JSP com tags padrão

Helder da Rocha
www.argonavis.com.br

Objetivos

- *Utilizar os marcadores padrão do JSP com o objetivo principal de reduzir a quantidade de código Java nas páginas e promover a separação de responsabilidades*
  - *Marcadores para uso de JavaBeans (View Helper pattern): <jsp:useBean>, <jsp:getProperty>, <jsp:setProperty>*
  - *Marcadores para divisão de páginas em blocos menores que são compostos em tempo de execução ou de compilação (Composite View pattern): <%@include%> e <jsp:include>*
  - *Marcadores para redirecionamento de requisição para outras páginas ou servlets <jsp:forward>*
  - *Marcadores para geração de código HTML de suporte a applets: <jsp:plugin>*

Usando beans

- JavaBeans são objetos escritos de acordo com um determinado padrão que permite tratá-los como componentes de um framework
  - Ótimos para separar os detalhes de implementação de uma aplicação de seus "serviços"
  - Permitem encapsular dados recebidos de outras partes da aplicação e torná-los disponíveis para alteração e leitura através de uma interface uniforme.
- Podem ser usados com JSP para remover grande parte do código Java de uma página JSP
  - Maior facilidade de manutenção e depuração
  - Separação de responsabilidade e reuso de componentes

Como incluir um bean

- *Para que um bean possa ser usado por uma aplicação JSP, ele deve estar compilado e localizado dentro do CLASSPATH reconhecido pelo servidor*
  - *No subdiretório WEB-INF/classes do seu contexto*
- *Para incluir:*

```
<jsp:useBean id="nome_da_referência"
             class="pacote.NomeDaClasse"
             scope="page|session|request|application">
</jsp:useBean>
```

- *O atributo de escopo é opcional e indica o tempo de vida do Java Bean. Se omitido, será page, que o limita à página*
  - *Com escopo de request, o bean pode ser recuperado com outra instrução <jsp:useBean> que esteja em outra página que receber a mesma requisição (via dispatcher)*
  - *Com escopo de session, o bean é recuperável em páginas usadas pelo mesmo cliente, desde que <%@page> não tenha session=false*

Como incluir um bean

- *O nome do bean (atributo id) comporta-se como uma referência a um objeto Java*
- *Incluir o tag*

```
<jsp:useBean id="bean" class="bean.HelloBean"
             scope="request" />
```

é o mesmo que incluir na página

```
<% Object obj = request.getAttribute("bean");
   bean.HelloBean bean = null;
   if (obj == null) {
      bean = new bean.HelloBean();
      request.setAttribute("bean", bean);
   } else {
      bean = (bean.HelloBean) obj;
   } %>
```

- *O id pode ser usado em scriptlets para usar membros do bean*

```
<%  bean.setValor(12); %>
```

Propriedades

- *JavaBeans possuem propriedades que podem ser somente-leitura ou leitura-alteração.*
- *O nome da propriedade é sempre derivada do nome do método getXXX():*

```
public class Bean {
  private String mensagem;
  public void setTexto(String x) {
    mensagem = x;
  }
  public String getTexto() {
    return mensagem;
  }
}
```

- *O bean acima tem uma propriedade (RW) chamada* `texto`

Propriedades

- *Páginas JSP podem ler ou alterar propriedades de um bean usando os tags*

```
<jsp:setProperty name="bean" property="propriedade"
                 value="valor"/>
```

que equivale a `<% bean.setPropriedade(valor); %>` e

```
<jsp:getProperty name="bean" property="propriedade"/>
```

que equivale a `<%=bean.getPropriedade() %>`

- *Observe que o nome do bean é passado através do atributo name, que corresponde ao atributo id em* `<jsp:useBean>`
- *Valores são convertidos de e para String automaticamente*
- *Parâmetros HTTP com mesmo nome que as propriedades têm valores passados automaticamente com <jsp:setProperty>*
  - *Se não tiverem, pode-se usar atributo param de <jsp:setProperty>*
  - *<jsp:setProperty ... property="*"/> lê todos os parâmetros*

Inicialização de beans

- *A tag <jsp:useBean> simplesmente cria um bean chamando seu construtor. Para inicializá-lo, é preciso chamar seus métodos setXXX() ou usar <jsp:setProperty> após a definição*

```
<jsp:useBean id="bean" class="bean.HelloBean" />
<jsp:setProperty name="bean" property="prop" value="valor"/>
```

- *Se um bean já existe, porém, geralmente não se deseja inicializá-lo.*
- *Neste caso, a inicialização pode ser feita dentro do marcador <jsp:useBean> e o sistema só a executará se o bean for novo (se já existir, o código será ignorado)*

```
<jsp:useBean id="bean" class="bean.HelloBean" />
  <jsp:setProperty name="bean" property="prop" value="valor"/>
</jsp:useBean>
```

ou

```
<jsp:useBean id="bean" class="bean.HelloBean" />
  <% bean.setProp(valor); %>
</jsp:useBean>
```

