Unidade 6: Java Server Pages

Prof. Daniel Caetano

Objetivo: Capacitar o aluno para produzir páginas usando a tecnologia JSP.

INTRODUÇÃO

Como foi observado na aula anterior, podemos dividir a tarefa de nossos Servlets entre aqueles que processam e aqueles que apresentam resultados, chamados respectivamente de Servlets de Processamento e Servlets de Apresentação.

Embora os Servlets desempenhem com ótimo resultado ambos os papéis, a construção de Servlets de Apresentação é um tanto desajeitada. Isso ocorre porque os Servlets de Apresentação têm a tarefa primordial de construir uma página web, em html, e a tecnologia Servlets em si não traz nenhuma facilidade para isso, tornando o processo tedioso e bastante propenso a erros.

Para mitigar este problema, a Sun Microsystems/Oracle criou a tecnologia **Java Server Pages** (ou, em bom português, Páginas em Servidor Java) - **JSP**, para os íntimos. Nesta aula veremos os fundamentos da tecnologia JSP e também transformaremos os servlets de aulas anteriores em JSPs.

1. O QUE É UM JSP?

Versão curta: se um servlet é um programa em Java com um pouco de script HTML...
... um JSP é um script HTML com um pouco de programa em Java.

Versão longa: JSP é uma forma alternativa de escrever um Servlet que facilita a vida do desenvolvedor quando a função principal do servlet é gerar uma página ou um formulário HTML. Uma forma alternativa de escrever um Servlet? Como assim?

Antes de mais nada, vamos ver o que é, qual a aparência, de uma página em JSP.

Uma página JSP tem a seguinte "cara":

index.jsp

```
<%--
    Document   : index
    Created on : 23/08/2011, 10:38:32
    Author     : djcaetano
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Teste</title>
    </head>
    <body>
        <h1>Olá mundo!</h1>
    </body>
</html>
```

Um desenvolvedor desavisado tomaria isso por um código HTML mas... observe que existem algumas tags estranhas, começando com **<%** e finalizando com **%>**. Essas tags servem para indicar um código JSP.

Apesar da aparência, entretanto, não se engane: estamos escrevendo um Servlet! Todo o código que funcionaria em um Servlet vai funcionar se for escrito dentro de <% ... %>!

Por outro lado, o servidor GlassFish ou TomCat **não** é capaz de executar um JSP; sempre que carregarmos um JSP pela primeira vez, o servidor irá transformar o JSP para um Servlet e irá, a partir de então, usar esse "Servlet gerado automaticamente" sempre que o JSP for solicitado.

2. FORMAS DE USAR CÓDIGO JAVA EM JSP

Existem várias formas de usar código Java em JSPs. Não faz parte do escopo do curso apresentar todas elas; contudo, iremos apresentar algumas mais importantes.

Basicamente existem formas para:

a) Inserir um código completo em Java, como feito no Servlet
b) Imprimir o valor de uma variável ou expressão
c) Chamar comandos específicos dos Servlets
d) Declarar variáveis e métodos
e) Diretivas

Cada uma destas formas será apresentada a seguir.

2.1. Inserindo Código Java no JSP (Scriplet Tag)

Primeiramente, vejamos como inserir código Java tradicional. Esse é o mais fácil: basta usar a tag <% ... %> para inserir código em qualquer lugar da página:

```
<%--
    Document   : index
    Created on : 04/03/2011, 10:13:15
    Author     : djcaetano
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Teste</title>
    </head>
    <body>
        <h1>Olá mundo!</h1>
        <%
            int a = 2;
            int b = 3;
            int c = a + b;
            out.println("<p>Resultado: " + c + "</p>");
        %>
    </body>
</html>
```

É importante ressaltar que vários **objetos** já estão pré-definidos nesse modo:

out	Objeto de impressão
request	Objeto de requisição
response	Objeto de resposta
session	Objeto de sessão (será visto posteriormente)

Existem outros dentre outros menos usados.

Qualquer código que quisermos executar em Java, basta colocar aí dentro. A única pegadinha é que, dentro de um scriplet, não podemos usar as palavrinhas "import" para indicar "onde estão as classes que estamos usando" e, então, temos que fazer isso toda vez que as utilizarmos. Por exemplo, para usar a classe **Date**, que está no pacote **java.util**, temos que escrever o seguinte no programa:

```
<%--
    Document   : index
    Created on : 04/03/2011, 10:13:15
    Author     : djcaetano
--%>
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Teste</title>
    </head>
    <body>
        <h1>Olá mundo!</h1>
        <%
            java.util.Date data = new java.util.Date();
            out.println("<p>Hoje é: " + data + "</p>");
        %>
    </body>
</html>
```

Mais adiante veremos uma forma mais simples de resolver isso.

2.2. Imprimindo o Valor de uma Variável ou Expressão (Expression Tag)

Em algumas situações, como foi visto no exemplo anterior, queremos apenas imprimir um valor ou resultado, tornando excessivo o uso de tanto código. Para evitar isso, foi criada uma *tag* específica para a impressão de resultados: **<%=** ... **%>**.

Esta *tag* funciona da seguinte forma:

```
<%--
    Document   : index
    Created on : 04/03/2011, 10:13:15
    Author     : djcaetano
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Teste</title>
    </head>
    <body>
        <h1>Olá mundo!</h1>
        <%= new java.util.Date() %>
    </body>
</html>
```

O **<%=** inicial implica que haverá apenas uma instrução e o resultado dela deverá ser impresso. Não é possível inserir múltiplas instruções separando-as por ponto-e-vírgula dentro dessa seção. Na verdade, **não finalize a instrução por ponto-e-vírgula!**

Se quisermos pegar um valor da requisição e imprimí-lo, podemos fazer como é indicado no código a seguir:

```
<%--
    Document   : index
    Created on : 04/03/2011, 10:13:15
    Author     : djcaetano
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Teste</title>
    </head>
    <body>
        <h1>Olá mundo!</h1>
        <%= request.getParameter("nome") %>
    </body>
</html>
```

E se precisarmos fazer algum cálculo complexo antes de imprimir os resultados?

