Java Web com Servlets e JSPs

JavaBeans, EL e JSTL

SoftBlue
www.softblue.com.br

Tópicos Abordados

- Java Beans
  - Características
  - Métodos getters e setters
  - Trabalhando com propriedades
- EL
  - O que é
  - Operadores "[]"e "."
  - Objetos implícitos
- JSTL
  - O que é
  - Como configurar
  - Algumas tags importantes

Evite o Uso de Código Java em JSP

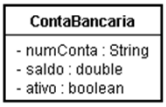
- Apesar de JSPs terem sido criados para possibilitarem a mistura de HTML e código Java, escrever código Java em JSPs não é uma boa prática
  - Dificulta o trabalho de web designers, que não são programadores
  - Para páginas complexas, o código fica confuso
  - Dificuldade de manutenção

Alternativas aos Scriptlets

- JavaBeans
- EL (**E**xpression **L**anguage)
- JSTL (**J**ava **S**erver Pages Standard **T**ag **L**ibrary)

JavaBeans

- É uma especificação Java que define um padrão de classe
- Uma classe é um JavaBean se:
  - Possui um construtor público sem argumentos
  - Possui métodos getters e/ou setters definidos corretamente



Assinatura dos Getters e Setters

- A assinatura dos getters e setters segue um padrão

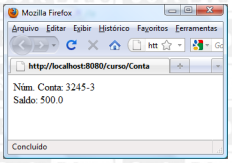
Atributo	Getter	Setter
numConta	getNumConta()	setNumConta()
saldo	getSaldo()	setSaldo()
ativo	isAtivo()	setAtivo()

Para atributos booleanos, o padrão do getter é **isXXX()**, mas **getXXX()** também pode ser utilizado

Lendo as Propriedades do Bean

Servlet

```
...
ContaBancaria c = new ContaBancaria();
c.setNumConta("3245-3");
c.setSaldo(500.0);
c.setAtivo(true);

request.setAttribute("conta", c);
...
```



JSP

```
<jsp:useBean id="conta" class="model.ContaBancaria" scope="request" />

<html>
<body>
Núm. Conta: <jsp:getProperty name="conta" property="numConta" />
<BR>
Saldo: <jsp:getProperty name="conta" property="saldo" />
</body>
</html>
```

<jsp:useBean>

```
<jsp:useBean
    id="conta"
    class="model.ContaBancaria"
    scope="request"
/>
```

id
Nome pelo qual o bean será referenciado. Equivale ao nome do atributo.

class
Fully qualified name da classe do bean instanciada na memória.

scope
Escopo de onde o bean é carregado. O default é page.

Se o bean não existir, o *<jsp:useBean>* cria um novo bean

<jsp:getProperty>

```
<jsp:getProperty
    name="conta"
    property="numConta"
/>
```

name
Nome do bean, definido pelo *id* em *<jsp:useBean>*.

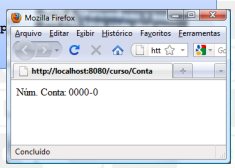
property
Nome da propriedade para leitura. Será chamado o método getter correspondente (*getNumConta()*).

Alterando as Propriedades do Bean

- Além de ler as propriedades de um JavaBean, é possível também alterá-las

JSP

```
<jsp:useBean id="conta" class="model.ContaBancaria" scope="request" />

<jsp:setProperty name="conta" property="numConta" value="0000-0" />

<html>
<body>
Núm. Conta: <jsp:getProperty name="conta" p
</body>
</html>
```



<jsp:setProperty>

```
<jsp:setProperty
    name="conta"
    property="numConta"
    value="0000-0"
/>
```

name
Nome do bean, definido pelo *id* em *<jsp:useBean>*.

property
Nome da propriedade que será alterada. Será chamado o método setter correspondente (*setNumConta()*).

value
Novo valor para a propriedade do bean.

