Aplicações de Linguagem de Programação Orientada a Objeto

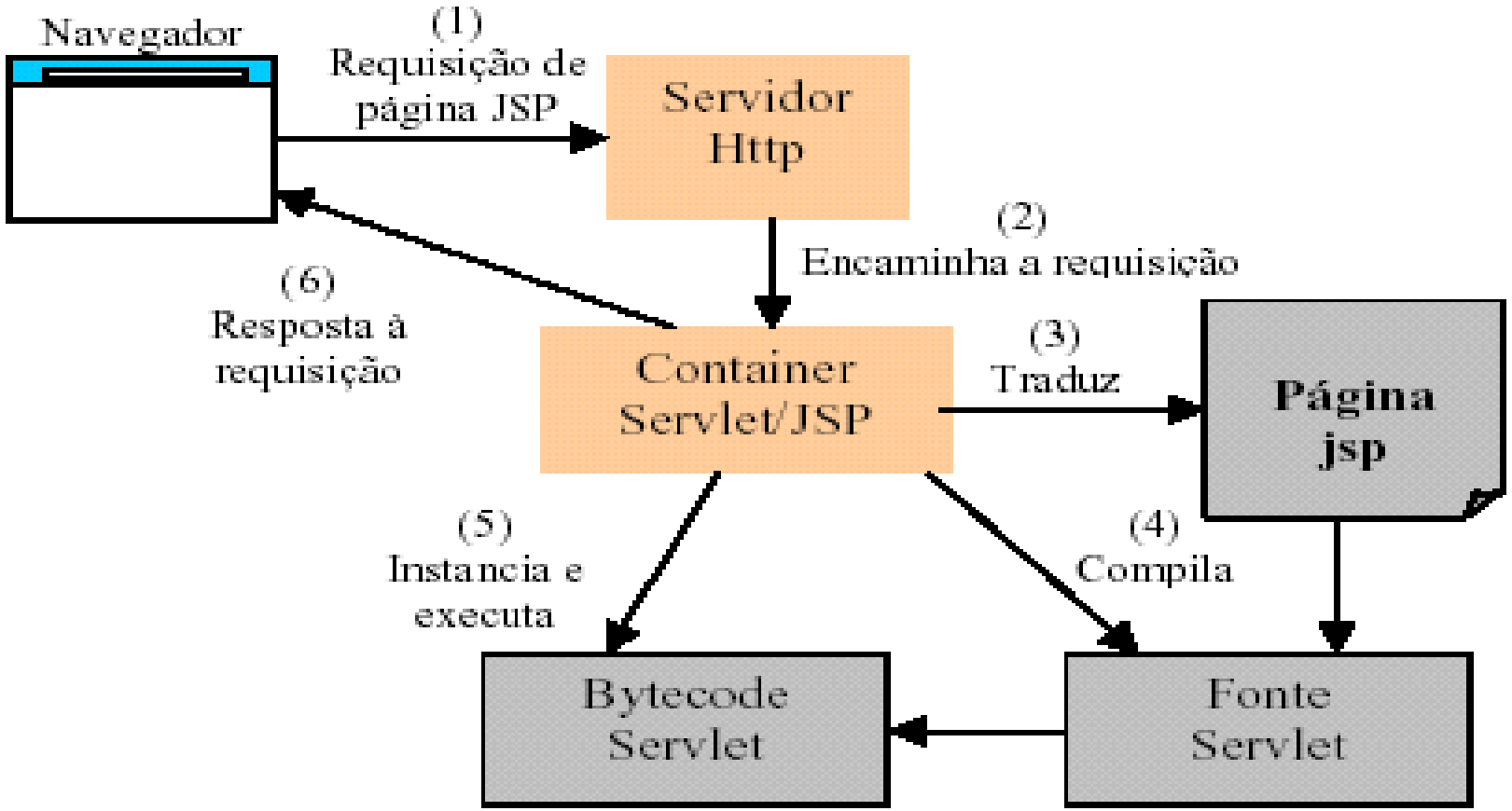
WEB
JSP

Professora Sheila Cáceres

JSP

- É uma tecnologia que permite incluir código Java dentro de páginas web.
- Uma página jsp é uma página HTML com trechos de programa Java embutidos e outras tags especiais.
- A página **JSP é automaticamente transformada em Servlet.**
- **Contenedor JSP:** Encarregado de substituir o código Java pelo resultado da sua execução na página web e enviá-la ao cliente.
- Assim, é possível desenhar páginas com partes fixas html e partes variáveis contendo código Java.

