

# Desenvolvimento de Aplicações WEB

## *Java Enterprise Edition*

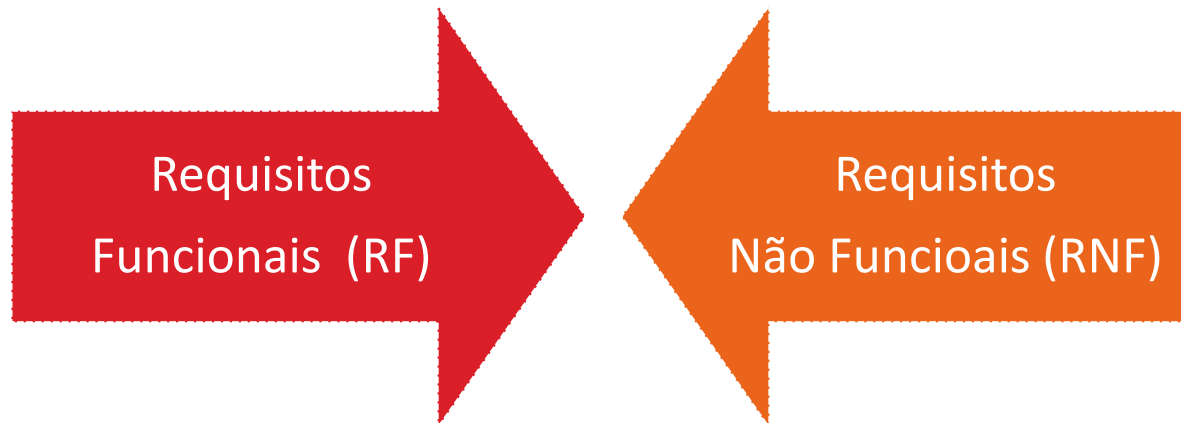
Profa. Joyce Miranda

Referência: <http://www.caelum.com.br/apostila-java-web>

## Java Enterprise Edition

---

### ► Especificação de aplicações web



### ► RNF aplicações web

#### ► Serviços de infraestrutura

- ❑ Gerenciamento de requisição HTTP, Gerenciamento de sessão, Persistência de Dados...

## Java Enterprise Edition

---

### ► Especificação de aplicações web

#### ► JEE

- É um conjunto de especificações que definem como alguns serviços de infraestrutura devem ser implementados.
- Reduz o custo e a complexidade do desenvolvimento.



## Java Enterprise Edition

### ► Algumas especificações do JEE

API	Função
JavaServer Pages (JSP) Java Servlets Java Server Faces (JSF)	Funcionalidades para aplicações web
Enterprise Javabeans (EJB) Java Persistence API (JPA)	Objetos distribuídos, clusters, acesso remoto.
Java API for XML Web Services (JAX-WS) Java API for XML Binding (JAX-B)	Trabalhar com arquivos XML.
Java Authentication and Authorization Service (JAAS)	API padrão do Java para segurança.
Java Transaction API (JTA)	Controle de transação no contêiner.
Java Message Service (JMS)	Troca de mensagens síncronas ou não.
Java Naming and Directory Interface (JNDI)	Espaço de nomes e objetos.
Java Management Extensions (JMX)	Administração e estatísticas da aplicação.

## Java Enterprise Edition

---

### ► **Servidor de Aplicação**

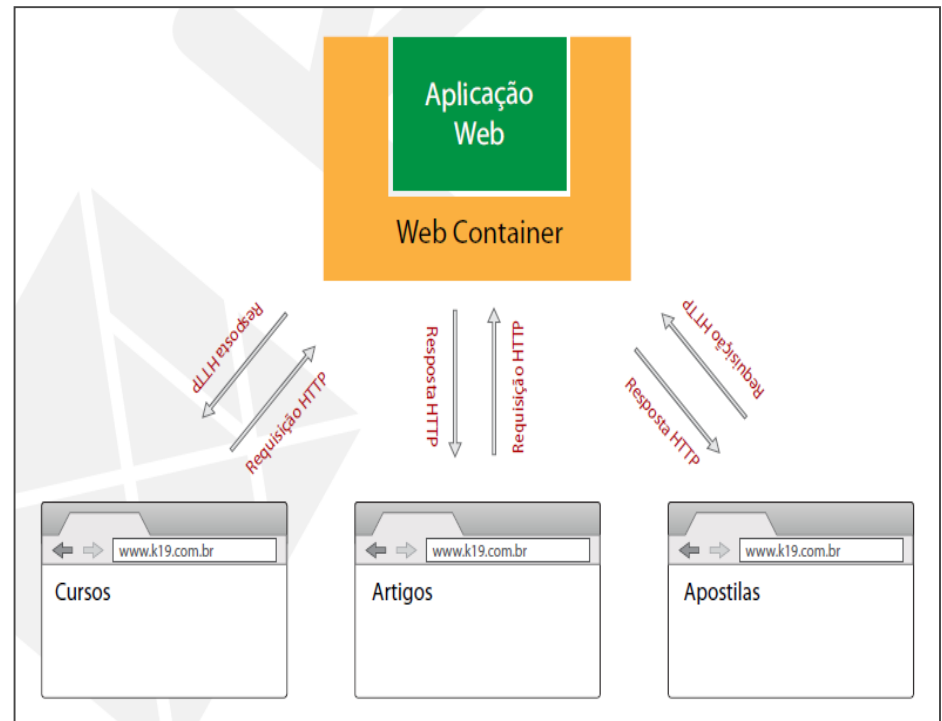
- Software utilizado para servir sua aplicação com serviços de infra-estrutura implementados.
- Exemplos:
  - *Jboss Application Server*
  - *Apache Geronino*
  - *GlassFish*



# Java Enterprise Edition

## ► Web Container (Servlet Container)

- Se restringe à implementação de funcionalidades para aplicações web
- Responsabilidades:
  - Envio e recebimento de mensagens HTTP;
  - Acesso simultâneo;
  - Conteúdo dinâmico.
- Aplicações Web devem ser implantadas em um web container



## Java Enterprise Edition

---

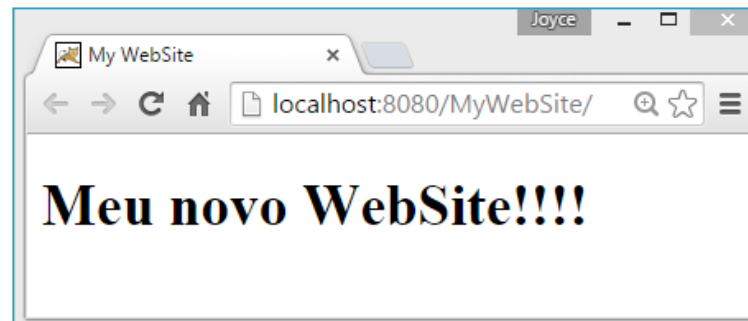
### ▶ **Servlet Container**

- ▶ Se limita a especificações JEE para aplicações web:
  - ▶ JSP
  - ▶ Servlets
  - ▶ JSTL
  - ▶ JSF
- ▶ Exemplos
  - ▶ Apache Tomcat
  - ▶ Jetty
- ▶ Servidores de Aplicação: JBoss, Glassfish podem ser usados pois possuem um web container interno

## Java Enterprise Edition

PROGRAMAÊ!