<jsp:setProperty>

- É possível definir um **<jsp:setProperty>** dentro da tag **<jsp:useBean>**
  - Neste caso as propriedades só serão alteradas se o bean estiver sendo criado pelo **<jsp:useBean>**

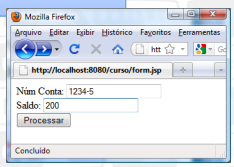
```
<jsp:useBean id="conta" class="model.ContaBancaria" scope="request">
  <jsp:setProperty name="conta" property="numConta" value="0000-0" />
</jsp:useBean>
```

O *<jsp:setProperty>* será ignorado caso o bean já exista na request

<jsp:setProperty>

- A tag **<jsp:setProperty>** também pode ser usada para alterar propriedades de um bean de acordo com informações vindas da request

```
<form action="conta.jsp">
Núm Conta: <INPUT type="text" name="numConta"><BR>
Saldo: <INPUT type="text" name="saldo"><BR>
<INPUT type="submit" value="Processar">
</form>
```

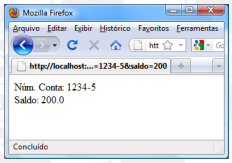


<jsp:setProperty>

conta.jsp

```
<jsp:useBean id="conta" class="model.ContaBancaria" scope="request" />
<jsp:setProperty name="conta" property="numConta" />
<jsp:setProperty name="conta" property="saldo" />

<html>
<body>
Núm. Conta: <jsp:getProperty name="conta" property="numConta" />
<BR>
Saldo: <jsp:getProperty name="conta" property="saldo" />
</body>
</html>
```

Se *value* não for definido, os parâmetros da request são pesquisados



<jsp:setProperty>

- É possível buscar todos os parâmetros da request que possuem os nomes iguais às propriedades do bean

```
<jsp:setProperty name="conta" property="*" />
```

- Se o parâmetro da request e a propriedade do bean tiverem nomes diferentes, é possível usar *param*

```
<jsp:setProperty name="conta" property="numConta" param="nc" />
```

Expression Language

- EL permite ainda mais facilidade na hora de ler informações presentes em um escopo

Servlet

```
request.setAttribute("user", "Carlos");
```

JSP

```
${user}
```

O resultado é "*Carlos*"

EL e JavaBeans

- Ler propriedades de JavaBeans é muito mais fácil usando EL

Servlet

```
ContaBancaria conta = new ContaBancaria();
conta.setNumConta("1234-5");
conta.setSaldo(400.0);
request.setAttribute("c", conta);
```

JSP

```
${c.numConta}
${c.saldo}
```

O resultado é "*1234-5*" e "*400.0*"

EL e JavaBeans

- Além do operador ".", existe o operador "[]"

JSP

```
${c.numConta}
${c.saldo}
```

JSP

```
${c["numConta"]}
${c["saldo"]}
```

Operador "[]" e Coleções de Dados

- O operador "[]" pode ser usado na presença de coleções de dados

c é um java.util.List

```
${c[1]}
${c["1"]}
```

Retorna a segunda posição da lista

c é um array

```
${c[1]}
${c["1"]}
```

Retorna a segunda posição do array

c é um java.util.Map

```
${c["Carlos"]}
${c.Carlos}
```

Retorna o valor do mapa cuja chave é "*Carlos*"

Limitações do Operador "."

- O operador "." só pode ser utilizado se o que estiver escrito do lado direito do operador for um identificador válido do Java
- Suponha que *c* é um *java.util.Map:*

```
${c.1}
```

Esta notação não funciona, pois "*1*" não é um identificador válido

```
${c[1]}
${c["1"]}
```

Esta notação funciona

Objetos Implícitos

Objeto	Descrição
pageScope	Map com os atributos do escopo page
requestScope	Map com os atributos do escopo request
sessionScope	Map com os atributos do escopo session
applicationScope	Map com os atributos do escopo application
param	Map com os parâmetros da request
paramValues	Map com os parâmetros da request
header	Map com o request HTTP header
headerValues	Map com o request HTTP header
cookie	Map com os cookies
initParam	Map com os context init parameters
pageContext	Referencia o objeto pageContext

JSTL

- JSTL é um conjunto de tag libraries que complementa as facilidades providas pela EL
  - As tag libraries definem ações
  - Substituem códigos Java nos arquivos JSP
- JSTL é bastante extensa
  - Core library
  - SQL library
  - Formatting library
  - XML library

