Java Web

AULA 07 – JSP

1

Objetivos e Conceitos

- Objetivos:
  - Apresentar JSP e a tradução para Servlet. Enumerar os objetos implícitos. Apresentar os blocos de código. Apresentar o ciclo de vida do JSP. Efetuar redirecionamentos em JSP.
- Conceitos:
  - JSP. Objetos implícitos. Scriptlets. Expressões. Diretivas. Ciclo de vida. Redirecionamentos.

2

Tópicos

- JSP
- Objetos Implícitos
- Blocos de Código
- Ciclo de Vida
- Redirecionamentos

3

JSP

4

JSP

Arquivo com formato HTML

Extensão **`.jsp`**

Contém código JAVA dentro das *tags* **`<% ... %>`**
- Chamado de *Scriptlet*

Do JSP é gerado uma SERVLET
- Contêiner converte automaticamente
- Ocorre na primeira invocação somente

O Servidor sempre executa a Servlet gerada

5

JSP Básico – Exemplo 1

```
<html>
   <head><title>Titulo</title>
   </head>
   <body>
      <b>Página HTML</b><br/>
      <h1><% out.println("Hello World!"); %></h1>
   </body>
</html>
```

Código Java
Scriptlet

6

JSP Básico – Exemplo 2.

```
<html>
   <head><title>Titulo</title>
   </head>
   <body>
      <b>Página HTML</b><br/>
      <h1><% out.println("Hello World!"); %></h1>
      <h2><%
         for (int i=0; i<10; i++) {
            out.println("Valor = " + i);
         }
      %></h2>
   </body>
</html>
```

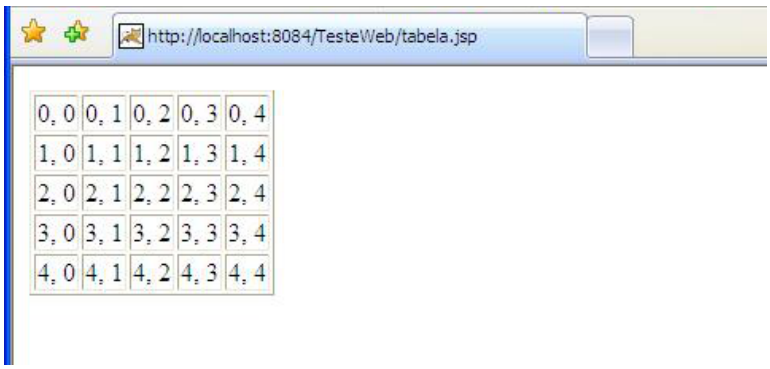

7

JSP: Misturando HTML e Java

```
<html><head><title>Tabelas</title></head>
<body>
   <table border="1">
      <% String str = "";
         for (int i=0; i<5; i++) {
            str += "<tr>";
            for (int j=0; j<5; j++) {
               str += "<td style=\"text-align:center\">";
               str += i + ", " + j;
               str += "</td>";
            }
            str += "</tr>";
         }
         out.println(str);  %>
      </table>
</body>
</html>
```


8

JSP: Misturando HTML e Java



9

JSP: Misturando HTML e Java.

```
<html><html><head><title>Tabelas</title></head>
<body>
   <table border="1">
   <% for (int i=0; i<5; i++) { %>
        <tr>
        <% for (int j=0; j<5; j++) { %>
               <td style="text-align:center">
              <%= i + ", " + j %>
              </td>
        <% } %>
        </tr>
    <% } %>
    </table>
</body>
</html>
```


10

JSP → Servlet

Servidor converte automaticamente o JSP para Servlet

Gera uma classe que herda de **HttpJspBase**

```
public final class portal_jsp
    extends org.apache.jasper.runtime.HttpJspBase
```

Todo seu código (HTML e Scriptlets) vão para dentro do método **_jspService()**, automaticamente

```
public void _jspService(HttpServletRequest request,
                        HttpServletResponse response)
    throws java.io.IOException, ServletException {
```


11

JSP → Servlet

Dentro do **_jspService()**

HTMLs viram

```
out.write("<h2>Oi mundo</h2>");
```

Scriptlets (códigos entre <% e %>) são copiados para dentro da Servlet

```
<%
    out.println("Oi mundo");
%>
```


12

JSP → Servlet

```
<html><head>
<title>Título</title>
</head><body>
      <%
         for (int i=0; i<10; i++) {
            out.println(i + " - ");
         }
      %>
</body></html>
```