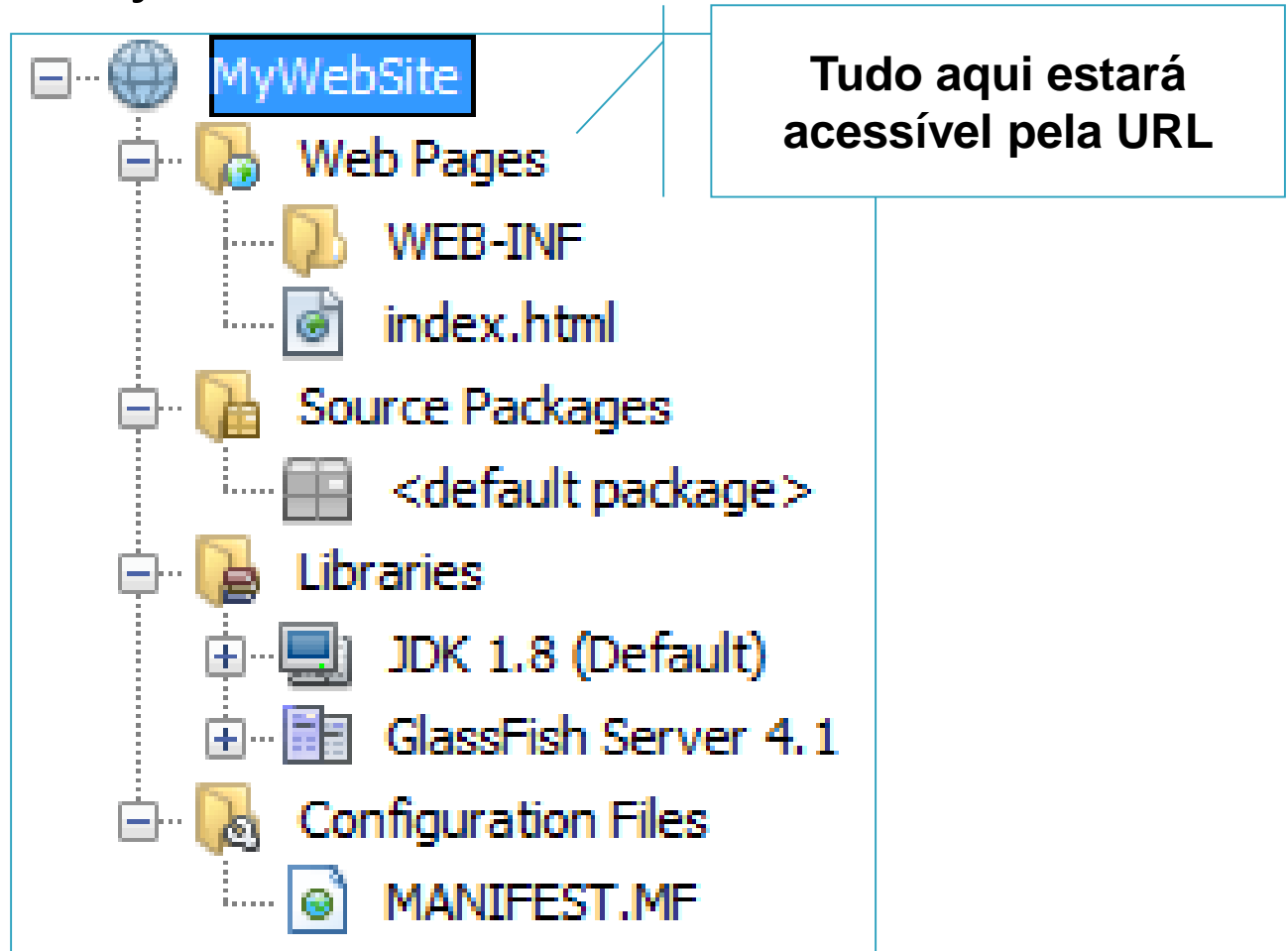
- ▶ Disponibilização da página no *web container*
  - ▶ IDE Netbeans + GlassFish
  - ▶ Crie um Projeto JAVA WEB: MyWebSite
    - ▶ Crie | Edite index.html
  - ▶ Execute o projeto
    - ▶ Inicie | Reinicie o servidor de aplicação (GlassFish)
  - ▶ Acesse
    - ▶ <http://localhost:8080/MyWebSite/>





