



Curso de **Java8** para **Web**

Professor
Antonio Benedito Coimbra Sampaio Jr

abc  | Treinamentos

www.abctreinamentos.com.br

Quarta Disciplina

JEE – Java Servlets e JSP

- **UNIDADE 1:** Introdução à Internet, WEB e HTML
- **UNIDADE 2:** Java Servlets
- **UNIDADE 3: JSP**
- **UNIDADE 4:** Padrão de Projeto MVC (Integração Servlet e JSP)

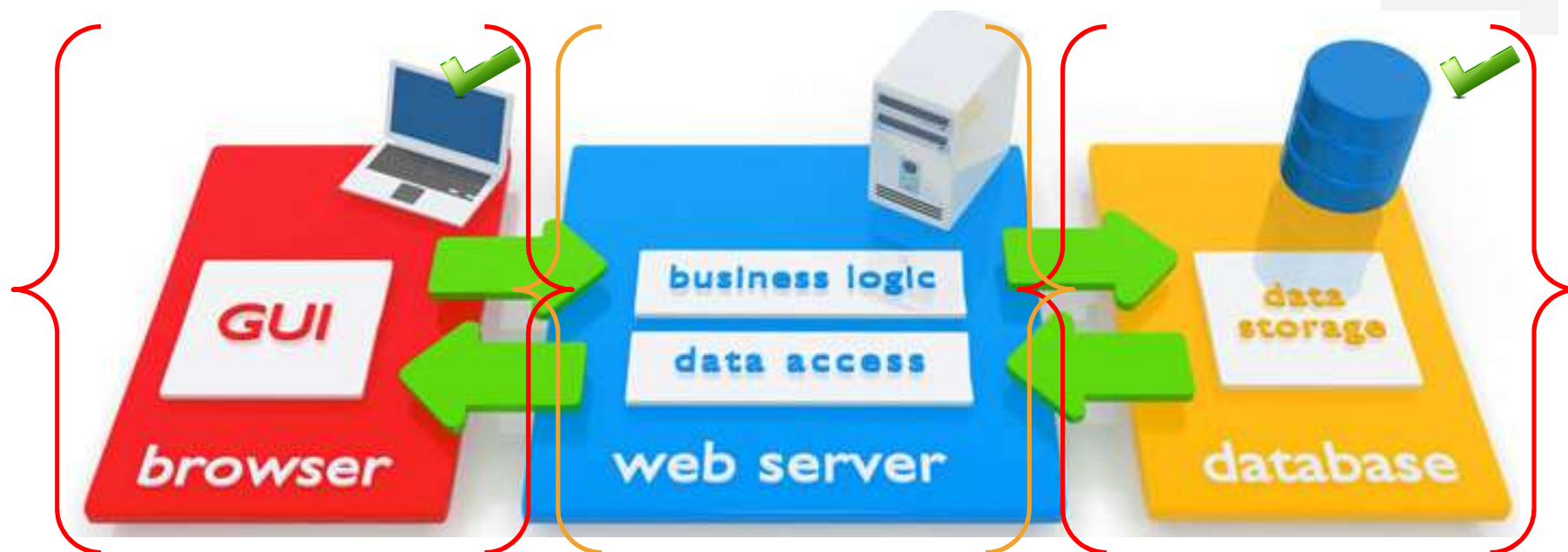
UNIDADE 3

JAVA SERVER PAGES (JSP)

Introdução ao JSP

Modelo de Aplicação JEE

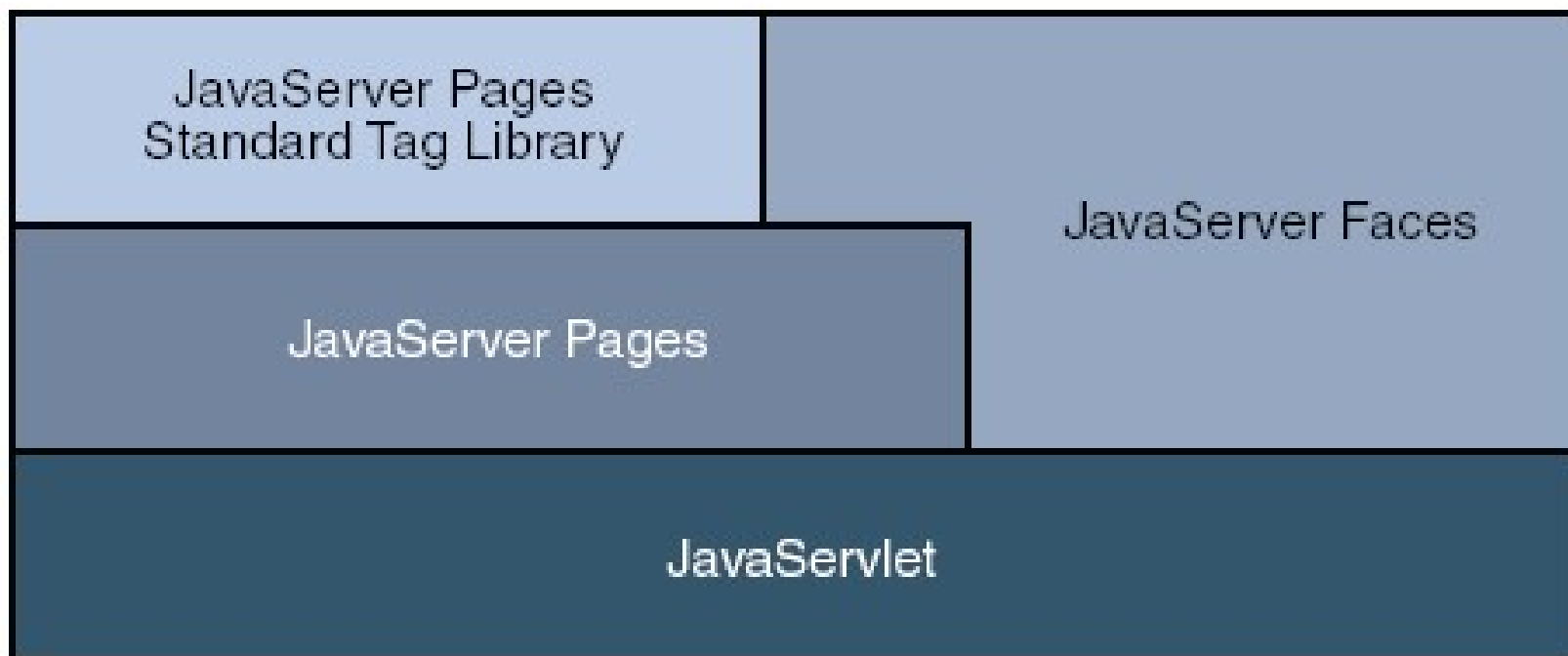
- A plataforma JEE utiliza um modelo de aplicação distribuída multicamada.



- A lógica da aplicação é dividida em componentes de acordo com a sua função.
- Os vários componentes que constituem uma aplicação JEE são instalados em diferentes equipamentos.

COMPONENTES WEB JEE

- A Plataforma JEE define quatro tecnologias básicas para a construção de Aplicações WEB:
 - **Java Servlets 3.1** ✓
 - **JavaServer Pages 2.2**
 - **JavaServer Faces 2.2**
 - **JavaServer Pages Standard Tag Library 1.2.1**

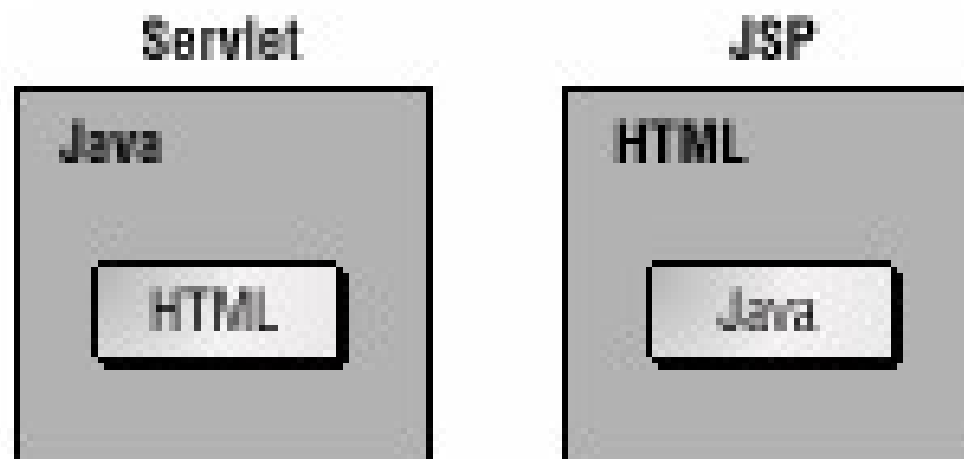


INTRODUÇÃO AO JSP

- O JSP é a tecnologia utilizada para a criação de componentes WEB que oferece mais recursos que a tecnologia Java Servlet, como **EL** (*Expression Language*) e **JSTL** (*Java Standard Tag Library*).

PRINCIPAIS CARACTERÍSTICAS DO JSP

- **(1) MENOS CÓDIGO**
 - No Servlet você cria uma classe Java com códigos HTML embutidos. No JSP é diferente! Você cria uma página HTML com código Java embutido.