13

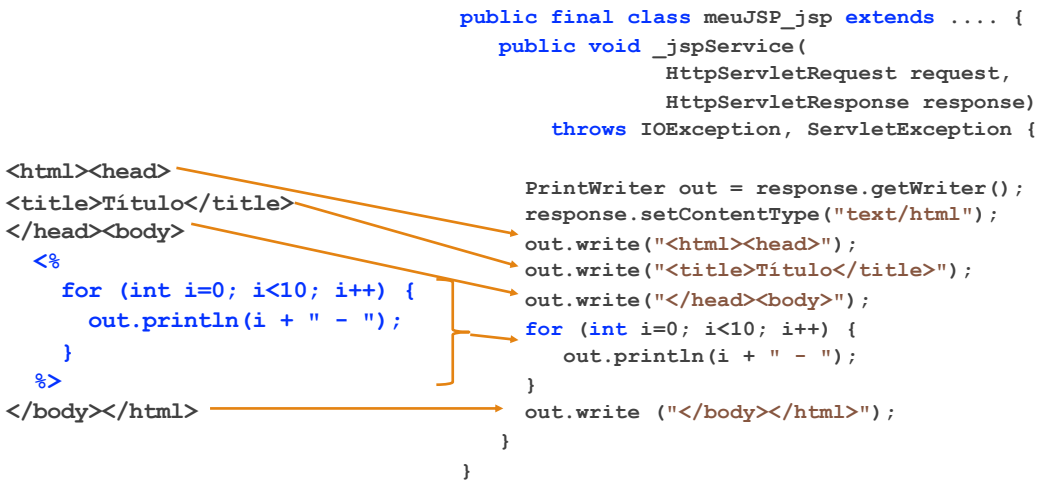
JSP → Servlet

```
public final class meuJSP_jsp
             extends org.apache.jasper.runtime.HttpJspBase
             implements org.apache.jasper.runtime.JspSourceDependent {

   public void _jspService(HttpServletRequest request,
                           HttpServletResponse response)
                     throws IOException, ServletException {
     PrintWriter out = response.getWriter();
     response.setContentType("text/html");
     out.write("<html><head>");
     out.write("<title>Título</title>");
     out.write("</head><body>");
     for (int i=0; i<10; i++) {
        out.println(i + " - ");
     }
     out.write ("</body></html>");
   }
}
```


14

JSP → Servlet.



15

Exercícios..

1. Crie e execute os JSPs vistos na subseção "Misturando HTML e Java"
2. Crie e execute o JSP visto na tradução de JSP para Servlet.
3. Executar os seguintes passos no Netbeans:
   a. Botão direito sobre o JSP
   b. Opção "View Servlet"
   c. Identificar a Servlet subjacente, para a qual o JSP é convertido
   d. Identificar nesta Servlet o código que gera a saída HTML e o seu código Java

16

Objetos Implícitos

17

Objetos Implícitos

Na tradução de JSP para Servlet o contêiner instancia vários objetos

Não precisam ser declarados na JSP de forma explícita

- Já estão disponíveis
- Por isso são implícitos

São declarados e instanciados na Servlet subjacente

18

Objetos Implícitos

request : representa os dados passados na requisição da página (**HttpServletRequest**)

```
<% String str = (String)request.getParameter("nome"); %>
```

response : representa os dados de saída para o cliente (**HttpServletResponse**)

```
<% response.setContentType("text/html;charset=UTF-8"); %>
```

out : representa a saída para o cliente (**javax.servlet.jsp.JspWriter**)

```
<% out.println("<h1>Oi mundo</h1>"); %>
```

session : representa uma área de dados para armazenar dados durante a aplicação (**HttpSession**)

```
<% session.setAttribute("nome", "Razer"); %>
```


19

Objetos Implícitos.

config : Objeto de configuração do Servlet/JSP (**javax.servlet.ServletConfig**)

application : Representa o contexto da aplicação, para acessar dados compartilhados entre todos os Servlets/JSP (**javax.servlet.ServletContext**)

page : Sinônimo para **this**. (**java.lang.Object**)

pageContext : Acesso a todos os escopos, atributos de página (como requisição e resposta), objetos implícitos, etc. (**javax.servlet.jsp.PageContext**)

exception : Representa uma exceção, para tratamento de erros, somente definido em páginas de erro (**java.lang.Throwable**)

