

JSTL: JSP Standard Tag Library

Tags de decisão
Tags de repetição
Tags de formatação



Tags da JSTL



Ações padrões e EL
são legais, mas e se eu
quiser iterar sobre
uma coleção para exibi-
la numa tabela HTML?

Tags da JSTL

■ Radiografia da JSTL 1.1

- Conjunto de biblioteca de tags prontas para uso
- Versão 1.1 incorporada à especificação de JSP 2.0
- Eliminam o uso de scripts dentro de páginas JSP
- Especificação pode (deve!) ser baixada em:
<http://java.sun.com/jsp/jstl> (veja link API Specification)
- Bibliotecas jar devem ser incorporadas à pasta **lib** de toda aplicação web (não automático)

- **jstl.jar**
- **standard.jar**

← No Tomcat 5.x pegue-as na aplicação **jsp-example**, pasta WEB-INF\lib

Tags da JSTL

■ Código em um servlet:

```
String[] cities = {"Jampa", "Maceio", "Natal"};  
request.setAttribute("cidades", cities);
```

■ Ao invés de usar scripts...



```
<table>  
<% String[] items =  
    (String[]) request.getAttribute("cidades");  
    String var=null;  
    for(int i=0; i<items.length ;i++) {  
        var=items[i];  
%> <tr><td><%=var%></td></tr>  
<% } //for %>  
</table>
```

Tags da JSTL


■ Código em um servlet:

```
String[] cities = {"Jampa", "Maceio", "Natal"};  
request.setAttribute("cidades", cities);
```

■ Que tal uma tag **<forEach>**?

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

```
<table>  
  <c:forEach var="cidade" items="${cidades}">  
    <tr>  
      <td>${cidade}</td>  
    </tr>  
  </c:forEach>  
</table>
```



Tags da JSTL

■ Anatomia de <forEach>

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

O prefixo pode ser qualquer um, mas **c** (de **core**) é usado universalmente! Não mude!

Identificador único da taglib

```
<table>  
  <c:forEach var="cidade" items="${cidades}">  
    <tr>  
      <td>${cidade}</td>  
    </tr>  
  </c:forEach>  
</table>
```

A variável **cidade** vai apontar para cada um dos elementos da coleção **\${cidades}**, em cada iteração do laço

Tags da JSTL

- Sintaxe de `<c:forEach>`, conforme a especificação:

```
<c:forEach [var="varName"] items="collection"  
           [varStatus="varStatusName"]  
           [begin="begin"] [end="end"] [step="step"]>  
    body content  
</c:forEach>
```

- Mais uma vez...

- Baixe a especificação em <http://java.sun.com/jsp/jstl>

Tags da JSTL

■ Condicionais:

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

```
<c:if test="${user eq 'admin'}">  
  <jsp:include page="viewpermissiona.jsp"/>  
</c:if>
```

↑
Não tem else

↑
Ooops! Que diabos é
isso nesta EL?

Um operador de EL!

Tags da JSTL

■ Aritméticos:

- Adição: +
- Subtração: -
- Multiplicação: *
- Divisão: / e **div**
- Resto: % e **mod**

```
${salario * 0.10}
```

■ Lógicos:

- E: && e **and**
- Ou: || e **or**
- Não: ! e **not**

```
${isAluno and aprovado}
```

Tags da JSTL

■ Relacional:

- Igual: `==` e **eq**
- Diferente: `!=` e **ne**
- Menor que: `<` e **lt**
- Maior que: `>` e **gt**
- Maior ou igual: `>=` e **ge**
- Menor ou igual: `<=` e **le**

```
${tipo == 'admin'}
```

↑
Chama o `equals()` de
String

```
${idade gt 35}
```

Tags da JSTL

■ Outra tag que lida com condicionais:

```
<c:choose>
  <c:when test="\${userType == 'simples'}">
    <!-- Algo exclusivo para usuários simples -->
  </c:when>
  <c:when test="\${userType == 'admin'}">
    <!-- Algo exclusivo para usuários admin -->
  </c:when>
  <c:otherwise>
    <!-- Ação Default -->
  </c:otherwise>
</c:choose>
```