INTRODUÇÃO AO JSP

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/OlaMundo")
public class OlaMundo extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<HTML>");
        out.println("<BODY>");
        out.println("Alo Mundo!");
        out.println("</BODY>");
        out.println("</HTML>");
    }
}
```

```
<HTML>
  <BODY>
    Alo Mundo
  </BODY>
</HTML>
```

AloMundo.jsp

INTRODUÇÃO AO JSP

PRINCIPAIS CARACTERÍSTICAS DO JSP

- **(2) FACILIDADE DE USO**
 - É mais fácil codificar com JSP do que com Servlets.
- **(3) FAZ USO DE TODAS AS APIS JAVA**
 - Os JavaServer Pages são construídos em cima da API Java Servlets, e assim como Servlets, também têm acesso a todas as poderosas APIs JEE, tais como JDBC, JNDI, EJB, JAXP, etc.
- **(4) É UTILIZADO EM COMBINAÇÃO COM SERVLETS**
 - As páginas JSP devem ser usadas em combinação com servlets para lidar com a lógica de negócios, fazendo uso do padrão de projeto MVC.

INTRODUÇÃO AO JSP

- Uma página JSP é um documento que contém dois tipos de texto:
 - 1) dados estáticos, que podem ser expressos em qualquer formato baseado em texto (como HTML, SVG, WML e XML);
 - 2) elementos JSP que constroem conteúdo dinâmico.



```
<%  
    Date data = new Date();  
    System.out.println("Data atual="+data);  
%>
```

data.jsp

- A extensão de arquivo para uma página JSP é **.jsp**.

API JSP

- A API JSP é composta por dois pacotes: **javax.servlet.jsp** e **javax.servlet.jsp.tagext**.
- O pacote **javax.servlet.jsp** define um conjunto de classes e interfaces utilizados na comunicação entre os JSPs e o seu ambiente de execução (web container).
 - Exemplo das Interfaces: JspPage e HttpJspPage.
 - Exemplo das Classes: JspWriter, PageContext, JspFactory, JspEngineInfo, JspException e JspError.
- O pacote **javax.servlet.jsp.tagext** define um conjunto de classes e interfaces utilizados na tecnologia JSTL.

CICLO DE VIDA JSP

- As quatro fases do ciclo de vida JSP são muito semelhantes ao ciclo de vida dos Servlets e são as seguintes:
 - **(1) Compilação**
 - **(2) Inicialização**
 - **(3) Execução**
 - **(4) Destruição**

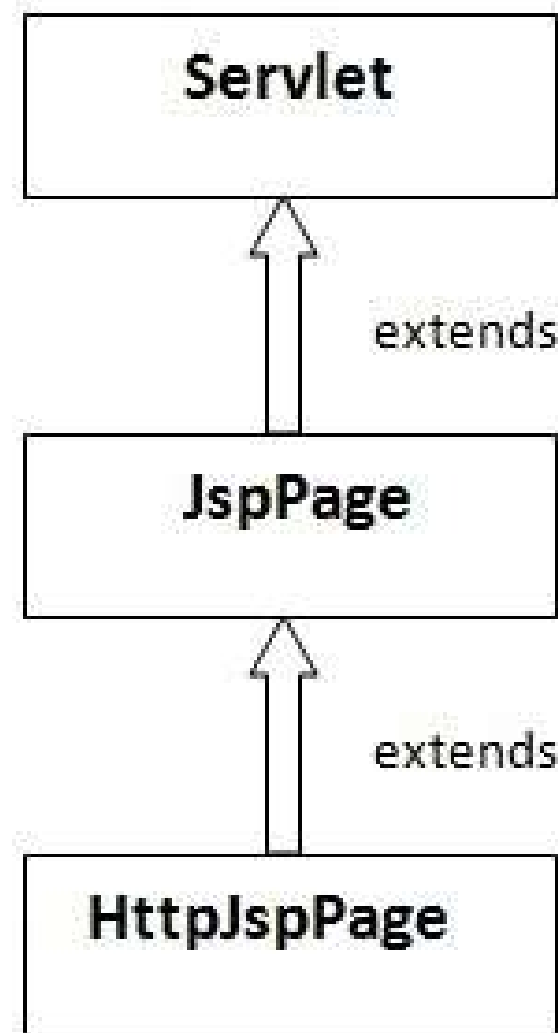
(1) COMPILAÇÃO

- Quando um navegador solicita um documento JSP, o motor JSP primeiro verifica se ele precisa compilar a página. Se a página não for compilada, o motor JSP compila a página.
- O processo de compilação envolve três etapas: a) Analisar o JSP; b) Transformar o JSP em um Servlet; e c) Compilar o Servlet.

CICLO DE VIDA JSP

(1) COMPILAÇÃO

- Todo servlet gerado a partir de um JSP deverá implementar a interface JspPage.
- Esta interface define os métodos:
 - **public void jsplnit()**
 - **public void jspDestroy()**
 - **public void _jspService()**



CICLO DE VIDA JSP

(2) INICIALIZAÇÃO

- Quando um contêiner carrega um JSP, primeiro ele invoca o método **jspInit()**.

```
public void jspInit(){  
    // Initialization code...  
}
```

- A inicialização é executada apenas uma vez e é utilizada para inicializar conexões de banco de dados, abrir arquivos, etc..

(3) EXECUÇÃO

- Esta fase representa todas as interações cliente x servidor. Sempre que um navegador solicita uma página JSP, o método **_jspService()** é chamado.

```
void _jspService(HttpServletRequest request,  
                  HttpServletResponse response)  
{  
    // Service handling code...  
}
```

CICLO DE VIDA JSP

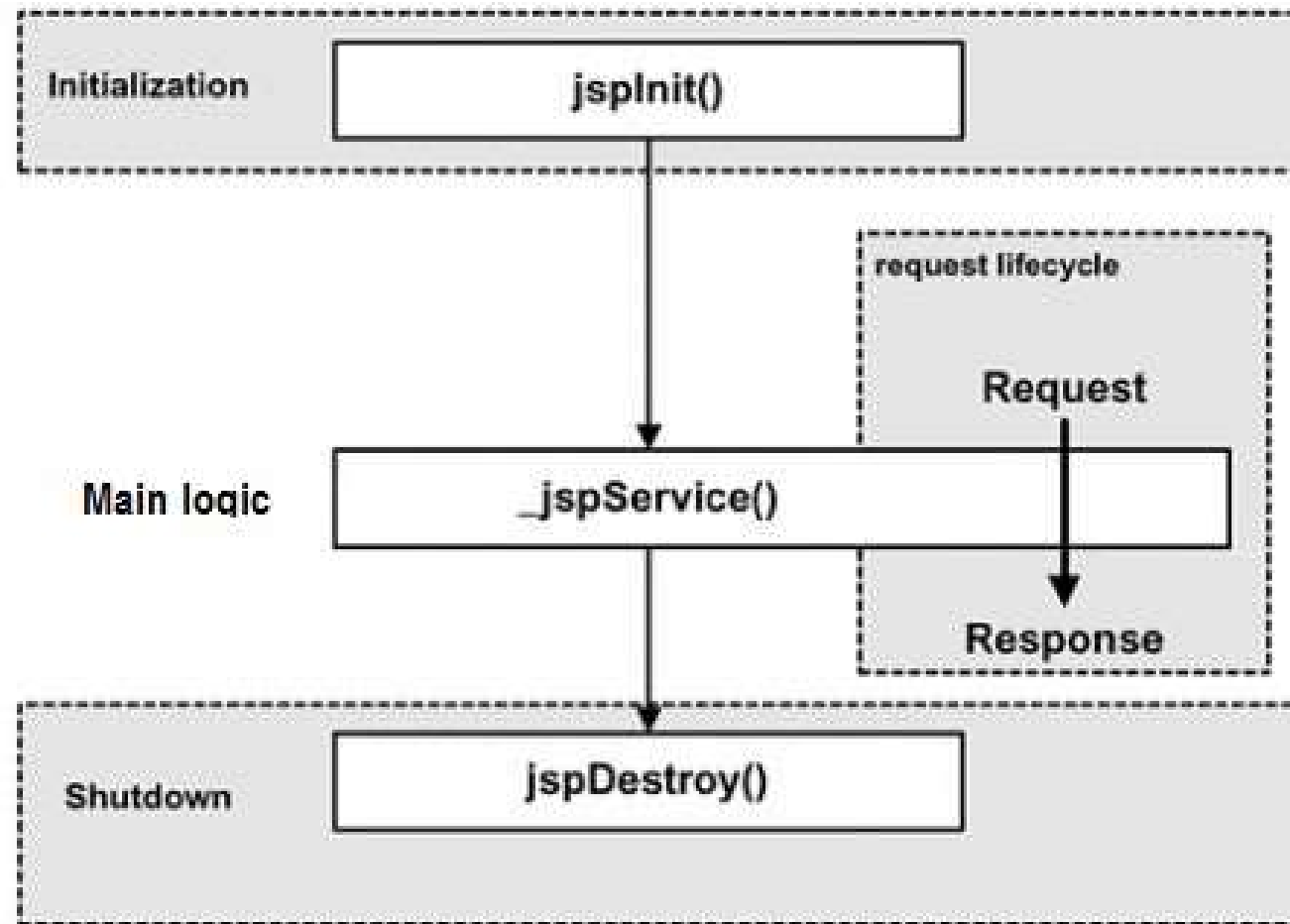
(4) DESTRUIÇÃO

- A fase de destruição é chamada quando um documento JSP deverá ser removido da memória. O método **jspDestroy()** é chamado.