20

Objetos Implícitos

```
public void _jspService(HttpServletRequest request,
                        HttpServletResponse response)
        throws java.io.IOException, ServletException {

    PageContext pageContext = null;
    HttpSession session = null;
    ServletContext application = null;
    ServletConfig config = null;
    JspWriter out = null;
    Object page = this;
    JspWriter _jspx_out = null;
    PageContext _jspx_page_context = null;
```


21

Objetos Implícitos.

```
try {
   response.setContentType("text/html");
   response.setHeader("X-Powered-By", "JSP/2.3");
   pageContext = _jspxFactory.getPageContext(this, request,
                    response, null, true, 8192, true);
   _jspx_page_context = pageContext;
   application = pageContext.getServletContext();
   config = pageContext.getServletConfig();
   session = pageContext.getSession();
   out = pageContext.getOut();
   _jspx_out = out;
```


22

Blocos de Código JSP

23

Blocos de Código JSP.

Scriptlets

```
<% ... %>
```

Comentários

```
<%-- ... --%>
```

Declarações

```
<%! ... %>
```

Expressões

```
<%= ... %>
```

Diretivas

```
<%@ ... %>
```


24

Scriptlets.

```
<html>
   <head><title>Titulo</title>
   </head>
   <body>
      <b>Página HTML</b><br/>
      <h1><% out.println("Hello World!"); %></h1>
      <h2><%
         for (int i=0; i<10; i++) {
            out.println("Valor = " + i);
         }
      %></h2>
   </body>
</html>
```


25

Comentários.

```
<html>
   <head><title>Titulo</title>
   </head>
   <body>
      <b>Página HTML</b><br/>
      <h1><%-- out.println("Hello World!"); --%></h1>
      <h2><%--
         for (int i=0; i<10; i++) {
            out.println("Valor = " + i);
         }
      --%></h2>
   </body>
</html>
```


26

Declarações

Geram **Atributos** e **Métodos** na Servlet para a qual o JSP é gerado

Estão entre **<%!** e **%>**

Código adicionado após a definição da Servlet, independente de onde está no JSP

Pode-se adicionar vários destes trechos e em qualquer parte do JSP

Atributos

```
<%! public String frase = "Jakarta EE Rulez!!!"; %>
```

Métodos

```
<%!
    private String fazerSaudacao() {
        return "Olá queriduxos!!!";
    }
%>
```


27

Exemplo

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Teste</h1>
        <%!
            private String obterSaudacao() {
                return "Olá queriduxos!!!";
            }
        %>
        <h2>    <%  out.println(obterSaudacao()); %>   </h2>
        <h2>    <%  out.println(frase); %>   </h2>
    </body>
        <%!
            public String frase = "Jakarta EE Rulez!!!";
        %>
</html>
```


28

Exemplo

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Teste</h1>
        <%!
            private String obterSaudacao() {
                return "Olá queriduxos!!!";
            }
        %>
        <h2>    <%  out.println(obterSaudacao()); %>    </h2>
        <h2>    <%  out.println(frase); %>    </h2>
    </body>
        <%!
            public String frase = "Jakarta EE Rulez!!!";
        %>
</html>
```

Uso do atributo e do método.

29

Exemplo

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class teste_jsp
    extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent,
                org.apache.jasper.runtime.JspSourceImports {

            private String obterSaudacao() {
                return "Olá queriduxos!!!";
            }
            public String frase = "Jakarta EE Rulez!!!";

...
```


30

Exemplo.

```
out.write("<!DOCTYPE html>\n");
out.write("<html>\n");
out.write("    <head>\n");
out.write("        <meta http-equiv=\"Content-Type\"
content=\"text/html; charset=UTF-8\">\n");
out.write("        <title>JSP Page</title>\n");
out.write("    </head>\n");
out.write("    <body>\n");
out.write("        <h1>Teste</h1>\n");
out.write("        <h2>    ");
out.println(obterSaudacao());
out.write("   </h2>\n");
out.write("        <h2>    ");
out.println(frase);
out.write("   </h2>\n");
out.write("    </body>\n");
out.write("</html>\n");
```


31

Exercícios.

1. Executar o JSP anterior, contendo o atributo **`frase`** e o método **`obterSaudacao()`**
2. Executar os seguintes passos no Netbeans:
   a. Botão direito sobre o JSP
   b. Opção "View Servlet"
   c. Identificar a Servlet subjacente, para a qual o JSP é convertido
   d. Identificar nesta Servlet: A declaração do atributos, a declaração do método, o uso destes dois elementos
3. Dado o seguinte trecho de código em JSP:

```
<html><body>
<%! int x = 42; %>
<% int x = 22; %>
<% out.println(x); %>
</body></html>
```

O que será apresentado e por quê?

32

Expressões

Usadas para facilitar a escrita de expressões que devem ser mostradas

Deve ser colocada entre **<%=** e **%>**

Não recebe ponto e vírgula

Vira argumento de um **out.print()**

```
<%= 10 %>
<%= "Oi Mundo!!" %>
<%= new java.util.Date() %>
```


33

Expressões

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
     <title>JSP Page</title>
  </head>
  <body>
     <h1><%= new java.util.Date() %></h1>
  </body>
</html>
```


34

Expressões.

```
...
out.write("<!DOCTYPE html>\n");
out.write("<html>\n");
out.write("    <head>\n");
out.write("        <meta http-equiv=\"Content-Type\"
content=\"text/html; charset=UTF-8\">\n");
out.write("        <title>JSP Page</title>\n");
out.write("    </head>\n");
out.write("    <body>\n");
out.write("        <h1>");
out.print( new java.util.Date() );
out.write("</h1>\n");
out.write("    </body>\n");
out.write("</html>\n");
...
```


35

Exercícios.

1. Executar o JSP anterior
2. Executar os seguintes passos no Netbeans:
   a. Botão direito sobre o JSP
   b. Opção "View Servlet"
   c. Identificar a expressão na Servlet subjacente

36

Diretivas

Usadas para dar instruções ao contêiner na hora de traduzir a página para Servlet

Deve ser colocada entre **<%@** e **%>**

Tem-se somente 3 diretivas (com atributos):

- **page**
- **include**
- **taglib**

Exemplo de uso no topo da JSP

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


37

Exemplo.

```
<%@ page import="java.util.*" %>
<html>
<head><title>Teste</title></head>
<body>
   <b>Página HTML</b><br/>
   <%
          ArrayList<Integer> arr = new ArrayList<>();
          arr.add(10);
          arr.add(20);
          arr.add(30);
          for (Integer i: arr) {
             out.println("<h2>" + i + "</h2>");
          }
   %>
</body>
</html>
```


38

Diretiva: **page**

A diretiva **page** possui vários atributos diferentes

- **info**: Seta a string retornada por **getServletInfo()**

```
<%@ page info="Minha JSP Linda" %>
```

- **language**: Seta a linguagem usada o JSP (default **"java"**)

```
<%@ page language="java" %>
```

- **contentType**: Seta a resposta gerada pelo JSP (default **"text/html"**)

```
<%@ page contentType="application/pdf" %>
```


39

Diretiva: **page**

- **contentType**: Seta a resposta gerada pelo JSP (default **"text/html"**)

```
<%@ page contentType="text/html; charset=UTF-8" %>
```

- **pageEncoding**: Seta a codificação do arquivos JSP

```
<%@ page pageEncoding="UTF-8" %>
```


40

Diretiva: **page**

- **buffer**: Seta o buffer do JSP para entregar a saída ao cliente (objeto *response*). Recebe **"none"** ou **"<valor>kb"**. Default **"8kb"**.

```
<%@ page buffer="none" %>
<%@ page buffer="8kb" %>
```

- **autoFlush**: Quando *bufferizado*, indica se deve ser enviado ao cliente automaticamente ou uma exceção de *buffer overflow* deve ser gerada. Default **"true"**

```
<%@ page autoFlush="true" %>
```

- Em geral os programadores adicionam **buffer** e **autoFlush** juntos:

```
<%@ page buffer="8kb" autoFlush="true" %>
```


41

Diretiva: **page**

- **isThreadSafe**: Por default as JSPs são *thread-safe*. Se marcar como **"false"**, o contêiner garante que num determinado momento somente uma *thread* executa o JSP. Default **"true"**.

```
<%@ page isThredSafe="false" %>
```

- **session**: Se a JSP usa sessões automaticamente. Se marcado como **"false"**, então não se pode usar o objeto implícito **session**, nem **<jsp:useBean ... scope="session" />**. Default **"true"**

```
<%@ page session="true" %>
```

- **extends**: Faz a Servlet para a qual a JSP é gerada estender da classe apontada. Esta classe deve seguir vários requisitos

```
<%@ page extends="fully.qualified.name.MyClass" %>
```


42

Diretiva: **page**

- **import**: Uma lista de classes que serão importadas e estarão disponíveis no JSP. Pode-se separar várias classes por vírgula ou usar vários **page import** na página.