Condicionais e iterações

- *Não é possível usar beans para remover de páginas Web o código Java de expressões condicionais e iterações como for do-while e while*
  - *Para isto, não há tags padrão. É preciso usar Taglibs*
- *Beans, porém, podem ser usados dentro de iterações e condicionais, e ter seus valores alterados a cada repetição ou condição*

```
<jsp:useBean id="mBean" class="MessageBean" scope="session" />
<% MessageBean[] messages = MessagesCollection.getAll();
   for (int i = messages.length -1; i >= 0; i--) {
      mBean = messages[i];
%>
   <tr><td><jsp:getProperty name="mBean" property="time"/></td>
   <td><%=mBean.getHost()%></td>
   <td><%=mBean.getMessage()%></td></tr>
<% } %>
```

Pode-se usar uma forma ou a outra

(MessageBean tem propriedades: time, host, message)

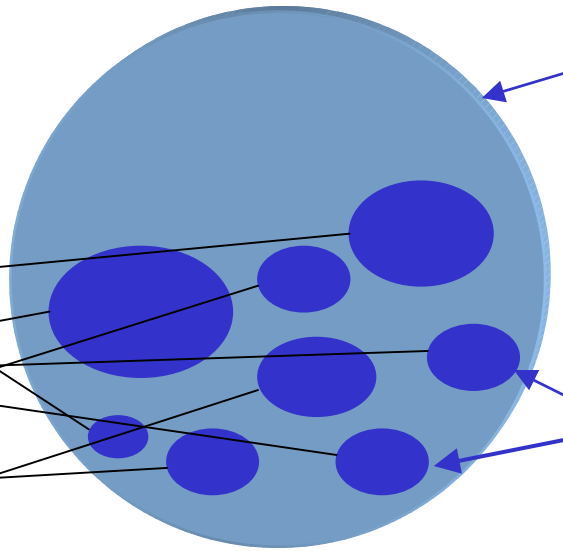
Matando beans

- *Beans são sempre gravados em algum objeto de escopo: page, request, session ou application*
  - *Persistem até que o escopo termine ou expirem devido a um timeout (no caso de sessões)*
- *Para se livrar de beans persistentes, use os métodos removeAttribute(), disponíveis para cada objeto de escopo:*

```
session.removeAttribute(bean);
application.removeAttribute(bean);
request.removeAttribute(bean);
```

Composite View

- *Páginas Web complexas (ex: portais) freqüentemente são divididas em partes independentes*
  - *Algumas partes são altamente dinâmicas, mudando frequentemente até na estrutura interna*
  - *Outras partes mudam apenas o conteúdo*
  - *Outras partes sequer mudam o conteúdo*
- *Gerar uma página dessas usando apenas um template é indesejável*

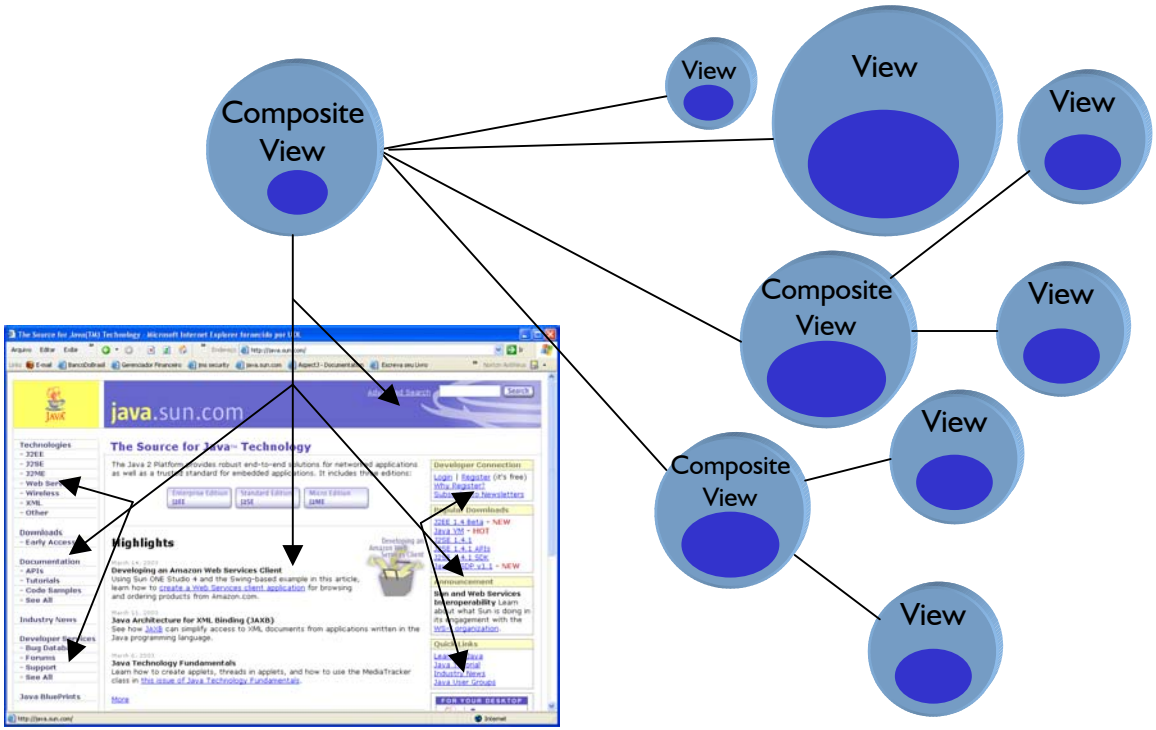


Template JSP

Código espalhado usado para gerar partes de cada seção da página

Composite View (2)