```
public void jspDestroy()
{
    // Your cleanup code goes here.
}
```

CICLO DE VIDA JSP

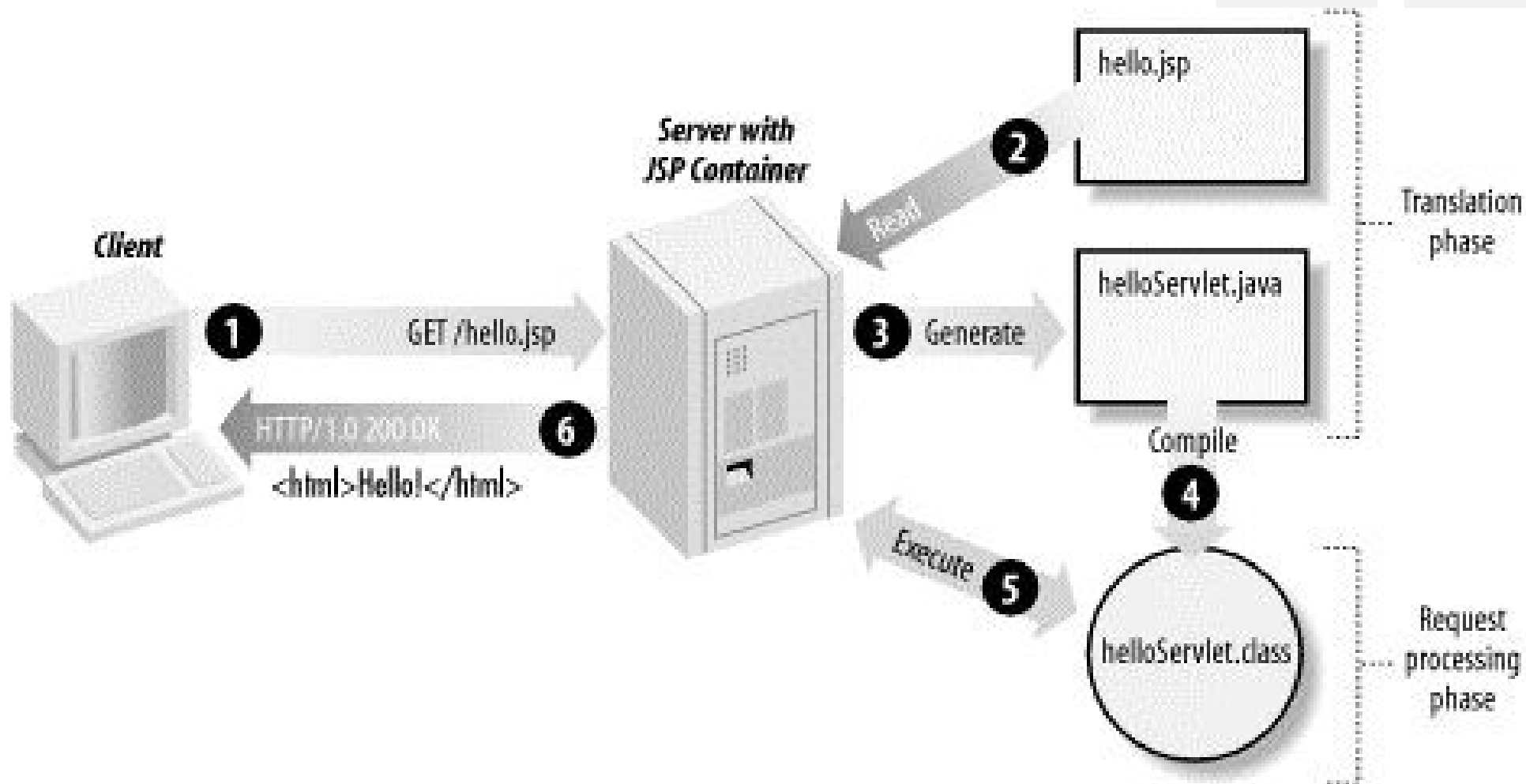
- RESUMO



https://www.tutorialspoint.com/jsp/jsp_life_cycle.htm

INTRODUÇÃO AO JSP

- CICLO DE VIDA



http://www.tutorialspoint.com/jsp/jsp_architecture.htm

INTRODUÇÃO AO JSP

CICLO DE VIDA

- Quando um **Cliente** faz uma requisição ao Componente WEB JEE '**hello.jsp**' no Servidor JEE (**Passo 1**), o container JSP retorna a página solicitada (**Passo 2**) e transforma-o em um Servlet (**helloServlet.java**) (**Passo 3**);
- O servlet em questão é compilado (**helloServlet.class**) (**Passo 4**) e executado no servidor JEE (**Passo 5**);
- Este servlet retorna uma resposta ao **Cliente**, após a chamada dos seus métodos **jspInit()** e **_jspservice()** (**Passo 6**);

Exercício

1) [COPEVE-UFAL - 2012 - ALGÁS] Dadas as afirmações a seguir, relativas às tecnologias Servlets e JSP,

- I. JSP e Servlets são tecnologias para desenvolvimento WEB com o mesmo potencial e expressividade. A diferença básica entre elas é o fato de JSP executar parte no cliente e parte no servidor.
- II. Enquanto JSP é considerada uma tecnologia do lado cliente (como JavaScript), Servlets é uma tecnologia que executa do lado servidor.
- III. Quando comparada com Servlets, a tecnologia JSP possibilita uma combinação entre HTML e Java de uma maneira muito mais facilitada, permitindo inclusive a intercalação de códigos HTML e JSP em um mesmo arquivo.
- IV. Apesar da semelhança entre Servlets e JSP, a “compilação” dos Servlets (geração de bytecode) é responsabilidade do programador, enquanto que os arquivos JSP são “compilados” pelo Servlet Container, no momento da sua primeira utilização.