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

Pode ter vários *imports* na mesma declaração, separados por vírgula

```
<%@ page import="java.util.List,java.util.Date" %>
```

Ou pode ter vários *imports* em várias declarações

```
<%@ page import="java.util.List" %>
<%@ page import="java.util.Date" %>
```


43

Diretiva: **page .**

- **errorPage**: Indica para qual página o fluxo será redirecionado, caso uma exceção ocorra. Se começar com "/" é relativa à raiz de contexto. Se não, é relativa à pasta do JSP atual.

```
<%@ page errorPage="/erro.jsp" %>
```

- **isErrorPage**: Indica se a página onde é declarada é uma página de erro, isto é, uma página que pode ser setada em **page errorPage** de outras JSPs. Default **"false"**.

```
<%@ page isErrorPage="true" %>
```

Em sendo uma página de erro, tem acesso ao objeto implícito **exception**:

```
<%=exception.message %>
```


44

Diretiva: **include**

Possui somente um atributo: **file**

- Inclusão de arquivos na hora da tradução para *Servlet*
- Os dois arquivos são combinados para virar um só
- Pode incluir arquivos HTML, JSP, TXT, etc
- O arquivo é colocado no lugar da diretiva

Pode ser colocado em qualquer local no JSP

Cuidado ao usar tags **<html>, </html>, <body>, </body>** em arquivos incluídos, para não conflitar com as do JSP

45

Exemplo

Arquivo **cabecalho.jsp**

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html><head><title>Minha JSP</title></head>
<body>
<h2>Meu cabeçalho</h2>
<hr/>
```

Arquivo **rodape.jsp**

```
<hr/>
Rodapé legal
</body></html>
```


46

Exemplo.

Arquivo **portal.jsp**

```
<%@ include file="cabecalho.jsp" %>

<p>Portal mais lindo do mundo</p>

<%@ include file="rodape.jsp" %>
```


47

Diretiva: **taglib.**

Será usado quando introduzirmos JSTL/EL

Adiciona uma biblioteca de *tags* e o prefixo para acesso a elas

Dois atributos:

- **uri**: define qual biblioteca de tags está sendo usada
- **prefix**: define qual o prefixo a ser utilizado

Pode-se ter várias bibliotecas adicionadas, mas o prefixo deve ser único

Deve-se declarar a biblioteca antes de começar a usar suas *tags*

Exemplo:

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core"
          prefix="c" %>

<c:out value="${param.nome}" />
```


48

Exercícios.

1. Executar o JSP com exemplo de uso da diretiva de importação
2. Executar o JSP com exemplo de uso da diretiva de inclusão (cabeçalho, rodapé e portal)

49

Ciclo de Vida

50

Ciclo de Vida

Sabe-se que uma JSP é transformada para Servlet

O ciclo de vida de uma JSP, portanto, é muito parecido com a da Servlet

Exceto que:

- Na primeira invocação, o contêiner transforma a JSP em Servlet
- Após isso, o ciclo de vida é igual à Servlet:
  - Contêiner carrega a Servlet
  - Contêiner instancia a Servlet
  - Contêiner inicializa a Servlet, invoca o método **`init()`**, que invoca o **`jspInit()`**
  - Contêiner cria uma *thread* para cada requisição
  - Contêiner invoca o método **`service()`**, que invoca o método **`_jspService()`**
- Ao ser destruído
  - Contêiner invoca o método **`destroy()`**, que invoca o método **`jspDestroy()`**

51

Ciclo de Vida

Pode-se sobrepor os métodos **`jspInit()`** e **`jspDestroy()`**

Usa-se declarações **`<%! ... %>`**

Dentro do **`jspInit()`**/**`jspDestroy()`** consegue-se obter os objetos **`ServletConfig`** e **`ServletContext`** da seguinte forma:

```
ServletConfig config = getServletConfig();

ServletContext context = getServletContext();
```


52

Ciclo de Vida

Exemplo de **jspInit()**

```
<%!
  public void jspInit() {
     ServletConfig config = getServletConfig();
     ServletContext context = getServletContext();
     context.setAttribute("email", "razer@ufpr.br");
  }
%>
```


53

Ciclo de Vida.

