Modelagem de Hipermídia
Prof. Giangiacomo Ponzo Neto
Aula 4

Tags ou Actions

Outra possibilidade na programação JSP é usar algumas *tags* específicas, também chamadas *actions*. Elas não figuram entre <% e %>, mas apenas < e > tal como as *tags* HTML. As *tags* JSP podem ser predefinidas ou carregadas de uma biblioteca externa de tags. As predefinidas começam por `<jsp:` como em `<jsp:include`, que serve para incluir conteúdo de outras páginas, tal como vimos na diretiva "include". Veja o código abaixo e compare com o que vimos na diretiva "include":

```
<html>
<body>
  Olá!
  <jsp:include page="outra.html"/>
</body>
</html>
```

Trocando o "`jsp:include`" para "`jsp:forward`" faz com que se vá para a página apontada (não dá nem para ver o "Olá!"). A diferença entre o método "response.sendRedirect" e a *tag* "jsp:forward" é que esta última requisita a nova página quando ainda no servidor, através de uma chamada a um método interno, o que faz com que *requests* e dados de sessão da página continuem usáveis na página chamada. A nova página continuará a requisição da anterior, como se fosse (embora não seja) uma página única.

Além das *tags* JSP predefinidas, podem existir extensões em estruturas chamadas "tag libraries", vindo com os servidores *web* e variando de acordo com o servidor e a versão. Cada biblioteca dessas tem sua documentação própria que deve ser estudada junto às demais características do servidor *web* escolhido.

Mantendo o Estado das Coisas

Já sabemos que o HTML é *stateless*, ou seja: não sabe o estado atual da transação em curso, em que ponto do "sistema" o usuário está, o que ocorreu antes, etc. Isso porque cada página HTML é uma entidade isolada. Na verdade, se formos olhar apenas as páginas, não existe um "sistema", só um monte de páginas. Essa é uma diferença fundamental entre um sistema *desktop* em Delphi tradicional ou em Java GUI com Frames e um sistema *web*: a aplicação desktop tem variáveis globais e um escopo comum onde tudo acontece em uma única estrutura lógica, a aplicação *web* não tem isso por natureza. Portanto, um usuário de sistema *web*, será um usuário de páginas. Se você programou essas páginas, será muito útil guardar dados globais e um escopo único para cada usuário. Mas como fazer isso se só temos um "monte de páginas"? Existem várias formas. Vamos dar uma olhada nelas.

Passar parâmetros de página em página

Talvez a pior forma, porém independe de armazenamento na máquina cliente (que pode estar bloqueada para esse tipo de operação) ou no servidor (que está sujeito a limites de tempo). O problema é que exige que as páginas passem como parâmetro umas às outras todas as variáveis e objetos globais que possam ser necessários.

Vamos ver um exemplo prático: o login[1] em um sistema *web*:

```
<html>
<body>
<form action="Entrada.jsp" method="post">
  Usuário <input type="text" name="txtUsuario" size=15><br>
  Senha <input type="password" name="txtSenha" size=15><br>
  <input type=submit value="Entrar">
</form>
</body>
</html>
```

[1] Um ótimo exemplo, por sinal, pois é muito comum precisarmos saber quem é o usuário durante toda a sua permanência no site, não só na página de entrada. Só para constar: para esse fim específico, existe uma possibilidade mais "RAD", usando action="`j_security_check`", mas didaticamente é melhor seguir o caminho proposto, que é de uso muito mais geral.

Crie o arquivo no NetBeans (html) e salve como "login" (login.html, a extensão é automática). Repare que a *action* do *form* permite enviar, via *post*, os dados para uma página chamada "Entrada.jsp". O propósito deste programa é testar usuário e senha e dar entrada no sistema (não me preocupei com o *layout*, apenas com a funcionalidade). Crie este novo arquivo (Arquivo-Novo Arquivo-Web-JSP) chamando-o de Entrada.jsp. Veja sua listagem a seguir:

```
<%
   String usuario=request.getParameter("txtUsuario");
   String senha= request.getParameter("txtSenha");
%>
<html>
<body>
  <% if ( (usuario.equals("abc")) && (senha.equals("123")) ) {
       response.sendRedirect("Outra.jsp?usuario="+usuario);
     }
     else {
  %>
       Cai fora!
  <%
     }
  %>
</body>
</html>
```

Compile o arquivo (F9). Repare que criamos duas *strings*: usuario e senha. Os valores vieram dos parâmetros passados pela página inicial. Depois, vem a validação da entrada: se o usuário for "abc" e tiver digitado a senha "123", ele chama outra página (Outra.jsp) e passa, via método GET (falamos disso no período passado, lembra?), o conteúdo do usuário. Se for necessário (e normalmente será) chamar outras páginas e manter quem é o usuário, esta informação deverá ser "perpetuada", ou seja: toda nova chamada para outras páginas deverá ser terminada com "?usuario=<%=usuário%>".

Atenção: você pode ser tentado a tirar os "{ }" do "if .. else", não é? Não é uma boa idéia... Lembre-se de que o arquivo JSP é traduzido para puro Java na criação do Servlet e pode ser que, na tradução, algumas linhas sejam transformadas em duas, três ou mais. Nesse caso acima, até funcionaria, mas como convenção geral, em JSP, sempre delimite os comandos que tenham escopo (if, while, do, etc.) com "{ }", para evitar problemas.

Como fazer para a nova página (Outra.jsp) ler o parâmetro "usuário"? Da mesma forma que a primeira. Veja abaixo:

```
<% String usuario=request.getParameter("usuario"); %>
<html>
<body>
  Bem-vindo <%=usuario%>!
</body>
</html>
```

Compile (F9), volte ao login.html e execute-o. Atenção: use SHIFT F6 para executar o login.html, estando em sua janela; se usar só F6 vai tentar rodar o index.jsp que é o principal.

Nota: para passar algum parâmetro não entrado pelo usuário, mas gerado pelo próprio sistema, usa-se um tipo de "input" HTML dentro dos *forms* que não usamos em LP6 (porque não tinha nenhuma utilidade àquela altura): o *hidden*. Veja o exemplo abaixo de um trecho em HTML:

```
...
<form action="MaisOutraPagina.jsp" method="post">
  <input type="hidden" name="confirma" value="S">
  <input type="submit">
</form>
...
```

No exemplo acima, a informação (confirma=S) será enviada para a página definida no *action* do *form.* Se precisarmos mandar algum tipo de informação adicional, podemos usar esse recurso.