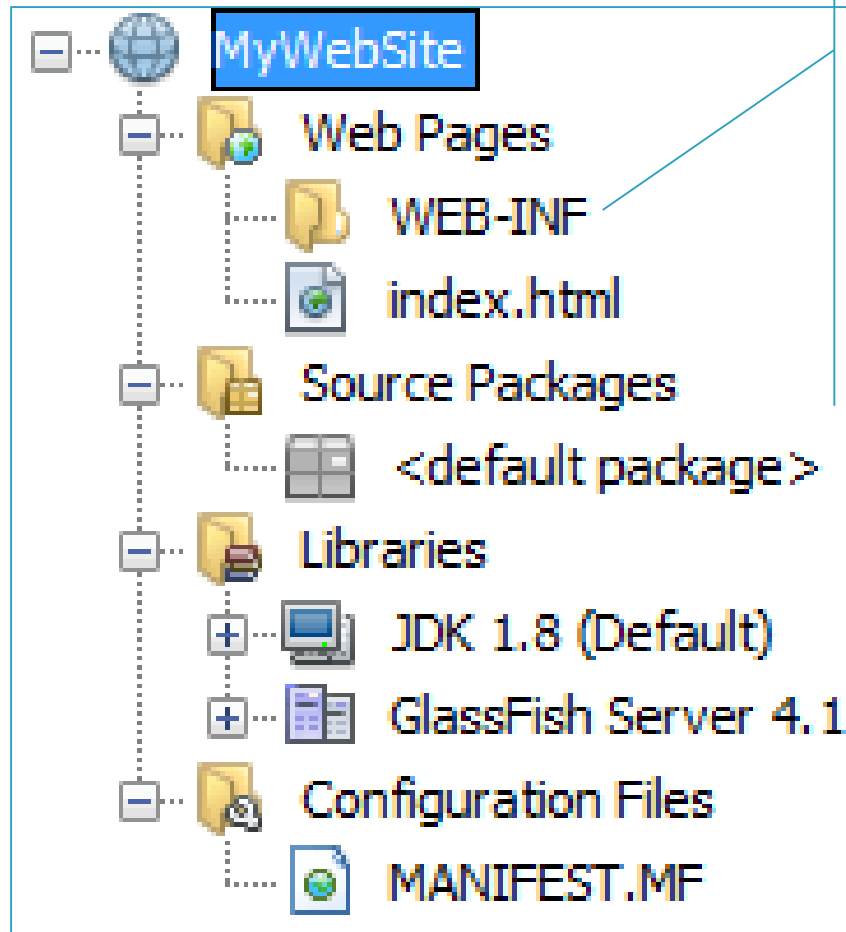
## Java Enterprise Edition

### ► Estrutura do Projeto WEB no Netbeans



## Java Enterprise Edition

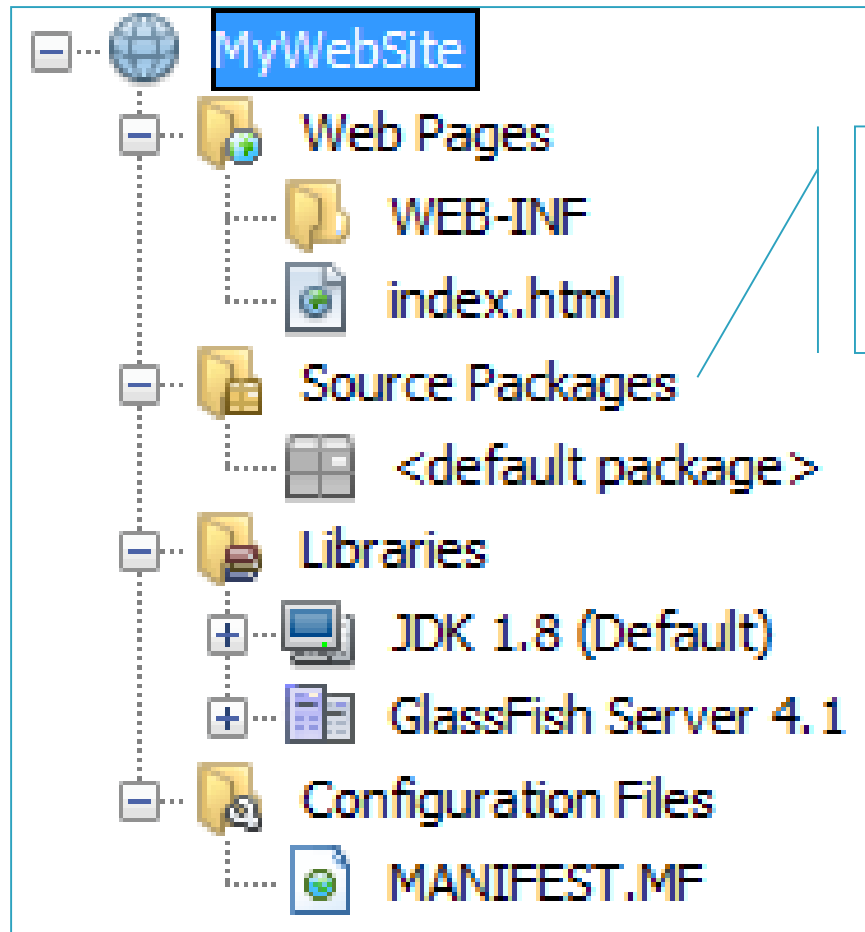
### ► Estrutura do Projeto WEB no Netbeans



**Pasta oculta.**  
**Contém**  
**configurações e**  
**recursos para o**  
**projeto rodar no**  
**servidor**

## Java Enterprise Edition

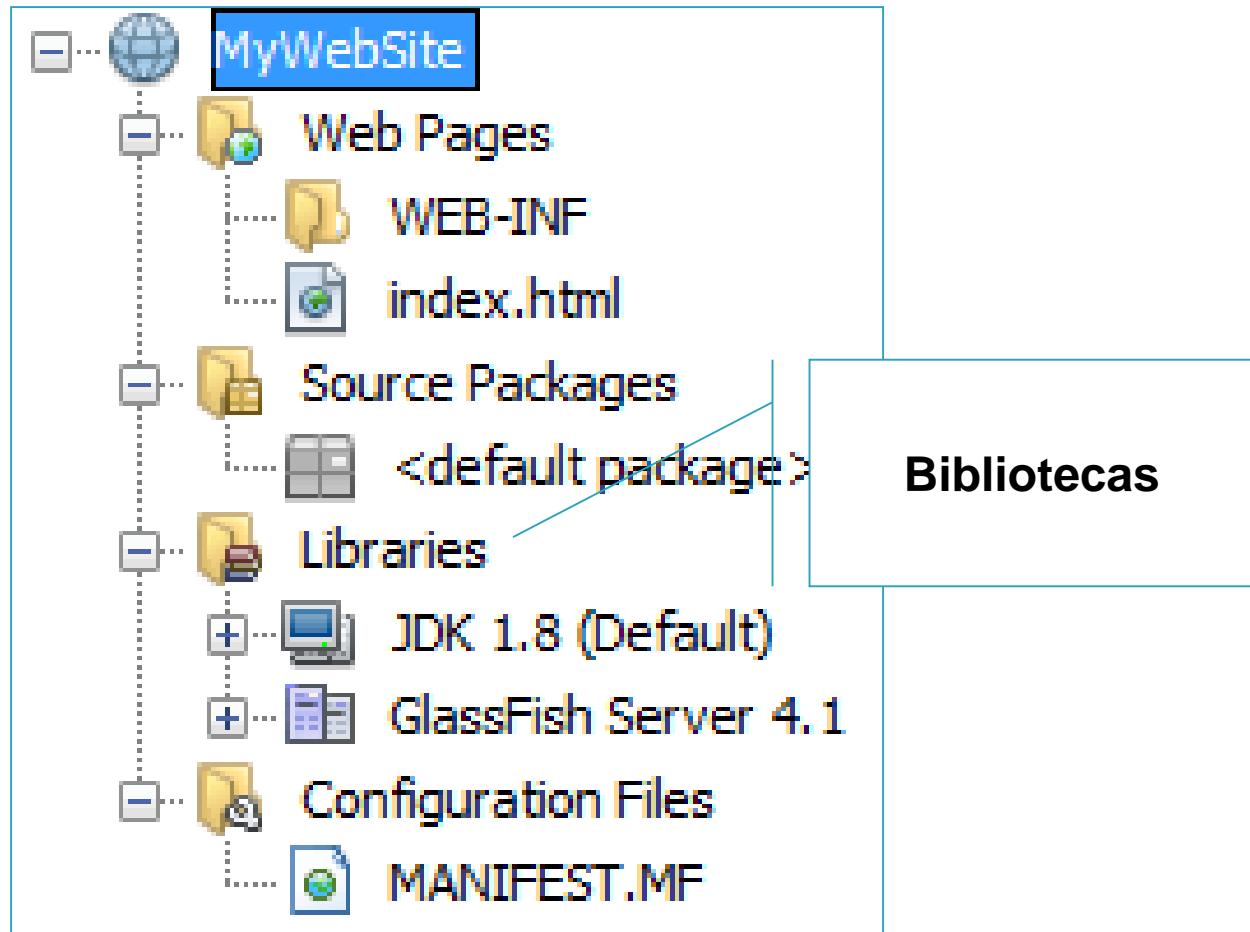
### ► Estrutura do Projeto WEB no Netbeans



