SERVLETS E JSP (JEE) - JSP - JSTL

Diogo Cezar Teixeira Batista

diogo@diogocezar.com.br

http://www.diogocezar.com

Universidade Tecnológica Federal do Paraná - UTFPR

Cornélio Procópio - 2012

JSP JSTL I

- JSTL - *JSP Standard Tag Library*
- qual o problema?
  - dificuldade de contruir páginas *JSPs* bem organizadas internamente;
  - páginas *JSPs* com muito código Java;
  - *web designers* não sabem programar em Java;
  - problemas na interação entre desenvolvedores e *web designers*.

JSP JSTL II

- o que queremos?
  - criar páginas dinâmicas bastante complexas sem escrever código Java dentro delas;
  - fornecer *tags* que tornem fáceis tarefas que exigiriam várias linhas de código Java, como formatação de números e datas seguindo configurações regionais do usuário;
  - facilitar a interação entre desenvolvedores e *web designers*.

JSP JSTL III

- qual a solução?
  - utilizar JSTL;

JSP JSTL IV

- JSTL
  - JSTL consiste em uma coleção de bibliotecas;
  - cada biblioteca tem um propósito bem definido;
  - permitem escrever páginas JSPs sem código Java;
  - aumentando a legibilidade do código e a interação entre desenvolvedores e *web designers*;
  - uma página JSTL é uma página JSP contendo um conjunto de tags JSTLs;
  - cada *tag* realiza um determinado tipo de processamento (equivalente a código Java dentro de JSP);
  - cada tag JSTL, faz parte uma biblioteca JSTL;
  - uma página JSTL pode utilizar várias bibliotecas JSTLs.

Código 1: Exemplo de uma página JSTL

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<html>
<body bgcolor="#FFFFFF">

<jsp:useBean id="agora" class="java.util.Date"/>
<br>
Versão Curta: <fmt:formatDate value="${agora}" />

<br> Versão Longa: <fmt:formatDate value="${agora}" dateStyle="full"↩
    />
</body>
</html>
```

JSP JSTL VI

- o exemplo apresenta a data atual em dois formatos: um curto e outro longo;
- observe que não existe nenhum código Java;
- na primeira linha, temos o seguinte: `<%@ taglib prefix="fmt"uri="http://java.sun.com/jsp/jstl/fmt"%>`;
- esta declaração informa ao compilador de JSPs utiliza um biblioteca de tags, cujos tags serão reconhecidos pelo prefixo "fmt", de modo a evitar possíveis conflitos com tags de mesmo nome de outras bibliotecas;
- a biblioteca de tags é identificada pelo atributo uri;
- a tag padrão `<jsp:useBean>` é utilizada para instanciar um objeto `java.util.Date` (inicializado por padrão com a data e hora atuais do sistema);

JSP JSTL VII

- a variável *agora*, criada pelo tag `<jsp:useBean>`, é depois referenciada dentro do atributo value dos tags `<fmt:formatDate>`, que aparece duas vezes na página;
- observe que o atributo `dateStyle` da tag `<fmt:formatDate>` define se será utilizado um formato resumido da data ou um formato longo.

Código 2: Saída de dados do exemplo anterior