- *O padrão de projeto Composite View sugere que tais páginas sejam separadas em blocos menores, que possam ser alterados individualmente e compostos durante a publicação (deployment) ou exibição*



- *JSP oferece duas soluções para obter esse efeito*
  - *Usando inclusão estática (no momento da compilação do servlet)*
  - *Usando inclusão dinâmica (no momento da requisição)*

Inclusão estática

- *Mais eficiente: fragmentos são incluídos em único servlet*
- *Indicada quando estrutura não muda com freqüência (conteúdo pode mudar)*
  - *Menus, Logotipos e Avisos de copyright*
  - *Telas com miniformulários de busca*
- *Implementada com <%@ include file="fragmento" %>*

```
<!-- Menu superior -->
<table>
<tr><td><%@ include file="menu.jsp" %></td></tr>
</table>
<!-- Fim do menu superior -->
```

Fragmento menu.jsp

```
<a href="link1">Item 1</a></td>
<td><a href="link2">Item 2</a></td>
<a href="link3">Item 3</a>
```

Se tela incluida contiver novos fragmentos, eles serão processados recursivamente

Inclusão dinâmica

- *Mais lento: fragmentos não são incluídos no servlet mas carregados no momento da requisição*
- *Indicada para blocos cuja estrutura muda com freqüência*
  - *Bloco central ou notícias de um portal*
- *Implementada com <jsp:include page="fragmento"/>*
- *Pode-se passar parâmetros em tempo de execução usando <jsp:param> no seu interior*

```
<!-- Texto principal -->
<table>
<tr><td>
<jsp:include page="texto.jsp">
   <jsp:param name="data" value="<%=new Date() %>">
</jsp:include>
</td></tr> </table>
<!-- Fim do texto principal -->
```

Repasse de requisições

- *Uma requisição pode ser repassada de uma página JSP para outra página ou servlet usando RequestDispatcher*

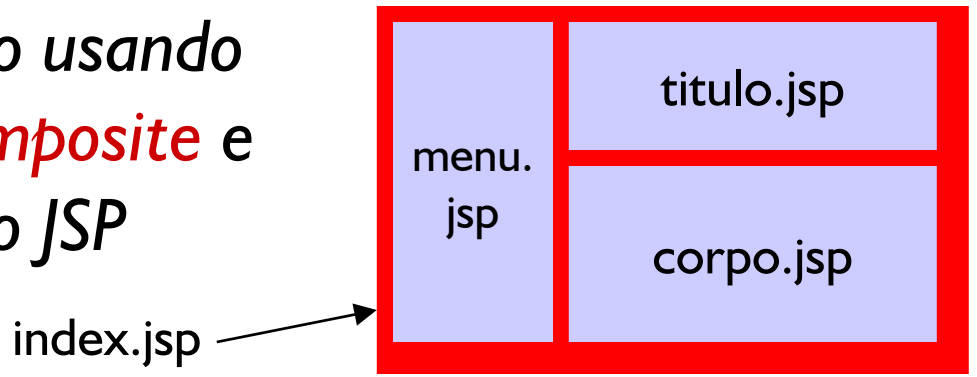
```
<% RequestDispatcher rd =
            request.getRequestDispatcher("url");
   rd.dispatch(request, response);
%>
```

- *O mesmo efeito é possível sem usar scriptlets com a ação padrão <jsp:forward>*
- *Assim como <jsp:include>, pode incluir parâmetros recuperáveis na página que receber a requisição usando request.getParameter() ou <jsp:getProperty> se houver bean*

```
<% if (nome != null) { %>
<jsp:forward page="segunda.jsp">
   <jsp:param name="nome" value="<%=nome %>">
</jsp:forward>
<% } %>
```

Exercícios

- *1. Use um JavaBean Mensagem, com propriedades email e mensagem para implementar o exercício 4 do capítulo anterior*
  - *Substitua todas as chamadas de new Mensagem() por <jsp:useBean> no escopo da sessão*
  - *Use <jsp:getProperty> para exibir os dados*
- *2. Altere gravarMensagens para que use <jsp:forward> para despachar a requisição para uma página erro.jsp, caso o usuário deixe os campos do formulário em branco, e para listarMensagens.jsp se tudo funcionar corretamente*
- *3. Monte a página ao lado usando os arquivos em cap10/composite e os marcadores de inclusão JSP*



helder@acm.org

argonavis.com.br