**Pacotes e  
Classes Java**

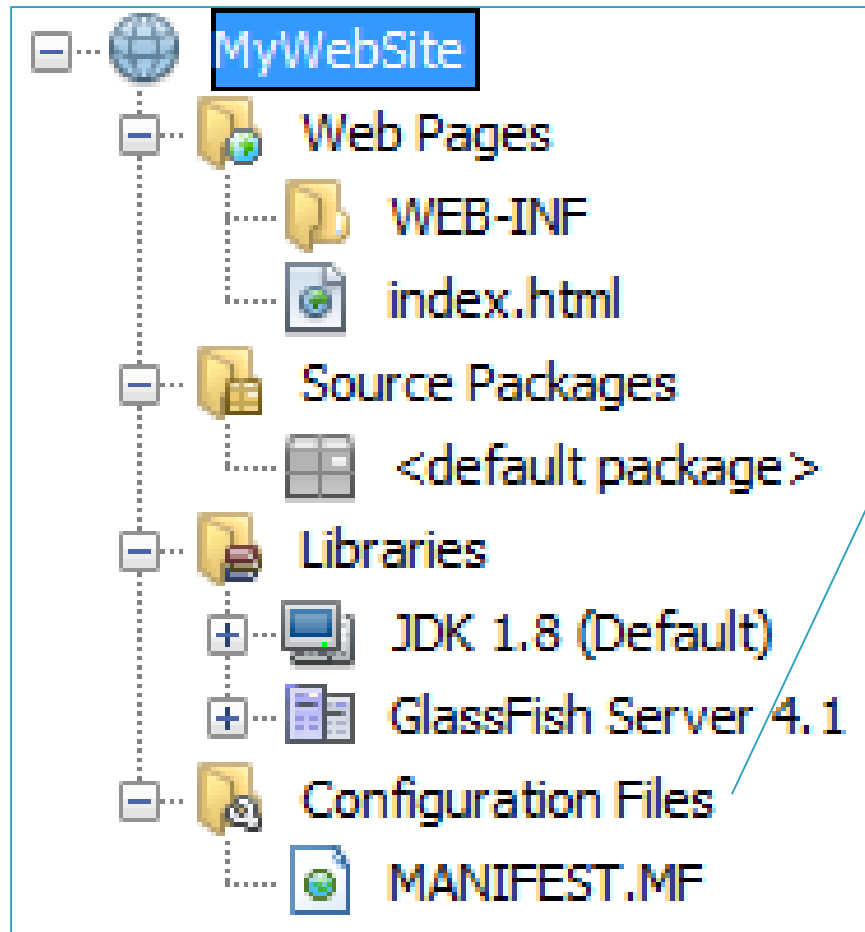
## Java Enterprise Edition

### ► Estrutura do Projeto WEB no Netbeans



## Java Enterprise Edition

### ► Estrutura do Projeto WEB no Netbeans



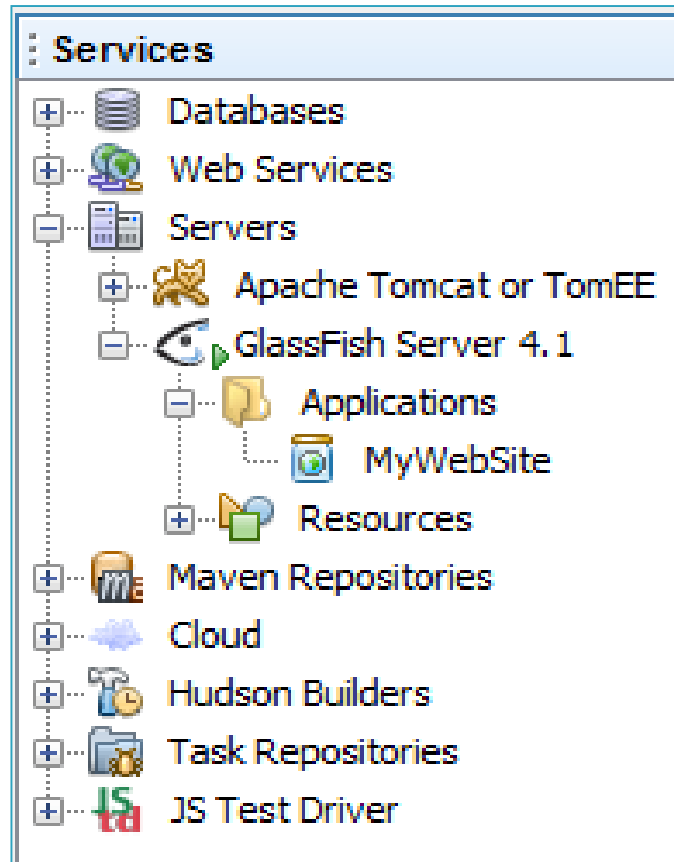
**Arquivos de  
configuração  
da aplicação  
web**

## Java Enterprise Edition

---

### ► Estrutura do Projeto

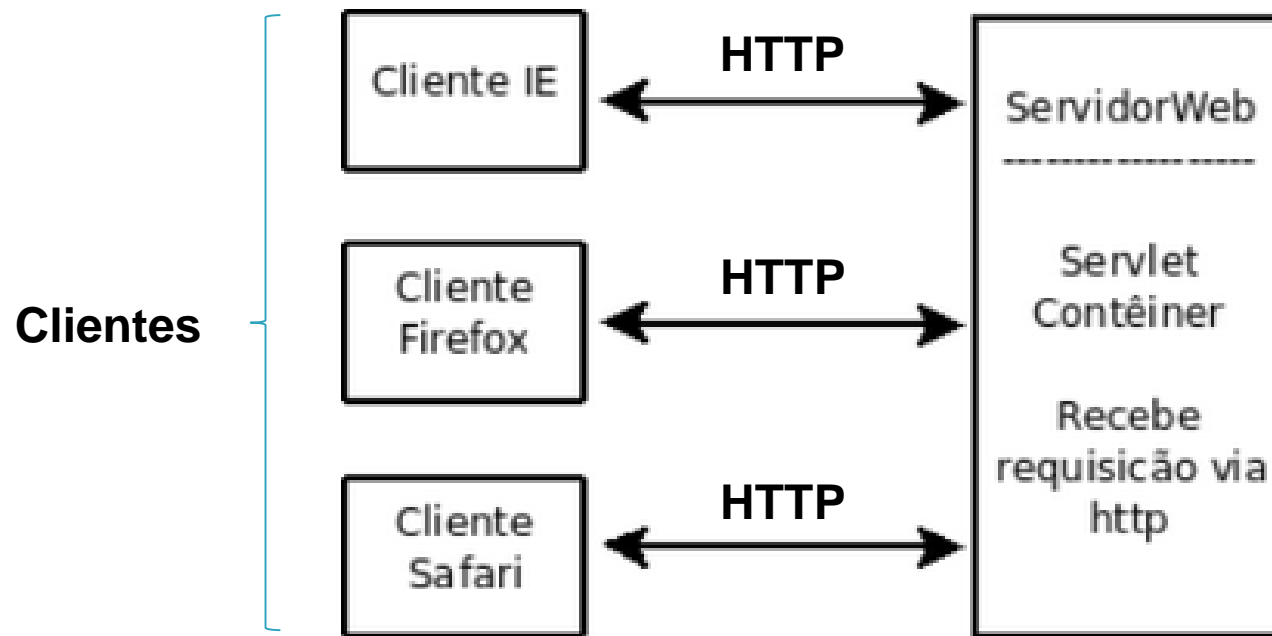
- A aplicação deverá estar implantada em um servidor web



