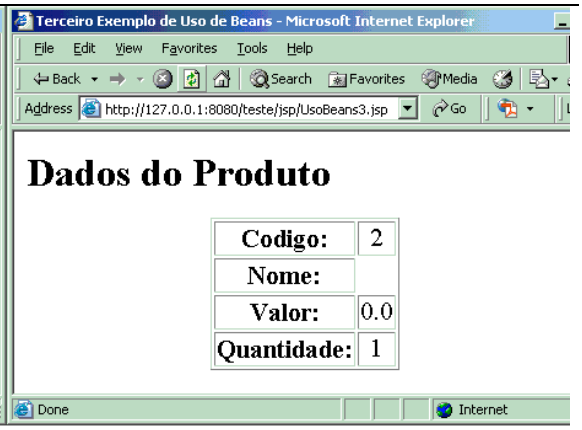
O compartilhamento dos Bean entre páginas JSP é feito através da sessão. Nesse caso, a instância do bean é associada a uma sessão.

Arquivo UsoBeans3.jsp - exemplo de Armazenamento em Sessão.

```
<jsp:useBean id="produto"
class="javaNaWeb.mercado.Produto"
scope="session"/>
<html>
<head><title> Terceiro Exemplo de Uso de Bean
</title></head>
<body>
<h2> Dados do Produto </h2>
<center>
<table border=1>
<tr align="center"><th>Codigo: </th>
       <td><jsp:getProperty name="produto"
           property="codigo"/></td></tr>
<tr align="center"><th>Nome: </th>
       <td><jsp:getProperty name="produto"
           property="nome"/></td></tr>
<tr align="center"><th>Valor: </th>
       <td><jsp:getProperty name="produto"
           property="valor"/></td></tr>
<tr align="center"><th>Quantidade: </th>
       <td><jsp:getProperty name="produto"
           property="quantidade"/></td></tr>
</table>
</body></html>
<jsp:setProperty name="produto"
property="codigo" value="2" />
```



a) primeira vez que a página é carregada



b) segunda vez que a página é carregada

8. Exemplo de Coleções.

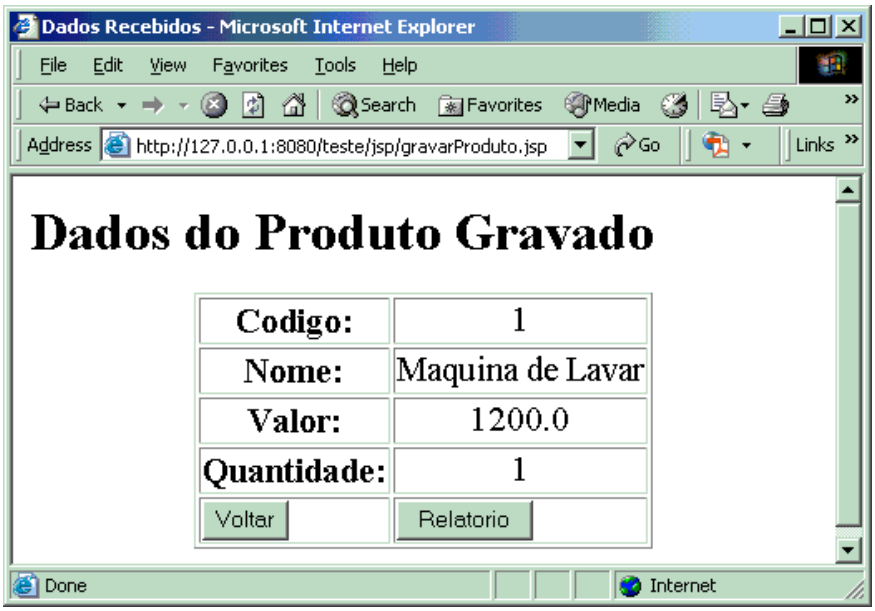
Arquivo de Cadastro de Produto (Cadastro.html).

```
<html>
<body>
<h1> Cadastro de Produtos </h1>
<center>
<form action="..\jsp\gravarProduto.jsp" method="post" border=1>
<table border=1 align="center">
  <tr bgcolor="silver"> <th> codigo: </th> <td>
       <input type="text" name="codigo" size=8 maxlength=8>
        </td></tr>
  <tr bgcolor="white"> <th> nome: </th> <td>
       <input type="text" name="nome" size=80 maxlength=250>
        </td></tr>
  <tr bgcolor="silver"> <th> valor: </th> <td>
       <input type="text" name="valor" size=8 maxlength=8>
        </td></tr>
  <tr bgcolor="White"> <td colspan="2" align="center">
        <input type="submit" value="Gravar" > </td></tr>
</table>
</body></html>
```