Configurando o JSTL

- Para usar o JSTL na sua aplicação, são necessários dois arquivos JAR no classpath
  - jstl-api-XX.jar
  - jstl-impl-XX.jar
- Os arquivos podem ser encontrados na página do projeto do JSTL
  - http://jstl.java.net

Configurando o JSTL

- É necessário referenciar a URI do JSTL para que você possa usar as taglibs

<c:forEach>

- Permite executar um loop em uma lista de elementos

Servlet

```
List<String> lista = new ArrayList<String>();
lista.add("laranja");
lista.add("leite");
lista.add("margarina");

request.setAttribute("listaCompras", lista);
```

<c:forEach>

JSP

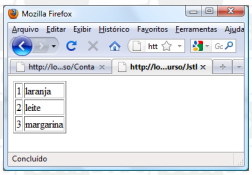
```
<%@ taglib prefix="c"
          uri="http://java.sun.com/jsp/jstl/core" %>

<table border="1">
<c:forEach var="item" items="${listaCompras}">
   <tr><td>${item}</td></tr>
</c:forEach>
</table>
```

A variável só existe dentro da tag

<c:forEach>

JSP

```
<table border="1">
<c:forEach var="item" items="${listaCompras}" varStatus="st">
  <tr>
    <td>${st.count}</td>
    <td>${item}</td>
  </tr>
</c:forEach>
</table>
```



<c:if>

- Permite testar uma determinada condição

Servlet

```
request.setAttribute("valor", 100);
```

JSP

```
<c:if test="${valor > 50}">
  O valor é maior que 50!
</c:if>
<c:if test="${valor < 50}">
  O valor é menor que 50!
</c:if>
<c:if test="${valor == 50}">
  O valor é maior que 50!
</c:if>
```

<c:choose>, <c:when>, <c:otherwise>

- Testam diversas condições de forma agrupada
- Apenas um bloco é executado

Servlet

```
request.setAttribute("tipoUsuario", "admin");
```

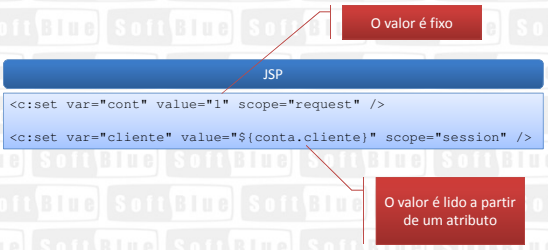
<c:choose>, <c:when>, <c:otherwise>

JSP

```
<c:choose>
  <c:when test="${tipoUsuario == 'admin'}">
    Bom dia, usuário administrador!
  </c:when>
  <c:when test="${tipoUsuario == 'gerente'}">
    Bom dia, usuário gerente!
  </c:when>
  <c:otherwise>
    Bom dia, usuário desconhecido!
  </c:otherwise>
</c:choose>
```

<c:set>

- Permite definir uma variável em um determinado escopo



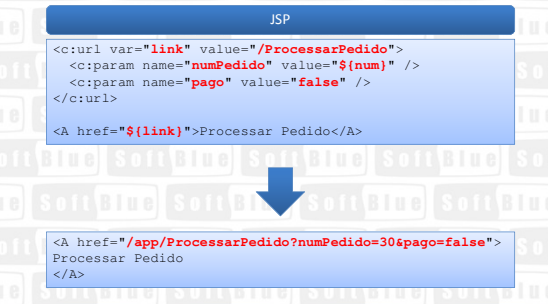
<c:set>

- Esta tag também pode ser usada para popular a propriedade de um bean

- E também de um *java.util.Map*

<c:url>

- Permite criar links



Etc, etc, etc...

- O JSTL é um conjunto amplo de tag libraries
- Para maiores informações, consulte a documentação no site oficial

Colocando em Prática...

Agora que você já aprendeu a teoria, acesse as vídeo-aulas práticas e pratique os assuntos abordados neste módulo!

Clique aqui para acessar as vídeo-aulas práticas