## Java Enterprise Edition

### ► *Servlet* - “Pequeno Servidor”

- Módulo de software executado em um servidor web para atender as requisições de aplicações cliente e prestar-lhes algum tipo de serviço

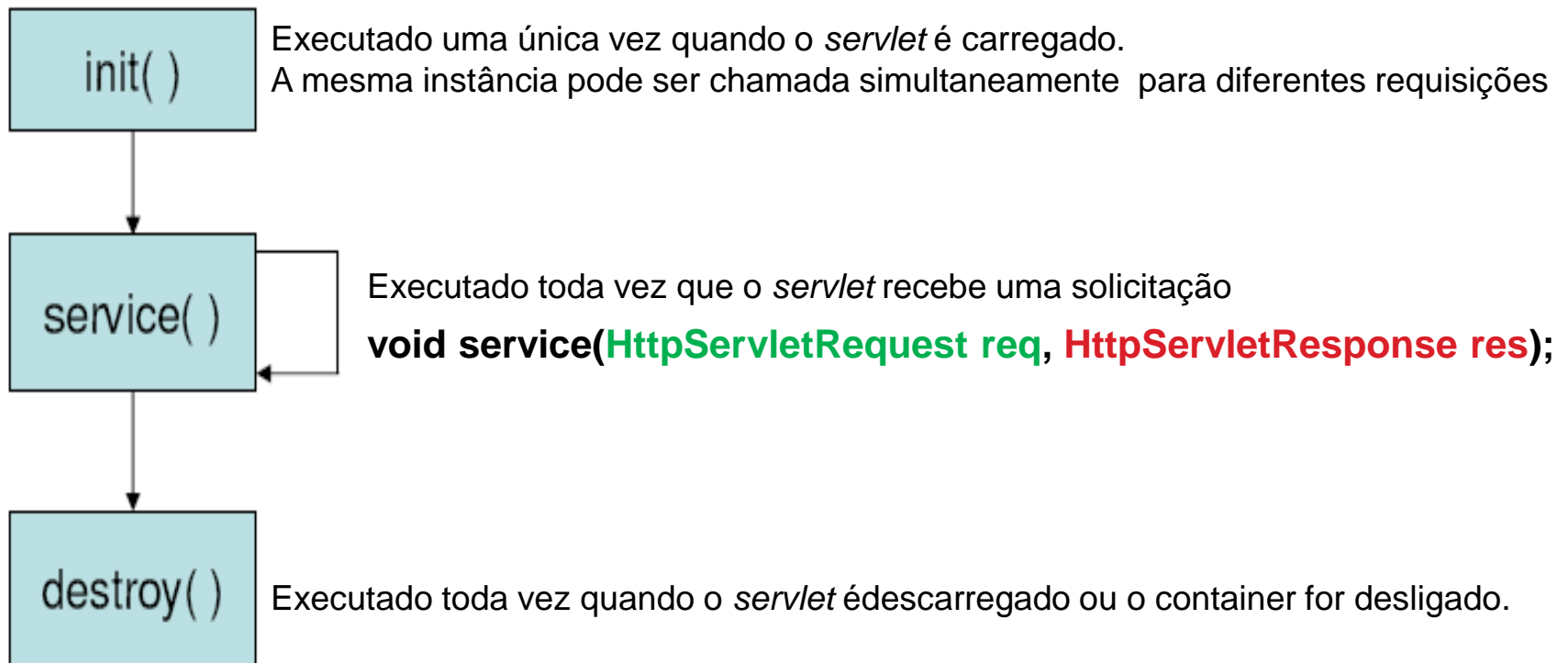


# Java Enterprise Edition

---

## ► Servlet

### ► Funcionamento





## Java Enterprise Edition

---

### ► *Servlet*

#### ► Escrevendo um Servlet

- Importar pacote `javax.servlet.http. HttpServlet` ;
- Criar classe que estenda `HttpServlet`
- Sobrescrever o método `service`

```
protected void service (HttpServletRequest request,  
                        HttpServletResponse response)  
    throws ServletException, IOException {  
    ...  
}
```

# Java Enterprise Edition

- ▶ **Servlet**
  - ▶ *Estrutura Básica*

```
@WebServlet("/minhaServlet")
public class MinhaServlet extends HttpServlet {

    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        log("Iniciando a servlet");
    }

    public void destroy() {
        super.destroy();
        log("Destruindo a servlet");
    }

    protected void service(HttpServletRequest request,
                           HttpServletResponse response)
        throws IOException, ServletException {
        //código do seu método service
    }
}
```

# Java Enterprise Edition

---

## ► Servlet

### ► Nosso Primeiro Servlet

```
public class HelloServlet extends HttpServlet {  
  
    @Override  
    protected void service(  
        HttpServletRequest request,  
        HttpServletResponse response)  
        throws ServletException, IOException {  
  
        PrintWriter out = response.getWriter();  
        out.println("<html>");  
        out.println("<body>Hello Servlet!!</body>");  
        out.println("</html>");  
    }  
}
```

## Java Enterprise Edition

---

### ▶ Servlet

#### ▶ Mapeando uma url a um Servlet

▶ <http://localhost:8080/MyWebSite/hello>

#### ▶ Passos

▶ Criar | Editar arquivo web.xml

```
<servlet>
    <servlet-name>helloServlet</servlet-name>
    <servlet-class>servlets.HelloServlet</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>helloServlet</servlet-name>
    <url-pattern>/hello</url-pattern>
</servlet-mapping>
```

## Java Enterprise Edition



- ▶ *A partir do Servlets 3.0 ...*
  - ▶ Mapeamento por meio de anotações

```
@WebServlet("/hello")  
public class HelloServlet extends HttpServlet {  
  
    @Override  
    protected void service(  
        HttpServletRequest request,  
        HttpServletResponse response)  
        throws ServletException, IOException {  
  
        PrintWriter out = response.getWriter();  
        out.println("<html>");  
        out.println("<body>Hello Servlet!!</body>");  
        out.println("</html>");  
    }  
}
```

- ▶ <http://localhost:8080/MyWebSite/hello>

# Java Enterprise Edition

---

## ► Servlet

### ► Enviando Parâmetros na Requisição

#### ► formAdicionaUsuario.jsp

- ❑ `${pageContext.request.contextPath}`
  - ❑ recupera caminho absoluto da aplicação

```
<form method="post"
    action="${pageContext.request.contextPath}/adicionaUsuario" >
    Email: <input type="text" name="email">
    <br><br>
    Senha: <input type="password" name="senha">
    <br><br>
    <input type="submit" value="Adicionar">
</form>
```