Basta realizar os cálculos ANTES!

```
<%--
    Document   : index
    Created on : 04/03/2011, 10:13:15
    Author     : djcaetano
--%>

<%    int a = 10;
      int b = 30;
      int c = (a*b)+a;
      String peso = request.getParameter("peso");
      String texto = "Os resultados são: " + peso + " e " + c;
%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Teste</title>
    </head>
    <body>
        <h1>Olá mundo!</h1>
        <%= texto %>
    </body>
</html>
```

2.3. Executando Recursos de Servlets Direto do JSP (Action Tag)

Alguns recursos dos Servlets estão acessíveis diretamente a partir dos JSPs, com o uso de tags especiais iniciadas por **jsp:** .

Por exemplo, para realizar a transferência da requisição de um JSP para um Servlet (ou mesmo para outro JSP), pode-se usar o **forward** como visto na aula passada... ou sua forma mais simples do JSP! Como é isso?

FORA de uma seção <% ... %> indica-se o forward com:

<jsp:forward page="/NomeDoServlet" />

Exemplo:

```
<%--
    Document   : index
    Created on : 04/03/2011, 10:13:15
    Author     : djcaetano
--%>

<jsp:forward page="outro.jsp" />

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Teste</title>
    </head>
    <body>
        <h1>Olá mundo!</h1>
    </body>
</html>
```

NOTA: só funciona com páginas e servlets no próprio servidor!

Um recurso insteressante é incluir um JSP dentro de outro, o que pode ser feito com o comando **include**:

<jsp:include page="outro.jsp" flush="true" />

Caso seja necessário definir parâmetros na chamada, usa-se:

<jsp:include page="/Outro.jsp" flush="true"/>
        <jsp:param name="nomeDoParametro" value="valorDoParametro" />
</jsp:include>

Observe o exemplo a seguir

index.jsp

```
<%--
    Document   : index
    Created on : 04/03/2011, 10:13:15
    Author     : djcaetano
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Teste</title>
    </head>
    <body>
        <jsp:include page="conteudo.jsp" flush="true" />
    </body>
</html>
```

conteudo.jsp

```
<%--
    Document   : index
    Created on : 04/03/2011, 10:13:15
    Author     : djcaetano
--%>

    <h1>Olá Mundo!</h1>
```

2.4. Declarando Variveis e Métodos (Declaration Tag)

Se tudo que precisamos em um momento é declarar uma variável com um certo valor ou mesmo um método, isso pode ser feito com a tag de declaração: **<%!** ... **%>**

Observe o exemplo:

```
<%--
    Document   : index
    Created on : 04/03/2011, 10:13:15
    Author     : djcaetano
--%>

<%! private int contador = 1345; %>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Teste</title>
    </head>
    <body>
        <h1><%= contador %></h1>
   </body>
</html>
```

2.5. Diretivas (Directive Tag)

Tags do tipo **<%@ ... %>** servem para indicar informações adicionais ao Java, como definir o tipo de codificação a ser usada na geração do código HTML:

<%@page contentType="text/html" pageEncoding="UTF-8" %>

Um atributo importante para a diretiva @page é o **import**, que premite declarar bibliotecas Java adicionais.

<%@page import="java.util.*" %>

```
<%--
    Document   : index
    Created on : 04/03/2011, 10:13:15
    Author     : djcaetano
--%>


<%@page contentType="text/html" pageEncoding="UTF-8" import="java.util.*"%>
<!DOCTYPE html>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Teste</title>
    </head>
    <body>
        <h1>Olá mundo!</h1>
        <%
            Date data = new Date();
            out.println("<p>Hoje é: " + data + "</p>");
        %>
    </body>
</html>
```

Uma outra diretiva útil é o **include** (é, existem duas formas de fazer um include!):

<%@include file="pagina.jsp" %>

A diferença desta **@include** "diretiva" para a **jsp:include** "action" é que a **jsp:include** ocorre sempre que o usuário solicitar a página novamente; a **@include** ocorre apenas no momento em que o JSP é convertido para Servlet na primeira vez.

3. ALTERANDO SISTEMAS COM SERVLETS PARA USAR JSPs

Basicamente, o processo é esse:

PASSO 1: Criar um JSP que substitua o Servlet de apresentação
Exemplo: para o servlet **Calcula.java**, crie o **Calcula.jsp**

PASSO 2: Ir até o descritor dos servlets (**Páginas Web > WEB-INF > web.xml**) e vá até a aba **Servlets**

PASSO 3: Procure o nome de seu servlet (por exemplo **Calcula -> /Calcula**) e, desmarque "**Classe do servlet**", ativando a opção "**Arquivo JSP**". Ao lado do "Arquivo JSP", indique o nome do seu JSP. Por exemplo: **Calcula.jsp** .

4. BIBLIOGRAFIA

DEITEL, H.M; DEITEL, P.J. **Java: como programar** - Sexta edição. São Paulo: Pearson-Prentice Hall, 2005.

QIAN, K; ALLEN, R; GAN, M; BROWN, R. **Desenvolvimento Web Java.** Rio de Janeiro: LTC, 2010.