



1

---

---

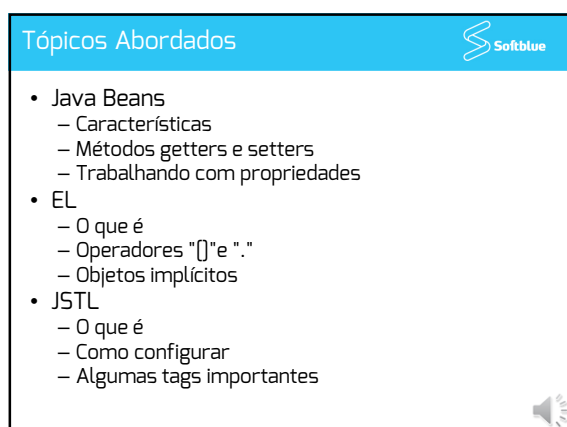
---

---

---

---

---



2

---

---

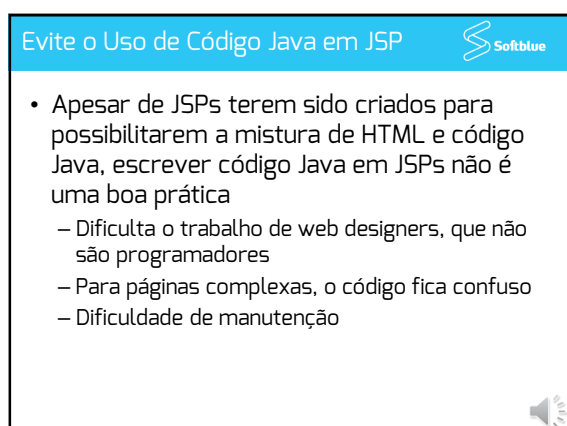
---

---

---

---

---



3

---

---

---

---

---

---

---

Alternativas aos Scriptlets

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

4

---

---

---

---

---

---

---

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

ContaBancaria

- numConta : String
- saldo : double
- ativo : boolean

5

---

---

---

---

---

---

---

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

6

---

---

---

---

---

---

---

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

```

7

---

---

---

---

---

---

---

---

## <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

8

---

---

---

---

---

---

---

---

## <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()).

9

---

---

---

---

---

---

---

---

## 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" />
</body>
</html>

```

10

## <jsp:setProperty>

```

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

```

name	property	value
Nome do bean, definido pelo <code>id</code> em <code>&lt;jsp:useBean&gt;</code> .	Nome da propriedade que será alterada. Será chamado o método setter correspondente ( <code>setNumConta()</code> ).	Novo valor para a propriedade do bean.

11

## <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

12

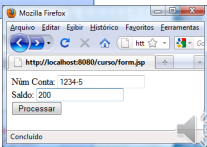
<jsp:setProperty>


- 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>

```



13

---

---

---


---

---

---

---

---

<jsp:setProperty>


conta.jsp

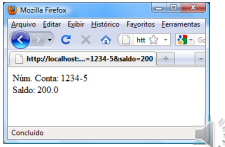
```

<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



14

---

---

---


---

---

---

---

---

<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" />

```



15

---

---

---

---

---

---

---

---

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"

16

---

---

---

---

---

---

---

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"

17

---

---

---

---

---

---

---

EL e JavaBeans

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

JSP

`<%= c.numConta %>`  
`<%= c.saldo %>`

JSP

`<%= c["numConta"] %>`  
`<%= c["saldo"] %>`

18

---

---

---

---

---

---

---

## Operador "[]" e Coleções de Dados



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

```
graph TD; A[c é um java.util.List] --> B["$ {c[1]}  
$ {c[*1]}"]; B --> C[Retorna a segunda posição da lista]; C --> D[c é um array]; D --> E["$ {c[1]}  
$ {c[*1]}"]; E --> F[Retorna a segunda posição do array]; F --> G[c é um java.util.Map]; G --> H["$ {c[\"Carlos\"]}  
$ {c.Carlos}"]; H --> I[Retorna o valor do mapa cuja chave é \"Carlos\"];
```

The diagram illustrates the use of the `get` method in Java for three different data structures: `java.util.List`, `array`, and `java.util.Map`. Each example shows the variable type, the code snippets for accessing elements, and the resulting output.

- Example 1:** `c` is a `java.util.List`. The code snippets are `$ {c[1]}` and `$ {c[*1]}`. The output is "Retorna a segunda posição da lista".
- Example 2:** `c` is an `array`. The code snippets are `$ {c[1]}` and `$ {c[*1]}`. The output is "Retorna a segunda posição do array".
- Example 3:** `c` is a `java.util.Map`. The code snippets are `$ {c["Carlos"]}` and `$ {c.Carlos}`. The output is "Retorna o valor do mapa cuja chave é \"Carlos\"".

19

---

---

---

---

---

---

## 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*:

<code>\$(c.1)</code>	Esta notação não funciona, pois "1" não é um identificador válido
<code>\$(c[1])</code> <code>\$(c["1"])</code>	Esta notação funciona

20

---

---

---

---

---

---

## 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

21

---

---


---

---


---

---

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



22

---

---

---


---

---


---

---

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>



23

---

---

---


---

---


---

---


Configurando o JSTL



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



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



24

---

---

---

---

---

---

---



<c:forEach>

Softblue

- 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);

```

25

---

---

---

---

---

---

---

<c:forEach>

Softblue

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>

```

Mozilla Firefox

A variável só existe dentro da tag

26

---

---

---

---

---

---

---

<c:forEach>

Softblue

JSP

```

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

```

Mozilla Firefox

27

---

---


---

---

---

---

---

<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>
```

28

---

---

---


---

---

---

---

---

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


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

Servlet

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

29

---

---

---


---

---

---

---

---

<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>
```

30

---

---

---

---

---

---

---

---

<c:set>

Softblue

- Permite definir uma variável em um determinado escopo

JSP

```

<c:set var="cont" value="1" scope="request" />
<c:set var="cliente" value="${conta.cliente}" scope="session" />

```

O valor é fixo

O valor é lido a partir de um atributo

31

---

---

---

---

---

---

---

<c:set>

Softblue

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

```

<c:set target="${conta}" property="numConta" value="1234-5" />

```

Bean

Propriedade

Valor

- E também de um java.util.Map

```

<c:set target="${clientes}" property="34" value="Carlos" />

```

java.util.Map

Chave

Valor

32

---

---

---

---

---

---

---

<c:url>

Softblue

- Permite criar links

JSP

```

<c:url var="link" value="/ProcessarPedido">
  <c:param name="numPedido" value="${num}" />
  <c:param name="pago" value="false" />
</c:url>
<A href="${link}">Processar Pedido</A>

```

↓

```

<A href="/app/ProcessarPedido?numPedido=30&pago=false">
  Processar Pedido
</A>

```

33

---

---

---


---

---


---

---

Etc, etc, etc...



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



---

---

---

---

---

---

---