Verifica-se que são(é) verdadeira(s) somente

- a) I e II. b) II. c) II e III. d) III. e) III e IV.

Exercício

1) [COPEVE-UFAL - 2012 - ALGÁS] Dadas as afirmações a seguir, relativas às tecnologias Servlets e JSP,

I. JSP e Servlets são tecnologias para desenvolvimento WEB com o mesmo potencial e expressividade. A diferença básica entre elas é o fato de JSP executar parte no cliente e parte no servidor.

II. Enquanto JSP é considerada uma tecnologia do lado cliente (como JavaScript), Servlets é uma tecnologia que executa do lado servidor.

III. Quando comparada com Servlets, a tecnologia JSP possibilita uma combinação entre HTML e Java de uma maneira muito mais facilitada, permitindo inclusive a intercalação de códigos HTML e JSP em um mesmo arquivo.

IV. Apesar da semelhança entre Servlets e JSP, a “compilação” dos Servlets (geração de bytecode) é responsabilidade do programador, enquanto que os arquivos JSP são “compilados” pelo Servlet Container, no momento da sua primeira utilização.

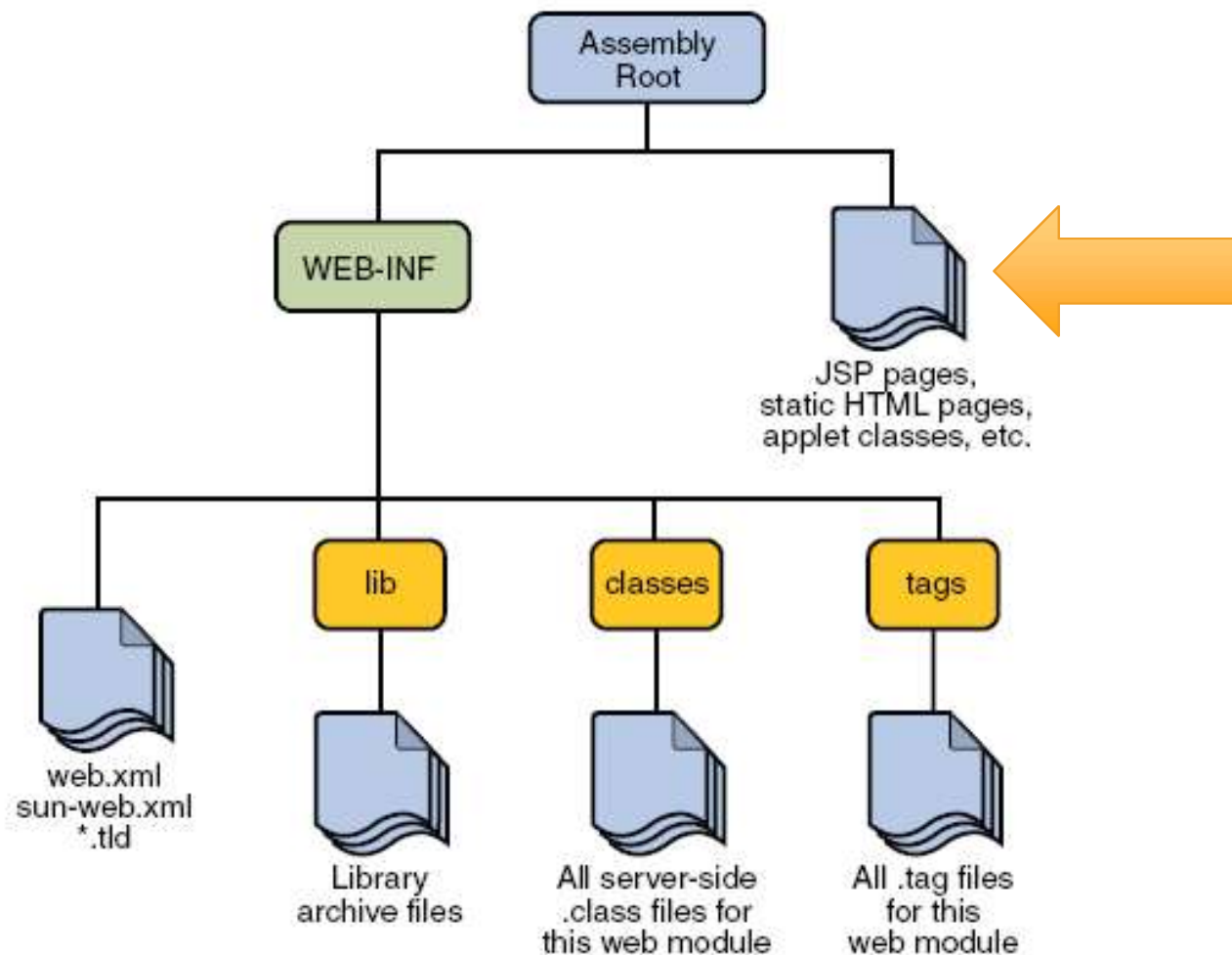
Verifica-se que são(é) verdadeira(s) somente

a) I e II. b) II. c) II e III. d) III. **e) III e IV.**

Primeiro JSP

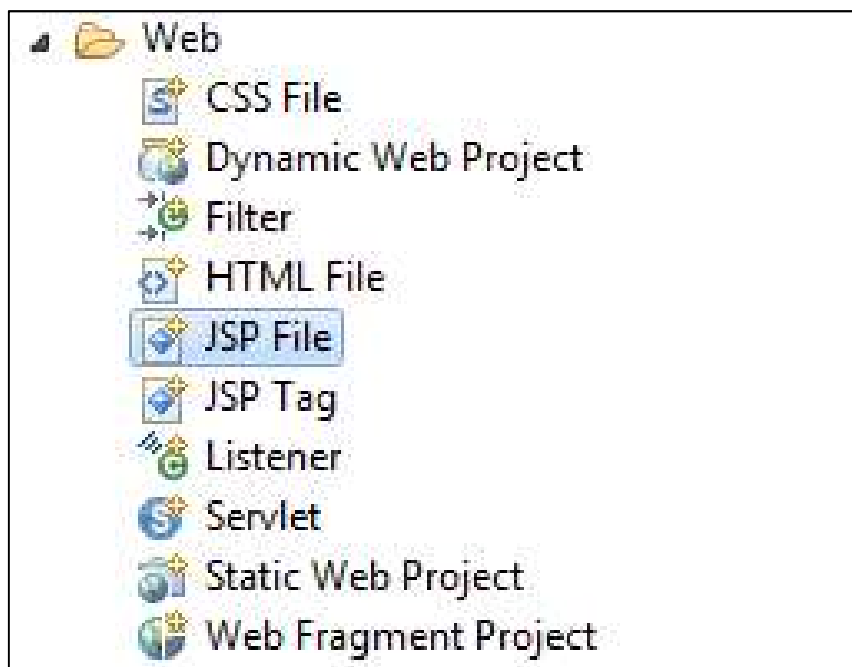
ESTRUTURA DE PROJETO WEB JEE

- Todo projeto WEB JEE segue a seguinte a estrutura de pastas:

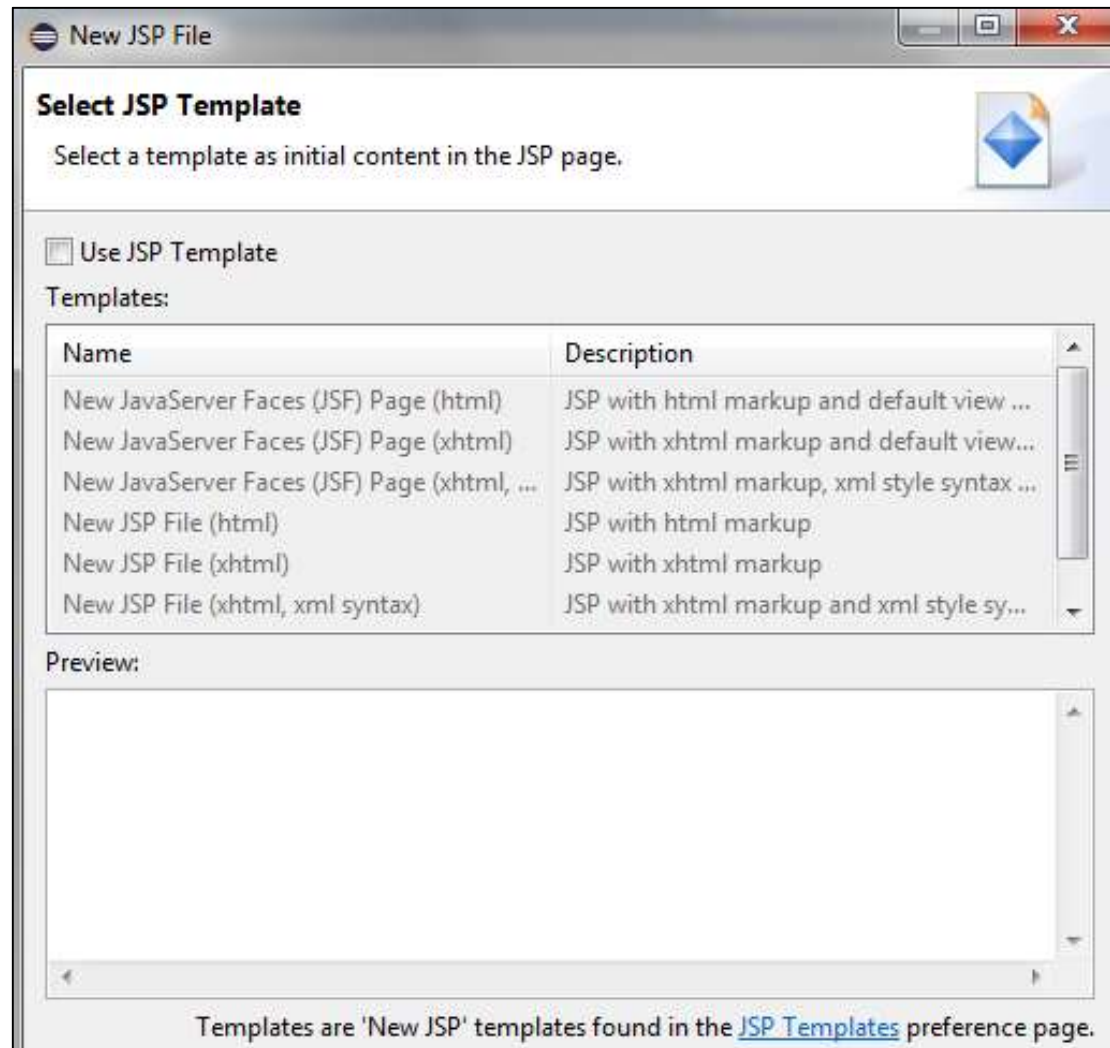


CRIAÇÃO DO PRIMEIRO JSP

- Criar o documento **data.jsp** na pasta **WebContent** do projeto **4Disciplina**.
- Selecionar a opção **“New”** ⇒ **“JSP File”**.



CRIAÇÃO DO PRIMEIRO JSP



- Selecionar a opção **Finish**.

CÓDIGO DO PRIMEIRO JSP