## Java Enterprise Edition

---

### ▶ *Servlet*

- ▶ Recebendo Parâmetros na Requisição
  - ▶ Capturar os dados enviados na requisição

```
String valorDoParametro = request.getParameter("nomeDoParametro");
```

# Java Enterprise Edition

---

## ► Servlet

- Recebendo Parâmetros na Requisição
  - Capturar os dados enviados na requisição

```
@WebServlet("/adicionaUsuario")
public class AdicionaUsuario extends HttpServlet{

    @Override
    protected void service(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {

        //recuperando valores
        String email = request.getParameter("email");
        String senha = request.getParameter("senha");

        //exibindo valores
        PrintWriter out = response.getWriter();
        out.println("<strong>Usuário:</strong> " + email);
        out.println("<strong>Senha:</strong> " + senha);
    }
}
```



## Java Enterprise Edition

---

### ► *Servlet* - Estudo de caso

- Criar um **Servlet** para receber informações de um **Contato** a partir de um formulário HTML, executar o método **adicionaContato()** e exibir uma resposta no navegador do cliente.

Contato
- id : int - nome : String - email : String - endereco : String
+ adicionaContato(contato : Contato) : boolean + getListaContato() : List<Contato>

## Java Enterprise Edition

### ► Servlet - Estudo de caso – 1º Passo

#### ► Criar o Bean: -> POJO -> Plain Old Java Object

- Classe JAVA que expõe atributos, seguindo uma convenção de métodos *getters* e *setters*

```
public class Contato {  
    private int id;  
    private String nome;  
    private String email;  
    private String endereco;  
  
    public Contato() {}  
  
    public Contato(int id, String nome, String email, String endereco)  
  
    public int getId() {...3 linhas }  
    public void setId(int id) {...3 linhas }  
    public String getNome() {...3 linhas }  
    public void setNome(String nome) {...3 linhas }  
    public String getEmail() {...3 linhas }  
    public void setEmail(String email) {...3 linhas }  
    public String getEndereco() {...3 linhas }  
    public void setEndereco(String endereco) {...3 linhas }  
}
```

## Java Enterprise Edition

---

### ► Servlet - Estudo de caso – 2º Passo

#### ► Criar classe DAO: Data Access Object

- Classe Java que implementa métodos relativos às funcionalidades do sistema

```
public class ContatoDAO {  
  
    /** estabelece conexão com BD */  
    public ContatoDAO() {  
        //codigo para carregar conexao com BD  
    }  
  
    public boolean addContato(Contato contato) {  
        //simulando sucesso de inserção no BD  
        return true;  
    }  
}
```

```

@WebServlet("/adicionaContato")
public class AdicionaContatoServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        /***** log *****/
        System.out.println("Criando um novo contato");
        /*****acessando bean *****/
        Contato contato = new Contato(0,
            request.getParameter("nome"),
            request.getParameter("email"),
            request.getParameter("telefone"));
        /*****adicionando ao BD *****/
        ContatoDAO dao = new ContatoDAO();
        boolean inseriu = dao.addContato(contato);
        /***** ok *****/
        PrintWriter out = response.getWriter();
        out.print("<html><body>");
        if(inseriu){
            out.println("Contato " +
                contato.getEmail() +
                " inserido com Sucesso!!");
        }else{
            out.println("Falha ao inserir!!");
        }
        out.print("</body></html>");
    }
}

```

**Servlets - Estudo de caso – 3º Passo**  
 - Criar o Servlet

## Java Enterprise Edition

---

### ► Servlet - Estudo de caso – 4º Passo

- Criar formulário JSP: *formAdicionaContato.jsp*

```
<form method="post"
      action="{pageContext.request.contextPath}/adicionaContato">
  Nome: <input type="text" name="nome">
  <br><br>
  Email: <input type="text" name="email">
  <br><br>
  Telefone: <input type="text" name="telefone">
  <br><br>
  <input type="submit" value="Adicionar">
</form>
```

# Java Enterprise Edition

PROGRAMAÊ!

## ► Servlet – Exercício

Usuario
<ul style="list-style-type: none"><li>- idUsuario : int</li><li>- nome : String</li><li>- login : String</li><li>- senha : String</li></ul>
<ul style="list-style-type: none"><li>+ addUsuario(usuario : Usuario) : boolean</li><li>+ getListUsuario() : List&lt;Usuario&gt;</li><li>+ updateUsusario(usuario : Usuario) : boolean</li><li>+ deleteUsuario(idUsuario : int) : boolean</li></ul>

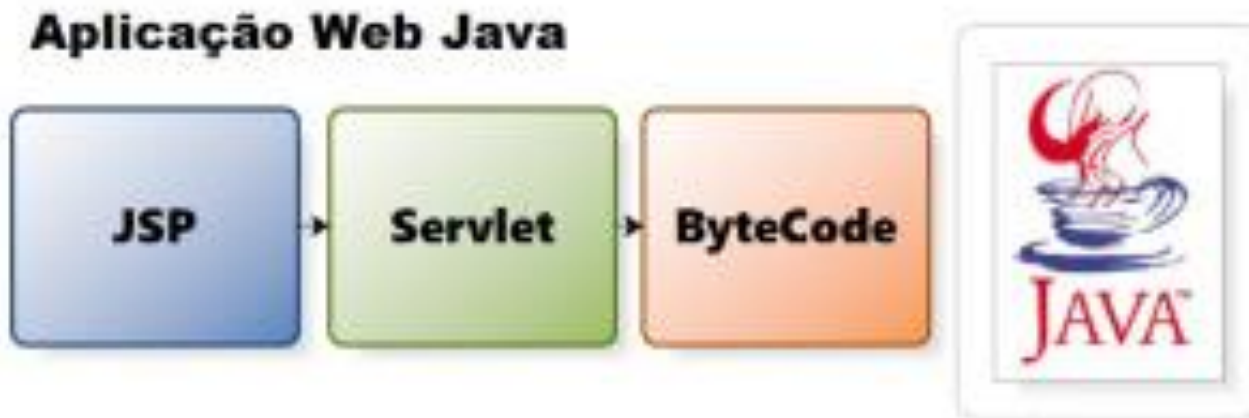
- Crie uma página JSP com um formulário para receber as informações de um usuário e enviar para um servlet.
- Crie o servlet AdicionaUsuario para:
  - receber as informações de um usuário;
  - executar o método addUsuario()
- Crie o servlet ListaUsuario para:
  - executar o método getListUsuario()

## Java Enterprise Edition

---

### ► Java Server Page – JSP

- Página baseada em HTML que processa script JAVA
  - Páginas JSP são transformadas em Servlets.
- Colocar lógica de apresentação dentro de um Servlet não é uma boa prática.