Tags da JSTL

■ Uma tag para "setar" valores

```
<c:set var="nome" scope="request" value="Fred"/>
```

Nome de um
atributo



Valor
(não necessariamente String)



```
<c:set var="curso" value="${aluno.curso}"/>
```

- Se o atributo não existir no escopo indicado, ele será criado (exceto se o valor for null)
- Se o escopo não for informado, **page** é usado

Tags da JSTL


■ Outra sintaxe:

```
<c:set target="$ {uf}" property="PB" value="Jampa" />
```

Referência para um
objeto



Nome de uma propriedade
(bean) ou de uma chave
(mapa)



- Se target for null, uma exceção será lançada pelo container
- Se não for um bean ou mapa, uma exceção será lançada
- Se for um bean e a propriedade não existir, idem.

Tags da JSTL

■ Outras tags de **core**

- Remove um atributo de um escopo:

```
<c:remove var="aluno" scope="request"/>
```

- Incluir outro conteúdo:

```
<c:import url="http://baixa.da.egua"/>
```

- Parametrizado:

```
<c:import url="http://baixa.da.egua" >  
  <c:param name="legenda" value="..." />  
</c:import>
```

Tags da JSTL

■ Outras tags de **core**

- Codifica uma URL (uso em sessões):

```
<c:url value="/cadastraUsuario.jsp"/>
```

```
<a href="<c:url  
value='/cadastraUsuario.jsp' />">click!</a>
```

- Equivale, em um servlet a:

```
out.println("<a href=\"\" +  
response.encodeURL(url) ">click</a>");
```

Tags da JSTL

■ Parametrizando uma URL

```
<c:url value="/cadastraUsuario.jsp"/>  
  <c:param name="id" value="123">  
    <c:param name="cidade" value="João Pessoa">  
</c:url>
```

```
/app/cadastraUsuario.jsp?id=123&cidade=João+P  
essoa
```


Tags da JSTL

■ Outras bibliotecas disponíveis na JSTL:

■ I18N:

- `<fmt:message>`
- `<fmt:setLocale>`
- `<fmt:bundle>`
- `<fmt:setBundle>`
- `<fmt:param>`
- `<fmt:requestEncoding>`

■ Formatação:

- `<fmt:timeZone>`
- `<fmt:setTimeZone>`
- `<fmt:formatNumber>`
- `<fmt:parseNumber>`
- `<fmt:parseDate>`

■ XML:

- `<xml:parse>`
- `<xml:out>`
- `<xml:set>`

■ SQL:

- `<sql:query>`
- `<sql:update>`
- `<sql:setDataSource>`
- `<sql:param>`
- `<sql:dateParam>`

Tags da JSTL

■ Exemplos da tag fmt

```
<fmt:formatDate type="date" value="${dataCriacao}" />
```

```
<fmt:formatDate pattern="dd-mm-yyyy" value="${dataCriacao}" />
```

```
<fmt:formatNumber type="currency" value="3.977">
```

■ EL functions:

- `${fn:length(alunos)}` → tamanho da lista alunos
- `${fn:trim(nome)}` → elimina brancos do String nome
- `${fn:toLowerCase(nome)}` → converte para minúsculas

Cartão de referência de JSP e JSTL (muito bom!):

<http://cs.roosevelt.edu/eric/books/JSP/jstl-quick-reference.pdf>

Bibliotecas de tags customizadas

- Bibliotecas de tags não oficiais
 - Como instalar e usar?
 - Como implementar uma biblioteca de tags?
- **Tag Library Descriptor**
 - Arquivo XML que "descreve" a biblioteca
 - Para usar uma [taglib](#) você tem que conhecer seu TLD