```
<%@ page language="java" %>
<%@ page import="java.time.LocalDate" %>
<HTML>
  <BODY>
    <p>A hora atual é <%= LocalDate.now() %></p>
    <p>2 x 5 = <% out.print(2*5); %></p>
  </BODY>
</HTML>
```

data.jsp



CÓDIGO DO SEGUNDO JSP

```
<%@ page language="java" %>
<!-- Página JSP --%>
<HTML>
<HEAD>
  <TITLE> Primeira Página JSP</TITLE>
</HEAD>
<BODY>
<% String seuNome = request.getParameter("nome");
   StringBuffer seuNomeReverso = new StringBuffer(seuNome)
     .reverse();
   out.println("<P> Seu Nome "+ seuNome +
     "lido de trás para frente é "+ seuNomeReverso); %>
</BODY>
</HTML>
```

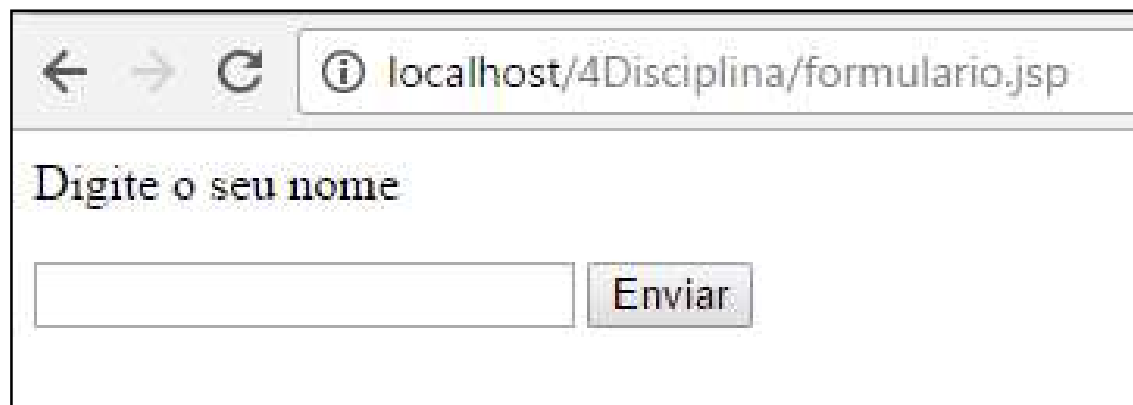
reverso.jsp

CÓDIGO DO SEGUNDO JSP

```
<HTML>
  <HEAD>
<TITLE>Formulário HTML</TITLE>
  </HEAD>
  <BODY>
    <P>Digite o seu nome</P>
    <form method="GET"
      action="http://localhost/4Disciplina/reverso.jsp">
      <input type="text" size="20" name="nome">
      <input type="submit">
    </form>
  </BODY>
</HTML>
```

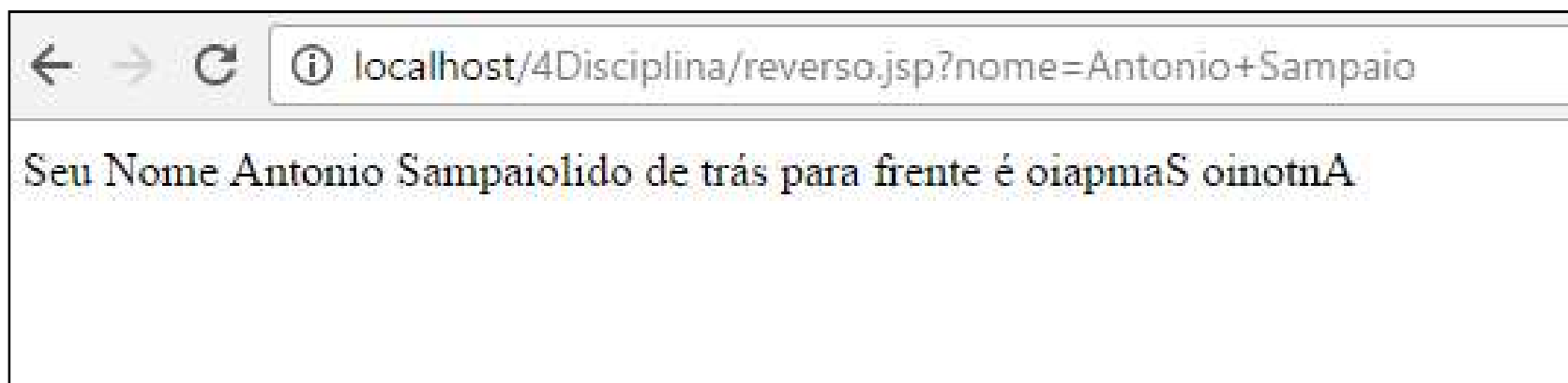
formulario.jsp

CÓDIGO DO SEGUNDO JSP



← → ↻ ⓘ localhost/4Disciplina/formulario.jsp

Digite o seu nome



← → ↻ ⓘ localhost/4Disciplina/reverso.jsp?nome=Antonio+Sampaio

Seu Nome Antonio Sampaio

lido de trás para frente é oiapmaS oinotnA

Exercício

- 1) Criar e executar os documentos **data.jsp**, **reverso.jsp** e **formulario.jsp** no projeto **4Disciplina**.

Elementos JSP

ELEMENTOS JSP

- São organizados em quatro grupos:
 - (1) Comentários
 - (2) Diretivas
 - (3) Scripting Elements
 - Expressões
 - Declarações
 - Scriptlets
 - (4) Ações

(1) COMENTÁRIOS

- Utilizados para comentar o código.
- Não são processados pelo container JSP.
- **SINTAXE:**

```
<%-- Este é um comentário --%>  
<%  
    //Este é outro comentário  
%>
```


(2) DIRETIVAS

- Utilizadas para enviar mensagens ao servidor JEE informando como o transformar um JSP no seu servlet correspondente.
- Podem ser de 03 Tipos: **include**, **page** e **taglib**.
- **SINTAXE:**

```
<%@ directiveType attributelist %>  
directiveType: {include, page e taglib}
```

(2) DIRETIVA 'INCLUDE'

- Utilizadas para incluir arquivos estáticos em uma página JSP.
- **SINTAXE:**

```
<%@ include file="URL" %> ou  
<jsp:directive.include file="URL" />
```

- **EXEMPLO:**

```
<%@ include file="reverso.jsp" %> ou  
<jsp:directive.include file="reverso.jsp" />
```

(2) DIRETIVA 'PAGE'

- Utilizada para descrever inúmeros atributos que serão considerados durante a criação do Servlet, tais como: importação de pacotes, tipo MIME de retorno, *thread safety*, etc.

- SINTAXE:**

```
<%@ page attributelist %> ou  
<jsp:directive.page attributeList />
```

- EXEMPLO:**

```
<%@ page import="java.io.*, java.util.*" %> ou  
<jsp:directive.page import="java.io.*,  
                                java.util.*" />
```

(2) DIRETIVA 'TAGLIB'

- Utilizada para incluir bibliotecas de tags.

- **SINTAXE:**

```
<%@ taglib uri="URIForLibrary"
        prefix="tagPrefix" %>
```

- **EXEMPLO:**

```
<%@ taglib uri="http://www.company.com/tags"
        prefix="public" %>

<public:loop>
....
</public:loop>
```

(3) SCRIPTING ELEMENTS

- Utilizados para inserir código Java em uma página JSP.
- Podem ser de três tipos:
 - Expressões
 - Declarações
 - Scriptlets

(3) EXPRESSÕES

- Utilizadas para inserir o valor de um atributo ou a chamada de um método.

- **SINTAXE:**

```
<%= expressão %> ou  
<jsp:expression>...</jsp:expression>
```

- **EXEMPLO:**

```
<%= new Date() %> ou  
<jsp:expression>new Date()</jsp:expression>
```

(3) DECLARAÇÕES

- Utilizadas para declarar variáveis e métodos a serem utilizados em uma página JSP.

- **SINTAXE:**

```
<%! declaração %> ou  
<jsp:declaration>...</jsp:declaration>
```

- **EXEMPLO:**

```
<%! int contador=0; %> ou  
<jsp:declaration>int contador=0;  
</jsp:declaration>
```

(3) SCRIPTLETS

- Utilizados para para incluir blocos de código Java em uma página JSP.
- **SINTAXE:**

```
<% código %> ou  
<jsp:scriptlet>...</jsp:scriptlet>
```

- **EXEMPLO:**

```
<% for(int i=0; i<contador; i++)  
    out.println(i); %> ou  
<jsp:scriptlet>  
    for(int i=0; i<contador; i++)  
        out.println(i);  
</jsp:scriptlet>
```


Exercício

- 1) Criar o documento **login.jsp** que recebe os dados **CPF** e **Senha** e fazer a autenticação do usuário via **ServletAutenticador**. Posteriormente, criar um objeto na sessão do usuário que informa se o mesmo está ou não autenticado. Posteriormente, redirecionar a navegação para uma página de sucesso (**sucesso.jsp**) ou de solicitação de nova autenticação (**login.jsp**).



The image shows a login form titled "Área do aluno" (Student Area). At the top, there is an icon of three people. Below the title, it says "Favor informar os dados abaixo" (Please provide the data below). There are two input fields: "CPF:" and "Senha:" (Password:). Below the "Senha:" field is a dark blue button labeled "Entrar" (Enter).

login.jsp

(4) AÇÕES

- São utilizadas para controlar o fluxo entre as páginas JSP, os Servlets e permitir o uso de JavaBeans.
- Podem ser classificadas em:
 - **Ações Padrões (já existentes)**
 - **Ações Personalizadas (criadas pelo usuário)**
- **São 05 os Tipos de Ações já existentes:**
 - **jsp:include**
 - **jsp:forward**
 - **jsp:plugin**
 - **jsp:param**
 - **jsp:useBean**
 - **jsp:setProperty**
 - **jsp:getProperty**

(4) AÇÕES 'INCLUDE' e 'FORWARD'

- A ação do tipo '**Include**' inclui um outro recurso na própria página JSP.
- **EXEMPLO:**

```
<jsp:include page="header.jsp">
```

- A ação do tipo '**Forward**' redireciona a requisição/resposta para outro recurso.
- **EXEMPLO:**

```
<jsp:forward page="gravacao.jsp">
```

(4) AÇÕES 'INCLUDE' e 'FORWARD'

- **EXEMPLO:**

```
<html>
<title>Index.jsp</title>
<body>
    <jsp:include page="data.jsp" />
</body>
</html>
```

index.jsp