```
Versão Curta: 25/07/2005
Versão Longa: Segunda-feira, 25 de Julho de 2005
```

JSP Expression Language I

- O valor de qualquer expressão pode ser acessado da seguinte forma: `$expressão`

JSP Expression Language II

Operador	Descrição	Exemplo	Resultado
== eq	Igualdade	${5 == 5}	true
!= ne	Desigualdade	${5 != 5}	false
< lt	Menor que	${5 < 7}	true
> gt	Maior que	${5 > 7}	false
<= le	Menor ou igual que	${5 le 5}	true
>= ge	Maior ou igual que	${5 ge 6}	false
empty	Checa se um parâmetro está vazio	${user.lastname}	depende
and &&	E	${param.month == 5 and param.day == 25}	depende
or ||	OU	${param.month == 5 or param.month == 6}	depende
0	soma	${4 + 5}	9
! not	Negação	${not true}	false

Bibliotecas Padrão

Biblioteca JSTL	Prefixo	URI	Tipos de uso	Exemplo de tag
Core	c	http://java.sun.com/jstl/core	Acessar e modificar dados em memória Comandos condicionais Loop	<c:forEach>
Processamento de XML	x	http://java.sun.com/jstl/xml	Parsing (leitura) de documentos Impressão de partes de documentos XML Tomada de decisão baseado no conteúdo de um documento XML	<x:forEach>
Internacionalização e formatação	fmt	http://java.sun.com/jstl/fmt	Leitura e impressão de números Leitura e impressão de datas Ajuda a sua aplicação funcionar em mais de uma lingua	<fmt:formatDate>
Acesso a banco de dados via SQL	sql	http://java.sun.com/jstl/sql	Leitura e escrita em banco de dados	<sql:query>

Tags básicas I

- tags de Iteração:
  - a biblioteca core do JSTL fornece tags para executar trechos repetidamente, de maneira similar aos comandos *for* e *while* da linguagem Java.
  - tag `<c:forEach>`
  - permite realizar um *loop*;
  - exemplo: imprime os valores entre dois e cinco.

Tags básicas II

Código 3: Exemplo da utlização de forEach

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

<c:forEach var="i" begin="2" end="5">
<c:out value="${i}" />;
</c:forEach>
</body>
</html>
```

- saída: 2;3;4;5;

Tags básicas III

- tag `<c:forTokens>`
- quebra uma *string* em substrings, de acordo com o delimitador indicado como atributo;
- exemplo: imprime os valores entre dois e cinco;

Tags básicas IV

Código 4: Exemplo da utlização de forTokens

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body bgcolor="#FFFFFF">
<c:forTokens var="i" delims="," items="2,3,4,5">
  <c:out value="${i}" />;
</c:forTokens >
</body>
</html>
```

- saída: 2;3;4;5;

Tags básicas V

- tags condicionais:
  - tag `<c:if>`
  - equivamente ao comando *if* ;
  - atributo test realiza o teste condicional;
  - não existe o complemento do comando *if*, ou seja, o comando *else* (caso se deseje criar fluxos alternativos, deve-se utilizar `<c:choose>`).

Tags básicas VI

Código 5: Exemplo da utlização de if

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<html>
<body bgcolor="#FFFFFF">
Elementos pares:
<c:forTokens var="i" delims="," items="2,3,4,5">
<c:if test="${i % 2 == 0}">
<c:out value="${i}" />;
</c:if>
</c:forTokens >
</body>
</html>
```

- saída: Elementos pares: 2; 4;

Tags básicas VII

- tag `<c:choose>`
- equivalente ao comando *switch*
- tags utilizadas:
  - `<c:when>`, realiza o teste condicional;
  - `<c:otherwise>`, se todos os testes condicionais falharem, ele será utilizado;

Tags básicas VIII

Código 6: Exemplo da utlização de choose

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

<c:forTokens var="i" delims="," items="2,3,4,5">
  <c:choose>
<c:when test="${i % 2 == 0}">${i} (par)</c:when>
<c:otherwise>${i} (impar)</c:otherwise>
  </c:choose>
  ;
</c:forTokens >
</body>
</html>
```

- saída: 2 (par) ; 3 (impar) ; 4 (par) ; 5 (impar) ;

Tags básicas IX

- tags de atribuição e importação:
  - tag `<c:import>`
  - permite importar páginas *web* do mesmo contexto *web*, de contextos diferentes e até mesmo de máquinas diferentes;

Atributo	**Descrição**	**Requerido?**	**Default**
url	URL a ser importada	Sim	Nenhum
context	"/" seguido do nome da aplicação web local	Não	Contexto corrente
var	Nome do atributo onde será armazenado o conteúdo da página importada	Não	Nenhum
scope	Escopo do atributo onde será armazenado o conteúdo da página importada Pode ser: page, request, session, application	Não	page

Tags básicas X

- tag `<c:set>`
- permite a atribuir valores a variáveis em um determinado escopo;

Atributo	Descrição	Requerido?	Default
value	Expressão a ser processada	Não	Nenhum
var	Nome do atributo onde será armazenado o resultado do processamento do atributo "value"	Não	Nenhum
scope	Escopo do atributo. Pode ser: page, request, session, application	Não	page

Tags básicas XI

- exemplo:
  - neste exemplo, a variável “title” é criada com o valor “Welcome to Page 1” com escopo “request”;
  - em seguida, a página “header.jsp” é carregada, e seu conteúdo é armazenado na variável “headerText”;
  - finalmente, imprimimos o conteúdo da variável “title” e da variável “headerText”

Código 7: Exemplo da set e import

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<html>
<body>
<c:set scope="request" var="title" value="Welcome to Page 1" />
<c:import var="headerText" url="header.jsp" />
<br>Minha página:${title}
<br>Texto importado:${headerText}
</body>
</html>
```