## Java Enterprise Edition

---

### ▶ Java Server Page – JSP

#### ▶ Scriptlet

- ▶ código escrito entre as tags `<% ...código... %>`

```
<%  
String mensagem = "Bem vindo!";  
%>
```

```
<% out.println(nome); %>
```

```
<%-- comentário em jsp --%>
```

```
<%= nome %>
```



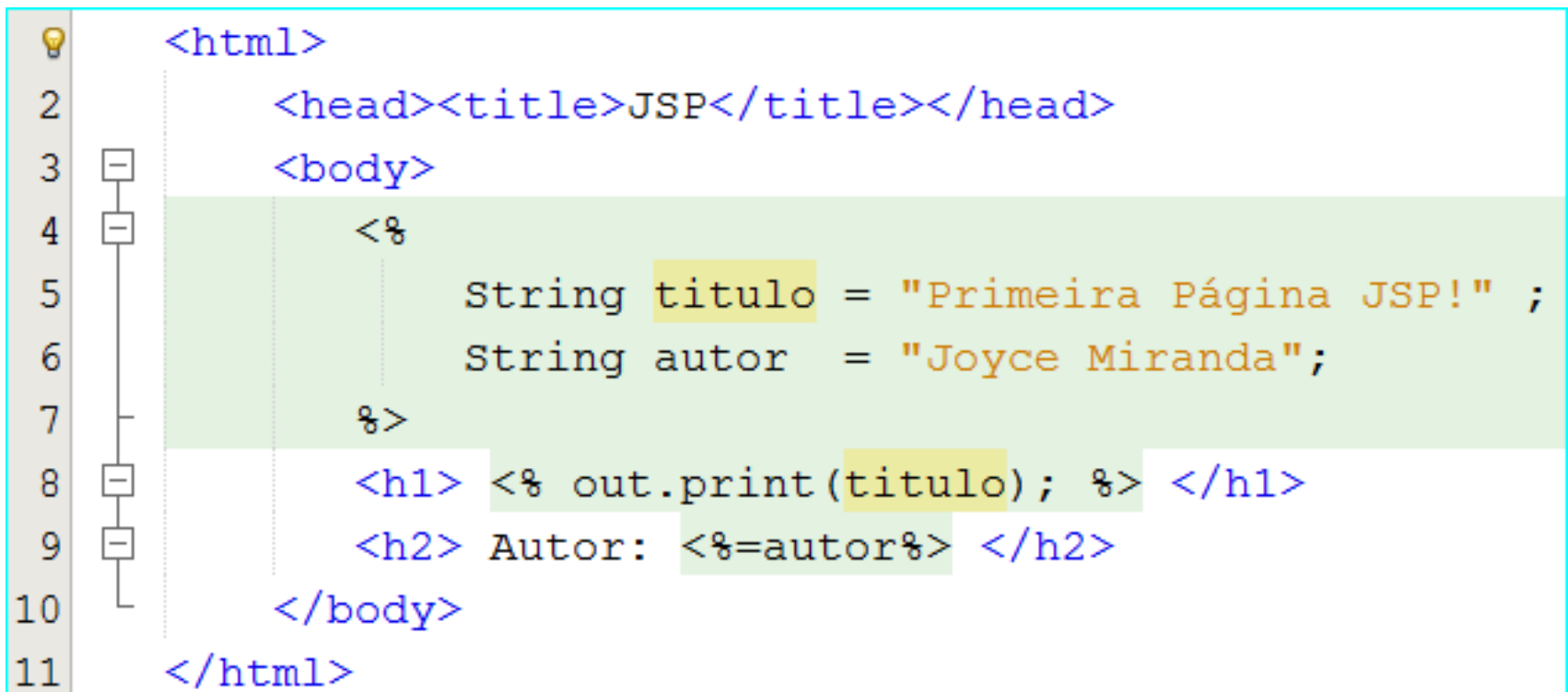
## Java Enterprise Edition

---

### ► Java Server Page – JSP

#### ► Exemplo:

[http://localhost:8080/WEB\\_BASIC\\_JEE/cont.jsp.page/welcome.jsp](http://localhost:8080/WEB_BASIC_JEE/cont.jsp.page/welcome.jsp)



```
1 <html>
2   <head><title>JSP</title></head>
3   <body>
4       <%
5           String titulo = "Primeira Página JSP!" ;
6           String autor  = "Joyce Miranda";
7       %>
8       <h1> <% out.print(titulo); %> </h1>
9       <h2> Autor: <%=autor%> </h2>
10  </body>
11 </html>
```

# Java Enterprise Edition

## ► Java Server Page – JSP

### ► Diretivas de Importação: `<%@page import="" %>`

```
<%@page import="java.util.ArrayList"%>
<%@page import="java.util.List"%>
<html>
  <body>
    <h1>Lista de Alunos</h1>

    <% List<String> listaAlunos = new ArrayList<String>();
      listaAlunos.add("Maria");
      listaAlunos.add("Pedro");
      listaAlunos.add("João");
    %>

    <ol>
      <% for(String aluno: listaAlunos){ %>
        <li><%=aluno%></li>
      <% } %>
    </ol>
  </body>
</html>
```

# Java Enterprise Edition

## ► Expression Language

### ► Remover o uso de scriptlet das páginas JSP

- ❑ **`#{param}`: responsável pelos parâmetros enviados pelo cliente**

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

    Email adicionado: #{param.email}

  </body>
</html>
```

Email:

Senha:

Email adicionado: mds.joyce@gmail.com

## Java Enterprise Edition

---

### ► TagLib

- Biblioteca de *tags* customizadas utilizadas na composição das páginas JSP
- Criando objetos

```
<%  
    Contato contato = new Contato();  
    contato.setNome("Joyce");  
    out.print(contato.getNome());  
%>
```

```
<jsp:useBean id="contato" class="classes.Contato"/>  
<jsp:setProperty name="contato" property="nome" value="Joyce"/>  
${contato.nome}
```

## Java Enterprise Edition

---

- ▶ **JSTL – Java Server Pages Standard Tag Library**
  - ▶ API que encapsula em **tags** funcionalidades de processamento

Pacote	Prefixo	Descrição
<b>JSTL core</b>	c	Tags relacionadas à lógica e controle
<b>JSTL fmt</b>	fmt	Tags para formatação e internacionalização de dados
<b>JSTL sql</b>	sql	Tags para CRUD em um servidor de banco de dados.
<b>JSTL xml</b>	xml	Tags para tratamento de dados XML.
<b>JSTL functions</b>	fn	Tags referentes à funções para o processamento de objetos Strings e coleções.

## Java Enterprise Edition

---

### ► JSTL

#### ► Utilização

- Definir um cabeçalho que vai indicar qual taglib s
- Definir um prefixo para .

#### ► URI

- Não implica em requisição HTTP e sim em um busca entre os arquivos .jar

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

```
<c:import url="cabecalho.jsp" />
```

## Java Enterprise Edition

---

### ▶ JSTL Core

#### ▶ c:url

- ▶ Recupera o caminho (url) absoluto de um recurso

```
<c:url var="pathProduto1" value="/produtos/produto1.jsp"/>  
<a href="${pathProduto1}">Produto 1</a>  
  