```
<html>
<title>Index2.jsp</title>
<body>
    <jsp:forward page="data.jsp" />
</body>
</html>
```

index2.jsp

(4) AÇÃO 'USEBEAN'

- É utilizada para manipular objetos Java Beans em uma página JSP.
- **O QUE SÃO JAVA BEANS?**
- JavaBeans são componentes Java que definem a lógica da aplicação.
- Toda classe Java que possui um construtor vazio, é serializável, têm atributos privados e possui métodos get/set é chamada de Java Bean.
- **SINTAXE:**

```
<jsp:useBean id="beanName" class="com.TaxBean"  
  scope="page | request | session | application" />
```

(4) AÇÃO 'USEBEAN'

- **ESCOPO DOS JAVA BEANS**
- Pode ser de 04 Tipos:
 - **page** – o Java Bean é válido somente na página JSP.
 - **request** - o Java Bean é válido durante toda a requisição.
 - **session** – o Java Bean é válido por toda sessão do usuário.
 - **application** - o Java Bean é válido por todo contexto da aplicação.
- **EXEMPLO DO JAVABEAN:**

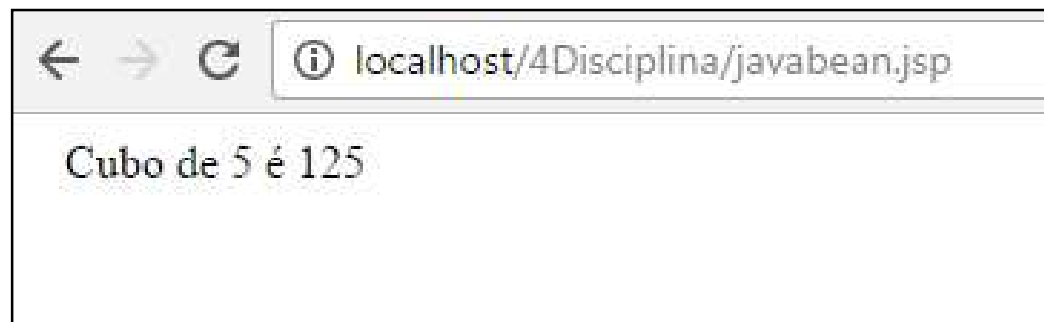
```
package unidade3;  
public class Calc{  
    public int cubo(int n){  
        return n*n*n;  
    }  
}
```

(4) AÇÃO 'USEBEAN'

- **EXEMPLO DA CHAMADA DO JAVABEAN**

```
<jsp:useBean id="obj" class="unidade3.Calc"/>
<%
    int cubo = obj.cubo(5);
    out.print("Cubo de 5 é "+cubo);
%>
```

javabeen.jsp



(4) AÇÕES 'GETPROPERTY' e 'SETPROPERTY'

- Utilizadas para manipular os atributos de um Java Bean.
- **SINTAXES:**

```
<jsp:setProperty name="beanName"  
                property="*"|"name" />
```

```
<jsp:getProperty name="beanName"  
                property="name" />
```


(4) AÇÕES 'GETPROPERTY' e 'SETPROPERTY'

- **EXEMPLO:**

```
<html>
  <body>
    <form action="process.jsp" method="post">
      Nome:<input type="text" name="nome"><br>
      Senha:<input type="password" name="senha"><br>
      Email:<input type="text" name="email"><br>
      <input type="submit" value="register">
    </form>
  </body>
</html>
```

index.jsp

(4) AÇÕES 'GETPROPERTY' e 'SETPROPERTY'

- **EXEMPLO:**

```
<jsp:useBean id="u" class="com.Usuario">  
</jsp:useBean>  
<jsp:setProperty property="*" name="user"/>
```

Registro:


```
<jsp:getProperty property="nome" name="user"/><br>  
<jsp:getProperty property="senha" name="user"/><br>  
<jsp:getProperty property="email" name="user"/><br>
```

process.jsp

(4) AÇÕES 'GETPROPERTY' e 'SETPROPERTY'

- **EXEMPLO:**

```
package com;  
  
public class Usuario {  
    private String name,password,email;  
    //setters and getters  
}
```

Exercícios

- 1) Criar os documentos **index.jsp** e **index2.jsp** no projeto **4Disciplina** e identificar a diferença no uso das ações **include** e **forward**.
- 2) Criar o JavaBean **unidade3.Calc** e a página que irá chamá-lo **javabeam.jsp**.
- 3) Criar o documento **autenticacao.jsp** que irá realizar todas as operações executadas pelo **ServletAutenticador**. Posteriormente, alterar o documento **login.jsp** para fazer a autenticação do usuário via documento jsp.

Objetos Implícitos e Tratamento de Exceção

OBJETOS IMPLÍCITOS

- Toda página JSP importa automaticamente os seguintes pacotes:
java.lang.*; java.servlet.*; javax.servlet.jsp.*; javax.servlet.http.*.
- Também pode acessar objetos Java que são criados automaticamente, tais como:
 - **application** (contexto do servlet), **pageContext** (atributos da página) e **config** (configuração do servlet)
 - **request** (requisição recebida) e **response** (resposta enviada)
 - **session** (sessão criada) e **out** (resposta do cliente)
 - **page** (representa uma instância do servlet)
 - **exception** (representa os erros)
- Estes objetos são chamados **objetos implícitos**, pois não é necessário declará-los explicitamente.
- O JSP suporta nove objetos implícitos.

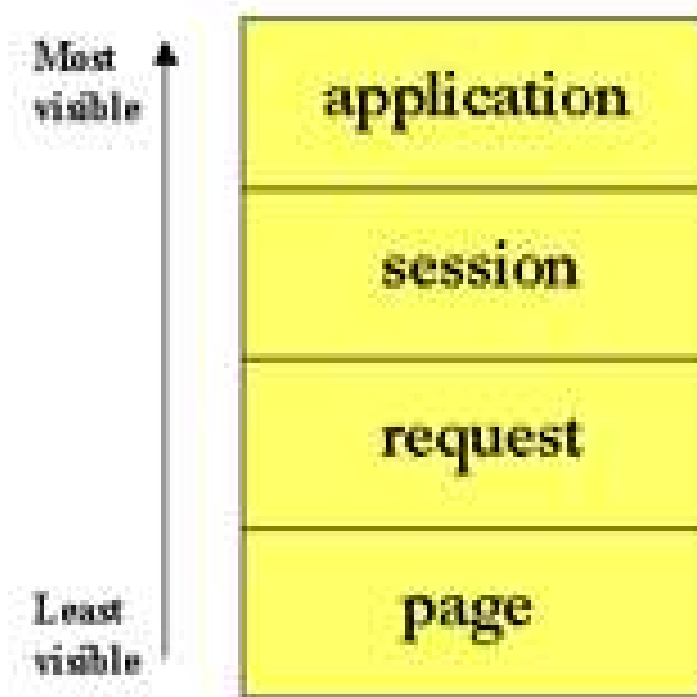
OBJETOS IMPLÍCITOS

Objeto	Classe ou Interface	Descrição
page	<code>javax.servlet.jsp.HttpJspPage</code>	Instância de servlet da página
config	<code>javax.servlet.ServletConfig</code>	Dados de configuração de servlet
request	<code>javax.servlet.http.HttpServletRequest</code>	Dados de solicitação, incluindo parâmetros
response	<code>javax.servlet.http.HttpServletResponse</code>	Dados de resposta
out	<code>javax.servlet.jsp.JspWriter</code>	Fluxo de saída para conteúdo da página
session	<code>javax.servlet.http.HttpSession</code>	Dados de sessão específicos de usuário
application	<code>javax.servlet.ServletContext</code>	Dados compartilhados por todas as páginas de aplicação
pageContext	<code>javax.servlet.jsp.PageContext</code>	Dados de contexto para execução da página
exception	<code>javax.lang.Throwable</code>	Erros não capturados ou exceção

© Ulisses Telemaco Neto

ESCOPO DOS OBJETOS

- Uma página JSP pode manipular diversos tipos de objetos, associando-os a um atributo de escopo.
- **São 04 os Escopos dos Objetos:**
 - **Page:** objetos acessíveis somente nas páginas em que foram criados;
 - **Request:** objetos acessíveis nas páginas que processam a mesma requisição;
 - **Session:** objetos acessíveis nas páginas que pertencem à mesma sessão em que os objetos foram criados;
 - **Application:** objetos acessíveis nas páginas que pertençam à mesma aplicação.



ESCOPO DOS OBJETOS

- A persistência das informações depende do escopo dos objetos onde elas estão disponíveis
- Métodos de **pageContext** permitem setar ou buscar atributos em qualquer objeto de escopo:

```
setAttribute(nome, valor, escopo)  
getAttribute(nome, escopo)
```

- Constantes da classe **javax.servlet.jsp.PageContext** identificam o escopo de objetos:

```
pageContext PageContext.PAGE_SCOPE  
request PageContext.REQUEST_SCOPE  
session PageContext.SESSION_SCOPE  
applicaJon PageContext.APPLICATION_SCOPE
```

OBJETOS IMPLÍCITOS

- **EXEMPLO:**