Prof. Vida.
vida@utfpr.edu.br

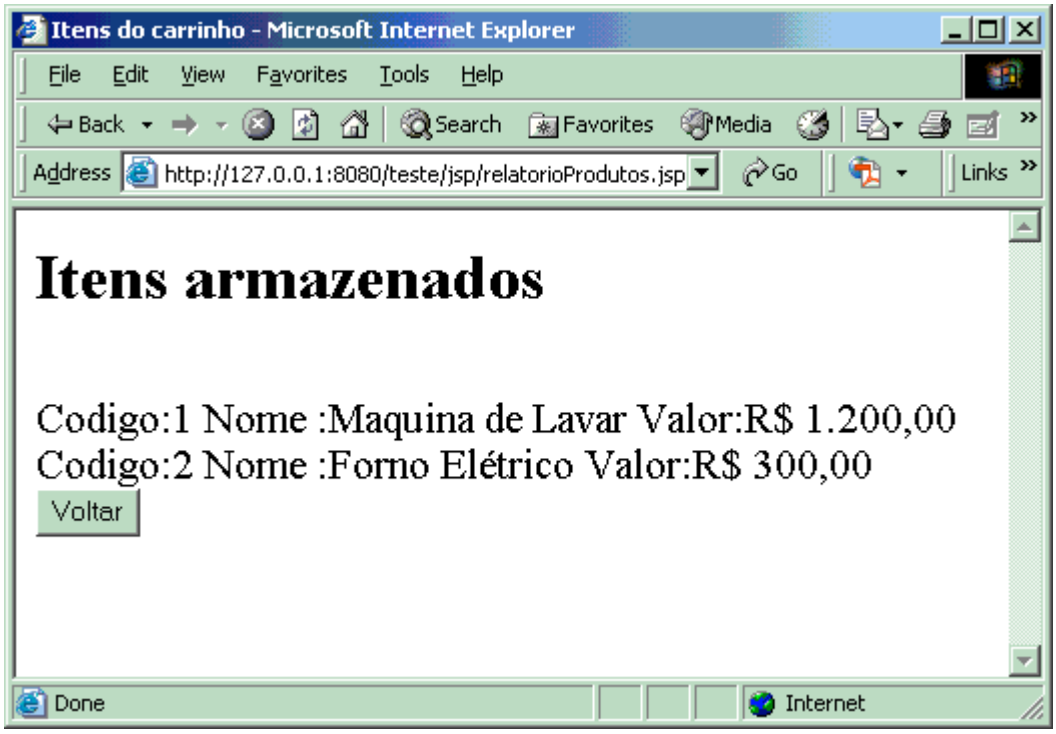
Arquivo de Gravação do Produto (gravarProduto.jsp).

```
<jsp:useBean id="lista" class="java.util.Vector" scope="session"/>
<jsp:useBean id="produto" class="javaNaWeb.mercado.Produto"/>
<jsp:setProperty name="produto" property="*" />
<% lista.add(produto); %>
<html> <head><title> Dados Recebidos </title></head>
<body>
<h2> Dados do Produto Gravado </h2>
<center><table border=1>
<tr align="center"><th>Codigo: </th>
 <td><jsp:getProperty name="produto" property="codigo"/></td></tr>
<tr align="center"><th>Nome: </th>
  <td><jsp:getProperty name="produto" property="nome"/></td></tr>
<tr align="center"><th>Valor: </th>
  <td><jsp:getProperty name="produto" property="valor"/></td></tr>
<tr align="center"><th>Quantidade: </th>
 <td><jsp:getProperty name="produto"
                property="quantidade"/></td></tr>

<tr><td><input type="button" value="Voltar"
onclick="javascript:history.back(1)"/></td>
    <td><form action = "relatorioProdutos.jsp" method=”post”>
     <input type="submit" value="Relatorio"/></form></td></tr>
</table></body></html>
```



Arquivo de Relatório do Carrinho de Compras (relatorioProdutos.jsp)

```
<jsp:useBean id="lista" class="java.util.Vector"
scope="session"/>

<html>
<head><title> Itens do carrinho </title></head>
<body>
  <h2> Itens armazenados </h2>
 <%
    for (int i =0 ; i < lista.size(); i++)
        out.println("<br>" + lista.get(i));
 %>
<br>
<input type="button" value="Voltar"
onclick="javascript:history.back(1)"/>
</body>
</html>
```



É interessante notar que a classe Vector não é um JavaBean, tendo em vista que não segue todas as convenções de sua especificação. Mas a tag `<jsp:useBean>` pode ser utilizada para criar uma instância de qualquer classe Java que contenha um contrutor sem parâmetros.