Tags básicas XII

Código 8: Arquivo header.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
#${title}#
```

Código 9: Saída

```
Minha página:Welcome to Page 1
Texto importado: #Welcome to Page 1#
```

Acessando objetos Java I

- exemplo 1: acessando uma coleção de objetos Java;
- seja a classe Java ColecaoDeNomes;

Código 10: Classe Java ColecaoDeNomes

```
package test;
import java.util.*;
import java.io.*;
public class ColecaoDeNomes implements Serializable{
  private Collection nomes = new ArrayList();
  public ColecaoDeNomes() {
    nomes.add("Maria");
    nomes.add("Zeca");
    nomes.add("Carlos");
  }
  public Collection getNomes() {
    return nomes;
  }
}
```

Acessando objetos Java II

Código 11: Utilizando a lista ColecaoDeNomes

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body bgcolor="#FFFFFF">
<jsp:useBean id="colecao" class="teste.ColecaoDeNomes"/>
<c:forEach var="nome" items="${colecao.nomes}">
  <br>${nome}
</c:forEach >
</body>
</html>
```

Acessando objetos Java III

- queremos iterar a coleção de nomes existente dentro da classe *ColecaoDeNomes*;
- observe que o objeto da classe *ColecaoDeNomes* foi instanciado utilizando a tag `<jsp:useBean>`;
- além disso, o nome da instância criada é "colecao";
- note que o atributo items da *tag* `<c:forEach>` faz referência à instância criada (batizada como "colecao");

Acessando objetos Java IV

- os objetos existentes dentro da coleção nomes (`java.util.Collection`) dentro da classe *ColecaoDeNomes* estão sendo acessado da seguinte forma: `$colecao.nomes`
- `$colecao.nomes`: colecao é o nome da instância da classe *ColecaoDeNomes* recém criada
- `$colecao.nomes`: nomes faz referência ao nome do método getNomes()

Código 12: Saída

```
Maria
Zeca
Carlos
```

Acessando objetos Java V

- exemplo 2: agora utilizando mais recursos da tag `<c:forEach>`

Acessando objetos Java VI

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<html>
  <body bgcolor="#FFFFFF">
  <jsp:useBean id="colecao" class="teste.ColecaoDeNomes" />
    <table border="0" cellpadding="2" cellspacing="2">
    <tr>
    <td bgcolor="#e0e0e0">Id</td>
    <td bgcolor="#e0e0e0">Nome</td>
    </tr>
    <c:forEach var="nome" items="${colecao.nomes}"
     varStatus="status">
    <tr>
    <td bgcolor="#f0f0f0">
    <c:choose>
    <c:when
     test="${status.first}">Primeiro</c:when>
    <c:when
     test="${status.last}">Último</c:when>
    <c:otherwise>Número
     ${status.count}</c:otherwise>
    </c:choose>
    </td>
    <td bgcolor="#f0f0f0">${nome}</td>
    </tr>
    </c:forEach >
    </table>
  </body>
</html>
```

Atividades

- Implemente as seguintes funcionalidades utilizando JSTL:
  - Laço de 1 a 37 que mostre apenas os números primos;
  - Laço de 1 a 45 que mostre os números divisíveis por 5;
  - Laço que mostre todas as letras do alfabeto;
  - Laço percorra todas as letras do alfabeto e mostre somente as vogais;
  - Utilizando forEach, percorra uma lista de pessoas e mostre apenas os nomes que começam com a letra a, ou tenham a letra b no nome;