```
<%@ page import="java.util.Date"%>
<h3>Informações do Servidor:</h3>
<%= application.getServerInfo() %>
<h3>Versao principal da API do servlet:</h3>
<%= application.getMajorVersion() %>
<h3>Informações do Host:</h3>
<%= request.getRemoteHost() %>
<h3>Informações da Sessão:</h3>
<%= session.getId() %>
<h3>Data de criação:</h3>
<%= new Date(session.getCreationTime()) %>
<h3>Data de criação:</h3>
<%= new Date(session.getLastAccessedTime()) %>
```

informacao.jsp

TRATAMENTO DE EXCEÇÃO

- É possível tratar exceções em tempo de execução de 02 maneiras:
 - usando os blocos **try...catch**
 - usando o objeto **exception**
- O objeto de **exceção** é uma instância de uma subclasse de Throwable (por exemplo, java.lang.NullPointerException) e só está disponível em páginas de erro.
- O JSP oferece uma opção para referenciar uma página de erro, por exemplo erro.jsp, sempre que uma página JSP provocar uma exceção.
- Para configurar uma página de erro é necessário utilizar a diretiva **<%@ page errorPage = “erro.jsp” %>**.

TRATAMENTO DE EXCEÇÃO

- **EXEMPLO:**

```
<%@ page errorPage="erro.jsp" %>
<%
    // Throw uma exceção para chamar erro.jsp
    int x = 1;
    if (x == 1)
    {
        throw new RuntimeException("Erro!");
    }
%>
```

pagina.jsp

TRATAMENTO DE EXCEÇÃO

- **EXEMPLO:**

```
<%@ page isErrorPage="true" %>
<html>
  <head>
    <title>Página de Erro</title>
  </head>
  <body>
    <pre>
      <% exception.printStackTrace
        (response.getWriter()); %>
    </pre>
  </body>
</html>
```

erro.jsp

Exercícios

- 1) Criar o documento **informacao.jsp** e testar o uso de objetos implícitos.
- 2) Criar o documento **pagina.jsp** e **erro.jsp** para testar o tratamento de exceção com o uso do objeto implícito **exception**.

Expression Language (EL)

INTRODUÇÃO À EL

- Expression Language (EL) foi introduzida no JSP 2.0 para facilitar o acesso aos dados de JavaBeans e aos Objetos Implícitos.
- **Com ela é possível:**
 - (1) Acesso conciso a objetos armazenados em escopos;

```
${aluno}
```

- (2) Notação resumida para propriedade de JavaBeans;

```
${aluno.nome}
```

- (3) Acesso simplificado para objetos de coleção;

```
${alunos[0]}
```


INTRODUÇÃO À EL

- **Com ela é possível:**

- (4) Acesso sucinto para parâmetros de requisição, cookies e outros dados de requisição.

```
${param.nome}
```

```
${cookie.nomeCookie.value }
```

- (5) Uso de operadores aritméticos, relacionais e lógicos.

INTRODUÇÃO À EL

Operador	Descrição	Exemplo	Resultado
==	Igualdade	<code>\${5 == 5}</code>	true
!=	Desigualdade	<code>\${5 != 5}</code>	false
<	Menor que	<code>\${5 < 7}</code>	true
>	Maior que	<code>\${5 > 7}</code>	false
<=	Menor ou igual que	<code>\${5 le 5}</code>	true
>=	Maior ou igual que	<code>\${5 ge 6}</code>	false
Empty	Checa se um parâmetro está vazio	<code>\${user.lastname}</code>	depende
And &&	E	<code>\${param.month == 5 and param.day == 25}</code>	depende
Or	OU	<code>\${param.month == 5 or param.month == 6}</code>	depende
+	soma	<code>\${4 + 5}</code>	9
! not	Negação	<code>\${not true}</code>	false

INTRODUÇÃO À EL

- Com o uso de EL as operações são simplificadas e a exibição de dados se torna mais fácil.

```
<html>
  <body>
    ${1<2}
    ${1+2+3}
    ${10.2 + 20.3}
    ${50 mod 8}
    ${ (10==20) ? "true" : "false"}
    ${param.cpf}
    ${param.senha}
  </body>
</html>
```

el.jsp

EL e JAVABEANS

- Um dos principais usos do EL está na manipulação de objetos JavaBeans.
- **EXEMPLO DO JAVABEAN:**

```
package unidade3;
public class Cliente {
    private long cpf;
    private String nome,email;
    //setters e getters
    public long getCpf() {
        return cpf;
    }
    public void setCpf(long cpf) {
        this.cpf = cpf;
    }
    ...
}
```

EL e JAVABEANS

- **PÁGINA JSP**

```
<jsp:useBean id="cliente"
              class="unidade3.Cliente"/>

<%
    cliente.setCpf(123);
    cliente.setNome("Antonio");
    cliente.setEmail("assoftbel@gmail.com");
    request.setAttribute("cliente", cliente);
%>
<h3>Cpf:${cliente.cpf}</h3>
<h3>Nome:${cliente.nome}</h3>
<h3>Email:${cliente.email}</h3>
```

cliente.jsp

Exercícios

- 1) Criar o documento **el.jsp** e testar o uso de EL.
- 2) Criar o documento **cliente.jsp** que faz acesso ao JavaBean **unidade3.Cliente**.

JSTL

(JSP Standard Tag Library)

JSTL

- Representa um conjunto de **tags** que simplificam o desenvolvimento de páginas JSP.
- **VANTAGENS**
 - **Maior agilidade no desenvolvimento de páginas JSP;**
 - **Reusabilidade de código;**
 - **Sem necessidade de usar scriptlets.**
- Uma página JSP pode conter várias **tags** JSTL.
- Cada **tag** realiza um determinado tipo de processamento (equivalente a um código Java dentro de JSP).
- Para cada **tag** JSTL, há uma biblioteca JSTL associada.

BIBLIOTECAS JSTL

Área	Sub-Função	Prefixo
Core	Suporte a Variáveis	c
	Controle de Fluxo	
	Gerenciamento de URL	
XML	Core	x
	Controle de Fluxo	
	Transformação	
Internacionalização	Local	fmt
	Formatação de Mensagem	
	Formatação de Datas e Números	
Banco de Dados	SQL	sql
Funções	Tamanho de Coleções	fn
	Manipulação de Strings	

© Andre Luis Schwerz

BIBLIOTECAS JSTL

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

© Andre Luis Schwerz

- A versão atual da JSTL é a 1.2 e pode ser encontrada no site:
<http://www.java2s.com/Code/Jar/j/Downloadjstl12jar.htm>
- Fazer o download do arquivo **jstl-1.2.jar.zip**.



[jstl/jstl-1.2.jar.zip\(367 k\)](#)



Related examples in the same category

1. [Download jstl-api-1.2-sources.jar](#)



2. [Download jstl-connector-3.0-prelude-embedded-m2-sources.jar](#)



JSTL-1.2.jar

- Descompactar o arquivo **jstl-1.2.jar.zip** e copiar o arquivo **jstl-1.2.jar** para a pasta **Tomcat8.0/lib**.

Local Disk (C:) > Program Files > Apache Software Foundation > Tomcat 8.0 > lib

<input type="checkbox"/>	Name	Date modified	Type	Size
<input type="checkbox"/>	annotations-api.jar	09/11/2016 05:50	Executable Jar File	18 KB
<input type="checkbox"/>	catalina.jar	09/11/2016 05:50	Executable Jar File	1.536 KB
<input type="checkbox"/>	catalina-ant.jar	09/11/2016 05:50	Executable Jar File	53 KB
<input type="checkbox"/>	catalina-ha.jar	09/11/2016 05:50	Executable Jar File	117 KB
<input type="checkbox"/>	catalina-storeconfig.jar	09/11/2016 05:50	Executable Jar File	74 KB
<input type="checkbox"/>	catalina-tribes.jar	09/11/2016 05:50	Executable Jar File	270 KB
<input type="checkbox"/>	ecj-4.5.1.jar	09/11/2016 05:50	Executable Jar File	2.338 KB
<input type="checkbox"/>	el-api.jar	09/11/2016 05:50	Executable Jar File	80 KB
<input type="checkbox"/>	jasper.jar	09/11/2016 05:50	Executable Jar File	572 KB
<input type="checkbox"/>	jasper-el.jar	09/11/2016 05:50	Executable Jar File	158 KB
<input type="checkbox"/>	jsp-api.jar	09/11/2016 05:50	Executable Jar File	61 KB
<input checked="" type="checkbox"/>	jstl-1.2.jar	21/03/2017 21:19	Executable Jar File	405 KB
<input type="checkbox"/>	ojdbc6.jar	28/01/2015 14:17	Executable Jar File	2.676 KB

JSTL CORE

- **TAGS**

- **<c:out >**
- **<c:set >**
- **<c:remove >**
- **<c:catch>**
- **<c:if>**
- **<c:choose>**
- **<c:when>**
- **<c:otherwise >**
- **<c:import>**
- **<c:forEach >**
- **<c:forTokens>**
- **<c:param>**
- **<c:redirect >**
- **<c:url>**

JSTL CORE

- **EXEMPLO DE USO**