Bibliotecas de tags customizadas

```
<taglib>
  <tlib-version>1.0</tlib-version>
  <short-name>MinhasTags</short-name>
  <uri>fcgp.tags</uri>

  <tag>
    <description>Apresenta um aluno</description>
    <name>showAluno</name>
    <tag-class>foo.AlunoTagHandler</tag-class>
    <body-content>empty</body-content>
    <attribute>
      <name>id</name>
      <required>true</required>
      <rtextpvalue>true</rtextpvalue>
    </attribute>
  </tag>
</taglib>
```

Obrigatórios

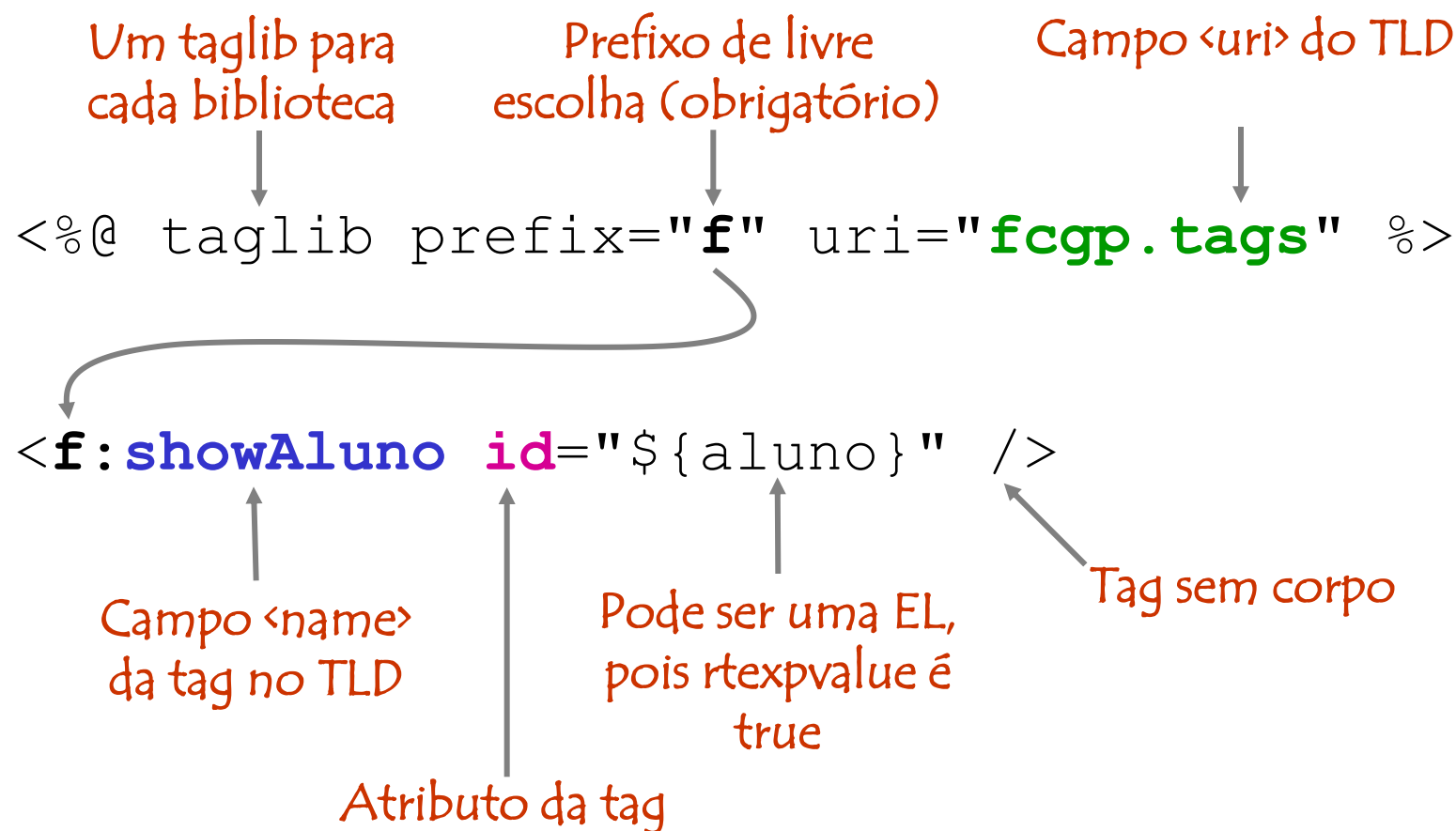
Usado na diretiva taglib

Tag sem corpo

Valor pode ser `<%=..%>` ou `${a.b}`

Bibliotecas de tags customizadas

■ Exemplo de uso desta tag:



Bibliotecas de tags customizadas

```
package foo;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.SimpleTagSupport;
import java.io.IOException;

public class AlunoTagHandler
    extends SimpleTagSupport {

    private Aluno a;

    public void doTag() throws JspException, IOException {
        ...
    }

    public void setId(Aluno id) { ... }
}
```

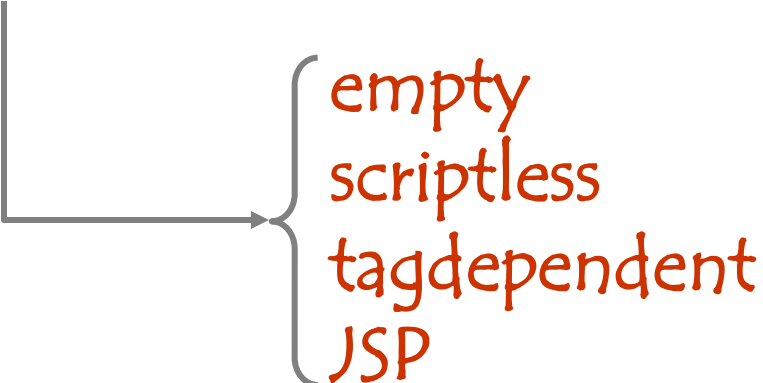
Chamado quando a tag é invocada

Chamado para atribuir o valor ao atributo

Bibliotecas de tags customizadas

■ O elemento `<body-content>` do TLD

```
<taglib>
...
<tag>
  <description>Apresenta um aluno</description>
  <name>showAluno</name>
  <tag-class>foo.AlunoTagHandler</tag-class>
  <body-content>empty</body-content>
  ...
</tag>
...
</taglib>
```



A diagram consisting of a vertical line and a horizontal arrow pointing from the `empty` value in the `<body-content>` tag to a large curly brace. Inside the brace, the following categories are listed in red text: `empty`, `scriptless`, `tagdependent`, and `JSP`.

Bibliotecas de tags customizadas

■ **empty**

- Não pode ter corpo nenhum

■ **scriptless**

- Não pode ter scripts (scriptlets, expressões ou diretivas), mas pode EL e ações

■ **tagdependent**

- O corpo da tag é tratado como texto puro, portanto EL e ações não são avaliadas

■ **JSP**

- O corpo pode ter qualquer elemento de JSP e EL

Bibliotecas de tags customizadas

- O que é uma tag com `<body-content>` **empty**?

- Tag só de fechamento:

`<f:showAluno id="${aluno}" />`



- Tag com abertura e fechamento mas sem corpo:

`<f:showAluno id="${aluno}"> </f:showAluno>`



- Tag com abertura e fechamento e `<jsp:attribute>`:

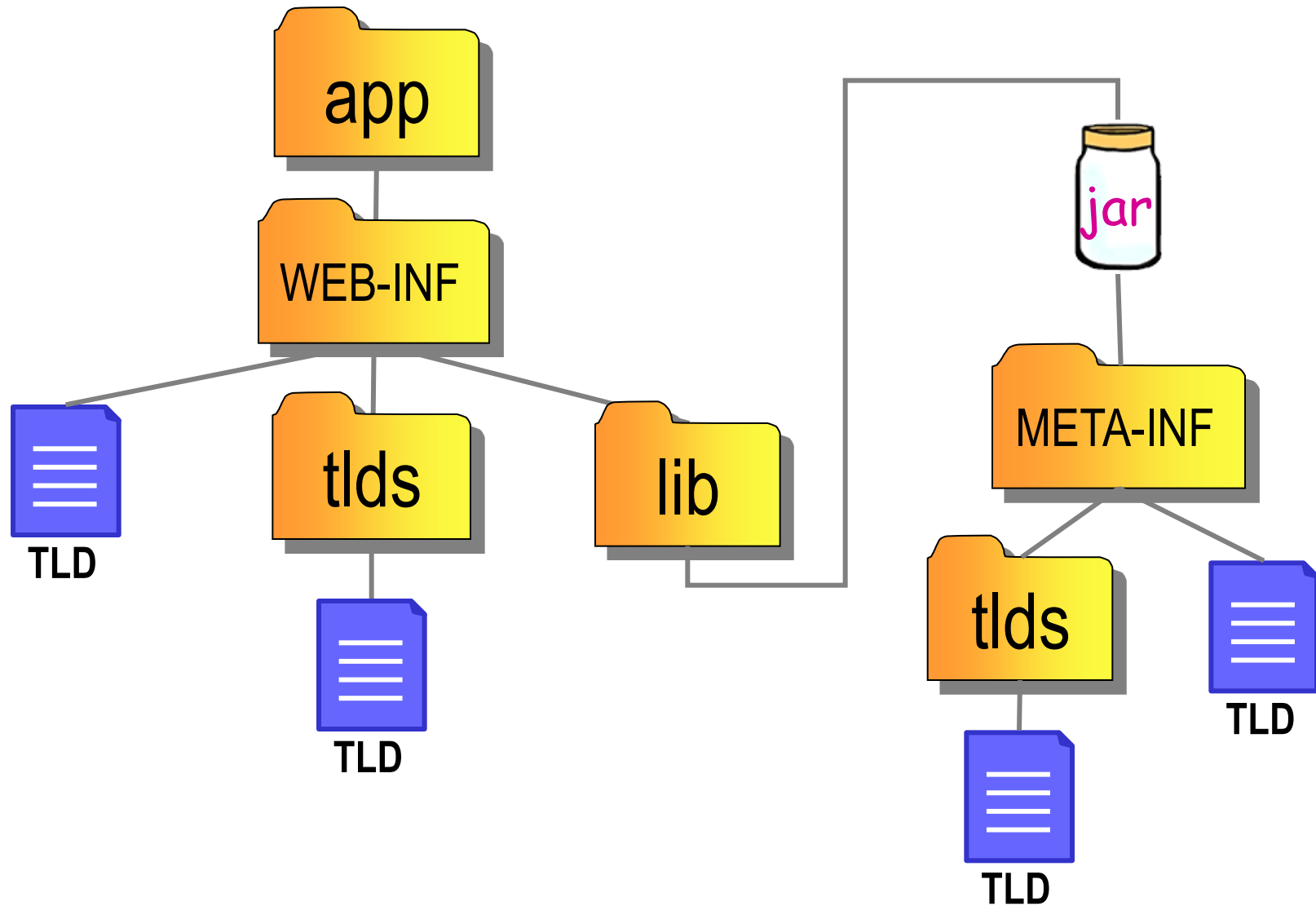
```
<f:showAluno>  
  <jsp:attribute name="id">${aluno}</jsp:attribute>  
</f:showAluno>
```

Bibliotecas de tags customizadas

- Como o container acha o arquivo do TLD?
 - Versão anterior a JSP 2.0, no DD:

```
<web-app>  
  <jsp-config>  
    <taglib>  
      <taglib-uri>fcgp.tags</taglib-uri>  
      <taglib-location>  
        /WEB-INF/fcgp.tld  
      </taglib-location>  
    </taglib>  
  </jsp-config>  
</web-app>
```

TLDs no JSP 2.0



Bibliografia

- **Bashan, B., Sierra, K. e Bates, B.** "Head First Servlets & JSP". Capítulo 9. 2005.