${pathProduto1}
```

/MyWebSite/produtos/produto1.jsp

## Java Enterprise Edition

---

### ► JSTL Core

#### ► c:if

```
<c:set var="numero" value="100"/>

<c:if test="${numero > 0}">
    Número Positivo
</c:if>

<c:if test="${numero < 0}">
    Número Negativo
</c:if>
```



## Java Enterprise Edition

---

### ► JSTL Core

#### ► c:choose || c:when || c:otherwise

```
<c:choose>
  <c:when test="{empty nome}">
    Variável nome nula ou vazia!!
  </c:when>
  <c:otherwise>
    {nome}
  </c:otherwise>
</c:choose>
```

# Java Enterprise Edition

## ► JSTL Core

### ► c:forEach – Acessando Métodos

- [http://localhost:8080/WEB\\_BASIC\\_JEE/cont.jsp.page/formListaContatos.jsp](http://localhost:8080/WEB_BASIC_JEE/cont.jsp.page/formListaContatos.jsp)

Contato
- id : long - nome : String - email : String - endereco : String
+ addContato() : boolean + getListaContatos() : List<Contato>

```
<jsp:useBean id="dao" class="mvc.dao.ContatoDAO" />
<table>
  <c:forEach var="contato" items="${dao.listaContatos}">
    <tr>
      <td>${contato.nome}</td>
    </tr>
  </c:forEach>
</table>
```



## ► JSTL - Exercício

Usuario
- idUsuario : int - nome : String - login : String - senha : String
+ addUsuario(usuario : Usuario) : boolean + getListaUsuario() : List<Usuario> + updateUsusario(usuario : Usuario) : boolean + deleteUsuario(idUsuario : int) : boolean

- Crie uma página JSP e exiba a lista de usuários (retornada pelo método `getListaUsuario()`) em uma tabela HTML, destacando com cores diferentes as linhas pares e as linhas ímpares.

## Java Enterprise Edition

---

### ► JSTL Function

- Tags referentes às funções para o processamento de objetos Strings e coleções.

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

```
<c:set var="nome" value="jéssica miranda" />
```

```
${fn:toUpperCase(nome)} <br>
```

```
<c:if test="${fn:contains(nome, 'miranda')}">
```

```
    Você é da minha família!
```

```
</c:if>
```

JÉSSICA MIRANDA  
Você é da minha família!

Functions Syntax	Description
<code>fn:contains()</code>	Returns true if string contains the substring
<code>fn:containsIgnoreCase()</code>	Returns true if string contains the substring,ignores case
<code>fn:endsWith()</code>	Returns true if the string ends with a suffix
<code>fn:escapeXml()</code>	Returns strings as it is that have special meaning in XML,converted to XML character entity code.
<code>fn:indexOf()</code>	Returns index for the first occurence of substring.
<code>fn:join()</code>	Returns strings from array with the given seperator.
<code>fn:length()</code>	Returns the number of elements from a array or a collection,characters from a string.
<code>fn:replace()</code>	Returns a string with the before strings with the after strings.
<code>fn:split()</code>	Returns an array with element got by separating the string with the seperator.
<code>fn:startsWith()</code>	Returns true if the string starts with the prefix.
<code>fn:substring()</code>	Returns a string with begin index till the end index.
<code>fn:substringAfter()</code>	Returns the part of the string after the substring.
<code>fn:substringBefore()</code>	Returns the part of the string before the substring.
<code>fn:toLowerCase()</code>	Returns the allcharacter of a string in lowercase.
<code>fn:toUpperCase()</code>	Returns the allcharacter of a string in uppercase.
<code>fn:trim()</code>	Returns the string removing trailing and leading whitespaces.

## Java Enterprise Edition

---

### ► JSTL Format

- Tags para formatação e internacionalização de dados

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

- *<fmt:formatDate>*

```
<jsp:useBean id="agora" class="java.util.Date" />
```

Data:

```
<fmt:formatDate type="date" value="${agora}" pattern="dd/MM/yyyy" />
```

```
<br>
```

Hora:

```
<fmt:formatDate type="time" value="${agora}" pattern="HH:mm" />
```

```
<br>
```

## Java Enterprise Edition

- ▶ JSTL Format - `<fmt:message>`
- ▶ Internacionalização do seu sistema



## Java Enterprise Edition

---



- ▶ JSTL Format - `<fmt:message>`
  - ▶ Internacionalização do seu sistema
    - ▶ Página: index.jsp



*formAdicionaUsuarioInter.jsp?lang=pt*    *formAdicionaUsuarioInter.jsp?lang=eng*



## Java Enterprise Edition

---

### ► Java Server Page – JSP

#### ► JSTL Format - *<fmt:message>*

##### ► Internacionalização do seu sistema

- 1º: Criar arquivos de *tokens*: *messages\_lang.properties*

```
site.titulo = Formulário de Cadastro  
saudacao = Bem vindo!
```

```
campo.email = Email:  
campo.senha = Senha:
```

```
botao.enviar = Enviar  
botao.cancelar = Cancelar
```

**messages\_pt.properties**

```
site.titulo = Registration Form  
saudacao = Wellcome!
```

```
campo.email = Email:  
campo.senha = Password:
```

```
botao.enviar = Send  
botao.cancelar = Cancel
```

**messages\_eng.properties**

# Java Enterprise Edition

---

- ▶ JSTL Format - *<fmt:message>*
  - ▶ Internacionalização do seu sistema
    - ▶ Página: formAdicionaUsuario.jsp
      - 2º: Colocar tokens nas páginas

```
<c:if test="{not empty param.lang}">
  <fmt:setLocale value="{param.lang}" />
  <fmt:setBundle basename="general.properties.messages" />
</c:if>

<h1><fmt:message key="site.titulo"/></h1>

<form method="post"
  action="{pageContext.request.contextPath}/adicionaUsuario" >
  <fmt:message key="campo.email"/>
  <input type="text" name="email">
  <br><br>
  <fmt:message key="campo.senha"/>
  <input type="password" name="senha">
  <br><br>
  <input type="submit" value="{<fmt:message key="botao.enviar"/>}">
  <input type="submit" value="{<fmt:message key="botao.cancelar"/>}">
</form>
```

## Java Enterprise Edition



- ▶ JSTL Format - `<fmt:message>`
  - ▶ Internacionalização do seu sistema



### Formulário de Cadastro

Email:

Senha:

Enviar

Cancelar

### Registration Form

Email:

Password:

Send

Cancel

## Java Enterprise Edition



### ► Internacionalização do seu sistema

#### ► Versão via Servlet

```
public class MudaLinguaServlet extends HttpServlet {  
    @Override  
    protected void service(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
        String language = request.getParameter("lingua");  
        Locale locale = new Locale(language);  
  
        Config.set(request.getSession(), Config.FMT_LOCALE, locale);  
        Config.set(request.getSession(), Config.FMT_FALLBACK_LOCALE, locale);  
  
        response.sendRedirect("index.jsp");  
    }  
}
```