```
<%@ taglib prefix="c"
    uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body>
    <c:forEach var="i" begin="1" end="10" step="1">
        <c:out value="${i}" /> <br />
    </c:forEach>
</body>
</html>
```

jstlcore.jsp

JSTL FMT

- **TAGS**
 - **<Fmt: formatNumber>**
 - **<Fmt: parseNumber>**
 - **<Fmt: formatDate>**
 - **<Fmt: parseDate>**
 - **<Fmt: bundle>**
 - **<Fmt: setLocale>**
 - **<Fmt: setBundle>**
 - **<Fmt: timeZone>**
 - **<Fmt: setTimeZone>**
 - **<Fmt: message>**
 - **<Fmt: requestEncoding>**

JSTL FMT

- **EXEMPLO DE USO**

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
  <body bgcolor="#FFFFFF">
    <c:set var="agora" value="<%=new java.util.Date()%>" />
    <p>Hora: <strong><fmt:formatDate type="time"
                                   value="{agora}" /></strong></p>
    <p>Dia: <strong><fmt:formatDate type="date"
                                   value="{agora}" /></strong></p>
  </body>
</html>
```

jstlfmt.jsp

JSTL XML

- **TAGS**

- **<x:out>**
- **<x:parse>**
- **<x:set >**
- **<x:if >**
- **<x:forEach>**
- **<x:choose>**
- **<x:when >**
- **<x:otherwise >**
- **<x:transform >**
- **<x:param >**

JSTL XML

- **EXEMPLO DE USO**

```
<clientes>
  <cliente>
    <cpf>123.456.789-00</cpf>
    <nome>Antonio Sampaio Jr</nome>
    <email>assoftbel@gmail.com</email>
  </cliente>
</clientes>
```

clientes.xml

JSTL XML

- **EXEMPLO DE USO**

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
<html>
<body>
<h3>Informações:</h3>
<c:import var="clienteInfo"
    url="http://localhost/4Disciplina/unidade3/clientes.xml"/>
<x:parse xml="${clienteInfo}" var="output"/>
<b>CPF</b>:
<x:out select="$output/clientes/cliente/cpf" />
<br>
<b>Nome</b>:
<x:out select="$output/clientes/cliente/nome" />
<br>
</body>
</html>
```

JSTL SQL

- **TAGS**
 - **<Sql: setDataSource>**
 - **<Sql: query>**
 - **<Sql: update>**
 - **<Sql: param>**
 - **<Sql: dateParam>**
 - **<Sql: transaction>**

JSTL SQL

- **EXEMPLO DE USO**

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<sql:setDataSource var="bd" driver="oracle.jdbc.driver.OracleDriver"
    url="jdbc:oracle:thin:@localhost:1521:XE"
    user="cursojava" password="schema"/>
<sql:query dataSource="${bd}" var="result">
SELECT * from Login
</sql:query>
<table border="1" width="100%">
<tr>
    <th>CPF</th><th>Senha</th>
</tr>
<c:forEach var="row" items="${result.rows}">
<tr>
    <td><c:out value="${row.cpf}" /></td>
    <td><c:out value="${row.senha}" /></td>
</tr></c:forEach>
</table>
```

JSTL FN

- **TAGS**

- **fn:contains()**
- **fn:containsIgnoreCase()**
- **fn:endsWith()**
- **fn:escapeXml()**
- **fn:indexOf()**
- **fn:join()**
- **fn:length()**
- **fn:replace()**
- **fn:split()**
- **fn:startsWith()**
- **fn:substring()** **fn:substringAfter()**
- **fn:substringBefore()** **fn:toLowerCase()**
- **fn:toUpperCase()**
- **fn:trim()**

JSTL FN

- **EXEMPLO DE USO**

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>
<html>
<body>
  <c:set var="string1" value="Antonio Sampaio Jr."/>
  <c:set var="string2" value="Professor do Curso Java8." />
  <p>Tamanho da String (1) : ${fn:length(string1)}</p>
  <p>Tamanho da String (2) : ${fn:length(string2)}</p>
</body>
</html>
```

jstlfn.jsp

JSTL E COLEÇÕES

- É muito comum acessar objetos Java de uma coleção fazendo uso das bibliotecas JSTL.

- **EXEMPLO**

```
package unidade3;

public class Clientes {
    private List<String> nomes = new ArrayList<String>();
    public Clientes() {
        nomes.add("Antonio");
        nomes.add("Benedito");
        nomes.add("Coimbra");
    }
    public List<String> getNomes() {return nomes;}
    public void setNomes(List<String> nomes) {
        this.nomes = nomes;
    }
}
```

JSTL E COLEÇÕES

- É muito comum acessar objetos Java de uma coleção fazendo uso das bibliotecas JSTL.

- **EXEMPLO**

```
<%@ taglib prefix="c"
      uri="http://java.sun.com/jsp/jstl/core" %>
<html>
  <body bgcolor="#FFFFFF">
    <jsp:useBean id="colecacao"
                  class="unidade3.Clientes"/>
    <c:forEach var="nome" items="${colecacao.nome}">
      <c:out value="${nome}" /> <br />
    </c:forEach>
  </body>
</html>
```

jstlcolecoes.jsp

Exercício

1) Criar as seguintes páginas JSP:

- **jstlcore.jsp**
- **jstlfmt.jsp**
- **jstlxml.jsp**
- **jstlsql.jsp**
- **jstlfn.jsp**
- **jstlcolecoes.jsp**

Aprender JSP ainda é útil?

Aprender JSP ainda é útil?

- O JSP é simples de trabalhar. Contudo, pode ser difícil manter programas com muito código Java e Html integrados.
- A tecnologia JSP era utilizada como a tecnologia de apresentação para Java Server Faces (JSF), contudo não suporta todas as novas funcionalidades disponíveis no JSF 2.0.
- Na documentação JEE 7, a tecnologia JSP é considerada obsoleta para se trabalhar com o JSF 2.0.
- Agora, a tecnologia Facelets, disponível como parte do JSF 2.0, é a tecnologia de apresentação padrão para a construção de aplicações JSF.

Aprender JSP ainda é útil?

- O JSF é um framework MVC próprio para a criação de projetos WEB em Java.
- O JSP é apenas uma tecnologia de apresentação, portanto, não está correto comparar JSP com JSF, pois esta última não é uma substituição do JSP.
- Para concluir, o JSP só estará em desuso se a tecnologia JSF for utilizada plenamente em seus recursos. Contudo, para vários frameworks baseados exclusivamente nas tecnologias Java Servlets e JSP, como por exemplo o Spring MVC, esta última ainda será muito útil.

Projeto Prático (Parte 3)

Projeto Prático

Algumas Diretrizes:

- 1. Necessário autenticação para acessar o **Sistema de Gerenciamento de Cursos**.



A login form with a light gray background. It contains two text input fields: the first is labeled 'CPF:' and the second is labeled 'Senha:'. Below the 'Senha:' field is a dark gray button with the text 'Entrar' in white.

login.jsp

- 2. Proibir o acesso a qualquer página sem realizar essa autenticação.
- 3. Realizar todas as operações de CRUD no banco de dados com JPA/Hibernate.
- 4. Criar a página “ConsultarTodos” para **Clientes, Cursos e Pagamentos**.

Exercício

1) Para atender as 04 Diretrizes informadas, realizar os passos abaixo no **ProjetoWEB:**

a) Implantar a Autenticação no Projeto

- Alterar o **web.xml** (única página de entrada: login.jsp)
- Copiar a página **/cliente/cadastro.html** para a raiz do projeto e alterar o seu nome para **login.jsp**
- Fazer as alterações necessárias (caminhos e campos)
- Incluir a página **autenticacao.jsp** no projeto
- Criar a página **logout.jsp**

b) Proibir o acesso a qualquer página sem realizar a Autenticação

- Alterar o formato (de html para jsp) todas as páginas de **Clientes, Cursos e Pagamentos.**
- Criar a página **topo.jsp** que verificar se o usuário está autenticado
- Incluir a página **topo.jsp** em todas as páginas jsp de **Clientes, Cursos e Pagamentos.**
- Alterar as referências às antigas páginas html nos arquivos **index.jsp**

Exercício

- 1) Para atender as 04 Diretrizes informadas, realizar os passos abaixo no **ProjetoWEB**:
 - c) **Realizar as operações de CRUD no BD**
 - Importar os pacotes **jpa/hibernate (...\\hibernate-release-5.2.7.Final\\lib\\required)** para a pasta lib do Tomcat
 - Copiar o pacote **com.abctreinamentos** do projeto **JPAApp** da **3Disciplina**
 - Copiar os arquivos **hibernate.cfg.xml** e **persistence.xml** para a pasta **src/META-INF** do projeto
 - Fazer as alterações necessárias no Servlet **Controlador**
 - d) **Criar as páginas ConsultarTodos para Clientes, Cursos e Pagamentos**

RESUMO

TÓPICOS APRESENTADOS

- Nesta aula nós estudamos:
 - **Introdução ao JSP**
 - **Primeiro JSP**
 - **Elementos JSP**
 - **Objetos Implícitos e Tratamento de Exceção**
 - **Expression Language (EL)**
 - **JSTL (JSP Standard Tag Library)**
 - **Aprender JSP ainda é útil?**
 - **Projeto Prático (Parte 3)**

ATIVIDADES PARA SE APROFUNDAR

- **1) Testar exaustivamente o ProjetoWEB para encontrar problemas. Proceder com as correções necessárias.**
- **2) Fazer um estudo dos principais frameworks Web utilizados na atualidade, listando as suas principais características e identificando quais são os Componentes WEB JEE utilizados.**