Exemplo de **jspDestroy()**

```
<%!
  public void jspDestroy() {
    ServletContext context = getServletContext();
    context.log("JSP teste.jsp finalizado.");
  }
%>
```


54

Exercícios..

1. Crie uma JSP e declare os métodos **`jspInit()`** e **`jspDestroy()`** como abaixo:

```
<%!
  public void jspInit() {
     System.out.println("### Inicializou a JSP.");
  }
  public void jspDestroy() {
     System.out.println("### Vai destruir a JSP.");
  }
%>
```

a. Execute a JSP e encontre no log do Glassfish a mensagem que deve ser apresentada o **`jspInit()`**
b. Efetue a *undeploy* da aplicação e encontre no log do Glassfish a mensagem que deve ser apresentada no **`jspDestroy()`**

55

Redirecionamentos

56

Redirecionamentos: **forward**

Uso em JSP através de *tags*

Redireciona, via ***forward***, para o recurso indicado

```
<jsp:forward page="<pagina>" />
```

Onde

- **page** : Página HTML, JSP ou Servlet

Esta *tag* é convertida para usar o objeto **PageContext**:

```
_jspx_page_context.forward("<pagina>");
return;
```

E o método **forward()** e **pageContext** é uma invocação de **forward()** de **RequestDispatcher**

57

Redirecionamentos: **forward**

Pode-se passar parâmetros para o redirecionamento

Use-se a *tag* **<jsp:param />**

```
<jsp:forward page="<pagina>" >
   <jsp:param name="<parametro1>" value="<valor1>" />
   <jsp:param name="<parametro2>" value="<valor2>" />
   <jsp:param name="<parametro3>" value="<valor3>" />
</jsp:forward>
```

Onde

- **name** : Nome do parâmetro passado (na URL do redirecionamento)
- **value** : Valor do parâmetro passado (na URL do redirecionamento)

Os parâmetros são obtidos via **request.getParameter()**

58

Redirecionamentos: **forward.**

Cuidado!!!

Por default, a página é *bufferizada*

Se for usada a diretiva:

```
<%@ page buffer="none" %>
```

Indicará que não está usando *buffer* de saída

Portanto:

- Ao usar o **jsp:forward**
- Se já foi enviado algum dado para a saída
- Será levantada uma exceção: **IllegalStateException**

59

Redirecionamentos: **include**

Uso em JSP através de tags

Inclui a saída do recurso indicado

```
<jsp:include page="<pagina>" />
```

Onde

- **page** : Página HTML, JSP ou Servlet

Esta *tag* é convertida para usar o objeto **PageContext**:

```
_jspx_page_context.include("<pagina>");
```

E o método **include()** e **pageContext** é uma invocação de **include()** de **RequestDispatcher**

60

Redirecionamentos: **include**.

Pode-se passar parâmetros para o redirecionamento

Use-se a *tag* **<jsp:param />**

```
<jsp:include page="<pagina>" >
   <jsp:param name="<parametro1>" value="<valor1>" />
   <jsp:param name="<parametro2>" value="<valor2>" />
   <jsp:param name="<parametro3>" value="<valor3>" />
</jsp:include>
```

Onde

- **name** : Nome do parâmetro passado (na URL do redirecionamento)
- **value** : Valor do parâmetro passado (na URL do redirecionamento)

Os parâmetros são obtidos via **request.getParameter()**

61

Include-File x Jsp-Include.

Diretiva: **include file**

- Inclui o arquivo no local
- Inclusão estática
- Ocorre antes da execução do JSP (*merge* de arquivos)

```
<%@include file="teste.jsp" %>
```

Ação JSP: **include**

- Requisita o arquivo e coloca a saída no local
- Inclusão dinâmica
- Ocorre na hora da execução do JSP (**include()**)

```
<jsp:include page="teste.jsp" %>
```


62

Exercícios.

1. Crie a página index.html e 2 páginas JSP
   a. **`index.html`**: contém um formulário com nome e idade. Submete para **`portal.jsp`**
   b. **`portal.jsp`**: se a idade recebida for maior ou igual a 18, apresenta o conteúdo da página. Caso contrário, redireciona para **`erro.jsp`**, passando como parâmetro no **`<jsp:forward>`** a idade recebida
   c. **`erro.jsp`**: apresenta uma mensagem com a idade passada indicando que é inválida

DICA: é possível fazer códigos em JSP misturando *tags* e *scriplets*:

```
<% if (algumTeste) { %>
        <jsp:forward .... >
<% } %>
```


63