Introdução


Navegador
(1)
Requisição de
página JSP
Servidor
Http
(2)
Encaminha a requisição
(6)
Resposta à
requisição
Container
Servlet/JSP
(3)
Traduz
Página
jsp
(5)
Instancia e
executa
(4)
Compila
Bytecode
Servlet
Fonte
Servlet

Servlet x JSP

- Servlets
  - Forçam o programador a embutir código html dentro de código java
  - Não adequado quando a maior parte do que tem que ser gerado é texto ou código HTML estático (muitos println).
  - Portanto, Não permite independência entre o designer e o programador. O programador tem que ser bom web designer

```
Date hoje = new Date();
out.println("<body>");
out.println("<p>A data de hoje é "+hoje+".</p>");
out.println("<body>");
```

HojeServlet.java

- JSP
  - Solução aos problemas expostos acima: escrever um arquivo **template**:

```
<body>
<p>A data de hoje é <!--#data#-->.</p>
<body>
```

template.html

  - Bem mais simples de tratar.
  - Um nível maior de abstração pro Servlets (no final, tudo vira servlets)
  - O Web Designer pode trabalhar independente do Web Developer e vice-versa.

Exemplo

- JSP se baseia em templates para servlets. O mecanismo que a traduz é embutido no servidor.
- Em um servidor que suporta JSP, o processamento de JSP passa por uma camada adicional onde a página é transformada (compilada) em um servlet.
- Acesso via URL usa como localizador a própria página.
- A solução completa do slide anterior seria:

```
<html>
<body>
      A data de hoje é:
      <h1><%=new java.util.Date()%>  </h1>
</body>
</html>
```

Exemplos de JSP

- A forma mais simples de criar documentos JSP, é
  - 1. Mudar a extensão de um arquivo HTML para .jsp
  - 2. Colocar o documento em um servidor que suporte JSP
- Fazendo isto, a página será transformada em um servlet
  - A compilação é feita no primeiro acesso
  - Nos acessos subseqüentes, a requisição é redirecionada ao servlet que foi gerado a partir da página
- Transformado em um JSP, um arquivo HTML pode conter blocos de código (scriptlets): <% ... %> e expressões <%= ... %>

```
<p>Texto repetido:
<% for (int i = 0; i < 10; i++) { %>
  <p>Esta é a linha <%=i %>
<% }%>
```

Sintaxe dos elementos JSP

- Podem ser usados em documentos de texto (geralmente HTML ou XML)
- Todos são interpretados no servidor (jamais chegam ao browser)
  - **diretivas**: <%@ ... %>
  - **declarações**: <%! ... %>
  - **expressões**: <%= ... %>
  - **scriptlets**: <% ... %>
  - **comentários**: <%-- ... --%>
  - **ações**: <jsp:ação ... />
  - **custom tags**: <prefixo:elemento ... />

(a) Diretivas

- Contém informações necessárias ao processamento da classe do servlet que gera a página JSP
- Sintaxe :

  <%@ diretiva atrib1 atrib2 ... %>
- Principais diretivas:
  - page: atributos relacionados à página
  - include: inclui outros arquivos na página
  - taglib: declara biblioteca de custom tags usada no documento
- Exemplos

```
<%@ page import="java.net.*, java.io.*"
   session="false"
   errorPage="/erro.jsp" %>
<%@ include file="navbar.jsp" %>
```

(a) Diretiva page

- *Atributos de* **<%@page ... %>**
- Os **...** podem ser substituidos por um ou várias das características apresentadas a seguir:

info="Texto informativo" *default: nenhum*
language="java" *(default)*
contentType="text/html; charset=ISO-8859-1" *(default)*
extends="acme.FonteJsp" *default: nenhum*
import="java.io.*, java.net.*" *default: java.lang*
session="true" *(default)*
buffer="8kb" *(default)*
autoFlush="true" *(default)*
isThreadSafe="true" *(default)*
errorPage="/erros/404.jsp" *default: nenhum*
isErrorPage= "false" *(default)*

(b) Declarações

- Dão acesso ao corpo da classe do servlet (fora dos métodos). Permitem a declaração de variáveis e métodos em uma página.
- Úteis para declarar:
  - Variáveis e métodos de instância (pertencentes ao servlet)
  - variáveis e métodos estáticos (pertencentes à classe do servlet)
  - Classes internas (estáticas e de instância), blocos static, etc.
- Sintaxe

