Introdução a Java Server Pages

Jomi Fred Hübner
jomi@inf.furb.br
FURB / DSC

Novembro - Dezembro, 2003

Hello World

```
<%@page contentType="text/html; charset=ISO-8859-1"%>
<html>
<head><title>Oi</title></head>
<body>
Olá Mundo!
</body>
</html>
```

Exemplo de um **Somador**

Formulário:

```
<HTML> <BODY>
  Formulário para Soma

  <FORM action="soma.jsp" method="post">

    <INPUT name='vl1' size='5' value='2'/> +
    <INPUT name='vl2' size='5' value='2'/>
    <INPUT name='submit' type='submit' value='Soma' />
  </FORM>
</BODY></HTML>
```

Processador JSP **soma.jsp**:

```
<%@page contentType="text/html; charset=ISO-8859-1"%>
<html><body>

<%
int vl1=Integer.parseInt(request.getParameter("vl1"));
int vl2=Integer.parseInt(request.getParameter("vl2"));
int soma=vl1+vl2;
%>

Resultado = <b> <%=soma%> </b>

</body></html>
```

JSP = html + java

Ciclo de **Vida** de uma página JSP

1. **Compilação**: o servidor gera um **servlet** se o arquivo JSP é mais novo que o servlet
   - um JSP é compilado
   - o desenvolvedor não altera o servlet, mas o JSP

2. **Execução**: quando uma página JSP é requisitada a primeira vez, o servidor
   - carrega a classe do servlet correspondente
   - instância um objeto desta classe
   - invoca o método `jspInit`
   - "executa" o servlet

   Na segunda chamada, somente a execução acontece.

Marcações específicas: **expressões**

<%= ... %>

```
<%@page contentType="text/html; charset=ISO-8859-1"%>
<html><body>

Raiz de nove é <%= Math.sqrt(9) %>

</body></html>
```

Marcações específicas: **scriptlets**

<% ... %>

```
...
<%@page import="java.util.*" %>
...
<%
   int hoje =
     new GregorianCalendar().get(Calendar.DAY_OF_WEEK);
   if (hoje == 6) {
%>
       hoje é sexta
<% }%>
...
```

Marcações específicas: **declarações**

<%! ... %>

```
...
<%!
// declaração de um atributo para o JSP
int cont = 0;
%>
...
O valor do contador é <%= cont++ %>
...
```

```
...
<%!
// declaração de um método para o JSP
String maiusculo(String s) {
  return s.toUpperCase();
}
%>
...
xml maiúsculo é <%= maiusculo("xml") %>
...
```

Obtenção de **parâmetros** vindos da requisição

```
método (get/post) = <%=request.getMethod()%>
ip remoto         = <%=request.getRemoteAddr()%>
host remoto       = <%=request.getRemoteHost()%>
browser           = <%=request.getHeader("user-agent")%>

<% Enumeration i = request.getParameterNames();
   while (i.hasMoreElements()) {
      String parId = i.nextElement().toString();
      String parVl = request.getParameter(parId);
%>
      parId= <%=parId%> , valor= <%=parVl%> <br/>
<% } %>
```

Entendendo o exemplo da **soma**

```
<%@page contentType="text/html; charset=ISO-8859-1"%>
<html><body>

<%
int vl1=Integer.parseInt(request.getParameter("vl1"));
int vl2=Integer.parseInt(request.getParameter("vl2"));
int soma=vl1+vl2;
%>

Resultado = <b> <%=soma%> </b>

</body></html>
```

Exercícios

- Rodar o exemplo **oi** e **soma**
- Rodar o exemplo que mostra as informações de `request`

Uso de **folhas de estilo** (advanced HelloWorld)

Alguns **imports**

```
<%@ page contentType="text/html; charset=ISO-8859-1"%>

<%@ page import = "org.xml.sax.*" %>
<%@ page import = "org.xml.sax.helpers.*" %>
<%@ page import = "org.w3c.dom.*" %>
<%@ page import = "javax.xml.parsers.*" %>
<%@ page import = "javax.xml.transform.*" %>
<%@ page import = "javax.xml.transform.dom.*" %>
<%@ page import = "javax.xml.transform.stream.*" %>
```

Inicialização do JSP

- para não criar um transformador a cada requisição, é criado um na inicialização do JSP

```
<%!
DocumentBuilder builder;  // o parser para DOM
Transformer     transHTML; // o transformador para HTML

public void jspInit() {
    // cria uma fábrica de parsers DOM
    DocumentBuilderFactory factory =
       DocumentBuilderFactory.newInstance();
    builder = factory.newDocumentBuilder();
```

```
        // Cria o transformador XML -> HTML
        String arqXSL =
           getServletContext().getRealPath(".") +
           "/oi/oiHTML.xsl";

        TransformerFactory tFactory =
           TransformerFactory.newInstance();

        transHTML =
           tFactory.newTransformer(new StreamSource(arqXSL));

    } // do jspInit
    %>
```

Aplicação da folha

```
<%
  String arqOi =
      getServletContext().getRealPath(".") +
      "/oi/oi.xml";
  Document docDOM =
      builder.parse(arqOi);
  transHTML.transform(
      new DOMSource( docDOM ),
      new StreamResult(out) );
}
%>
```

Inclusão e redirecionamento

- Uma página JSP pode **incluir** o resultado gerado por outra página

```
antes
<jsp:include page="oiAdvanced.jsp" />
depois
```

- Uma página JSP também pode repassar o "controle" para outra página:

```
antes
<jsp:forward page="oiAdvanced.jsp" />
depois
```

Java **Beans**

- Um java bean permite reutilizar outros objetos e compartilha-lo entre várias páginas.

```
<jsp:useBean id="lea"
             scope="application"
             class="lea.SistemaLEA" />
...
<%= lea.getProfessor("McLopes").getNome() %>
```

Valores de **escopo**

- **request**: o java bean existe somente para a requisição corrente
- **session**: o java bean pertence à sessão criado para o usuário
- **page**: o java bean é compartilhado por todas as execuções da página (mesmo de usuários diferentes)
- **application**: o java bean é compartilhado por toda a aplicação (todas as páginas, todas as sessões, ...)

Envio de **e-mail**

```
// seta o servidor de email
Properties props = new Properties();
props.put("mail.smtp.host", "penha.inf.furb.br");

// cria uma sessão com o servidor de email
Session mailSession =
    Session.getDefaultInstance(props, null);
mailSession.setDebug(false);

// cria a mensagem
Message msg = new MimeMessage(mailSession);
// assunto
msg.setSubject("Super oi");
```

```
// seta from e to
InternetAddress from =
    new InternetAddress("jomi@inf.furb.br");
msg.setFrom(from);

InternetAddress[] address =
  { new InternetAddress("jomi@inf.furb.br") };
msg.setRecipients(Message.RecipientType.TO, address);

// conteúdo
msg.setContent("oi", "text/plain");

// envia
Transport.send(msg);
```

Obs.: `mail.jar` e `activation.jar` devem estar no **classpath** ou, no caso de JSP, no `WEB-INF/lib`.

Como executar sem o NetBeans

- Selecionar o WebModule e a opção export WAR file
- Colocar o arquivo WAR no diretório `webapps` do **tomcat**.

Exercício

- Fazer uma página JSP que mostra o relatório de todas as reservas.

  $$\text{BD} \rightarrow \text{OO} \rightarrow \text{XML} \rightarrow (\text{via XSL} + \text{JSP}) \rightarrow \text{HTML}$$

- Sugestão de roteiro:
  - ⋆ Ver o exemplo do professor
  - ⋆ Aproveitar o método que retorna o documento DOM para as reservas a partir do BD (etapa BD $\rightarrow$ OO $\rightarrow$ XML as DOM)
  - ⋆ Criar uma página JSP que requisita ao método acima o documento DOM, aplica a folha de estilo e retorna o HTML.

- Alguns detalhes
  - ⋆ Colocar `mysql.jar` no `WEB-INF/lib`
  - ⋆ Colocar as classes do sistema no `WEB-INF/classes`
  - ⋆ Colocar as folhas de estilo no `WEb-INF/..`