  <%! declaração %>
- Exemplos

```
<%! public final static String[] meses =
      {"jan", "fev", "mar", "abr", "mai", "jun"};
%>
<%! public static String getMes() {
       Calendar cal = new GregorianCalendar();
       return meses[cal.get(Calendar.MONTH)];
    }
%>
```

(c) Expressões

- Quando processadas, retornam um valor que é inserido na página no lugar da expressão
- Sintaxe:

  <%= expressão %>
- Equivale a out.print(expressão), portanto, não pode terminar em ponto-e-vírgula
  - Todos os valores resultantes das expressões são convertidos em String antes de serem redirecionados à saída padrão

(d) Scriptlets

- São blocos de código que são executados sempre que uma página JSP é processada.
- Correspondem a inserção de sequências de instruções no método _jspService() do servlet gerado.
- Sintaxe: <% instruções Java; %>

(e) Comentários

- Comentários HTML <!-- --> não servem para comentar JSP

  <!-- Texto ignorado pelo browser mas não pelo

  servidor. Tags são processados -->

- Comentários JSP: podem ser usados para comentar blocos JSP

  <%-- Texto, código Java, <HTML> ou tags <%JSP%> ignorados pelo servidor --%>

- Pode-se também usar comentários Java quando dentro de scriptlets, expressões ou declarações:

  <% código JSP ... /* texto ou comandos Java

  ignorados pelo servidor */ ... mais código %>

(f) Ações padronizadas

- Sintaxe:

```
<jsp:nome_ação atrib1 atrib2 ... >
   <jsp:param name="xxx" value="yyy"/>
   ...
</jsp:nome_ação>
```

- Permitem realizar operações (e meta-operações) externas ao servlet (tempo de execução)
  - Concatenação de várias páginas em uma única resposta
    <jsp:forward> e <jsp:include>
  - Inclusão de JavaBeans
    <jsp:useBean>, <jsp:setProperty> e <jsp:getProperty>
  - Geração de código HTML para Applets
    <jsp:plugin>

(f) ações (exemplos)

```
<%
if (Integer.parseInt(totalImg) > 0) {
%>
  <jsp:forward page="selecimg.jsp">
     <jsp:param name="totalImg"
                value="<%= totalImg %>"/>
     <jsp:param name="pagExibir" value="1"/>
  </jsp:forward>
<%
} else {
%>
  <p>Nenhuma imagem foi encontrada.
<%
}
%>
```

Objetos implícitos JSP

- São variáveis locais previamente inicializadas
- Disponíveis nos blocos <% ... %> (scriptlets) de qualquerpágina (exceto session e exception que dependem de @page para serem ativados/desativados)

- Objetos do servlet
  - page
  - config
- Entrada e saída
  - **request**
  - **response**
  - **out**
- Objetos contextuais
  - session
  - application
  - pageContext
- Controle de exceções
  - exception

- **Request**:Referência para os dados de entrada enviados na requisição do cliente (objeto do tipo HttpServletRequest)

```
<%   String nome = request.getParameter("nome");
 %> <p>Bom dia <%=nome %></p>
```

Response: Referência aos dados de saída enviados na resposta do servidor enviada ao cliente (objeto do tipo HttpServletResponse)

- Serve para definir o tipo dos dados retornados (default: text/thml), criar cookies, definir cabeçalhos, redirecionar, etc:
- Exemplo response.sendRedirect("pagina2.html");

Out: Representa o stream de saída da página (texto que compõe o HTML que chegará ao cliente).

- É instância da classe javax.servlet.jsp.JspWriter (equivalente a response.getWriter())
- Os trechos de código abaixo são equivalentes

```
<% for (int i = 0; i < 10; i++) {
out.print("<p> Linha " + i);
} %>

<% for (int i = 0; i < 10; i++) { %>
<p> Linha <%= i %>
<% } %>
```

Imports

- Funcionam como os imports de java

<%@page import = “<pacote>.<classe>" %>

Ex:

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

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

<%@page import = “usuario.Usuario" %>

Imports

- Funcionam como os imports de java

<%@page import = “<pacote>.<classe>" %>

Ex:

<%@page import = "fachada.Fachada" %>

<%@page import = “usuario.Usuario" %>

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

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

Include

- Juntando vários arquivos em um único arquivo...

Logar.jsp

```
<%@ include file=“header.jsp" %>
<%@ include file=“form_logar.jsp" %>
<%@ include file="footer.jsp" %>
```

Bibliografia

- Documentação Oficial: https://docs.oracle.com/javaee/5/tutorial/doc/bnagx.html
- Slides do Prof. Sílvio Bacalá Júnior, Faculdade de Computação, Universidade Federal de Uberlândia.