Desenvolvimento de Aplicações WEB Java Enterprise Edition

Profa. Joyce Miranda

Materiais de Referência: http://www.caelum.com.br/apostila-java-web

Java Enterprise Edition

- **Aplicações WEB**
  - Requisitos funcionais
  - Requisitos não funcionais
    - Desempenho, confiabilidade, segurança, disponibilidade...
    - Serviços de Infra-Estrutura
      - Gerenciamento de conexões HTTP, Gerenciamento de sessões, Persistência de Dados...
  - JEE
    - É um conjunto de especificações que definem como alguns serviços devem ser implementados.
    - Reduz o custo e a complexidade do desenvolvimento.

Java Enterprise Edition

Algumas especificações do JEE

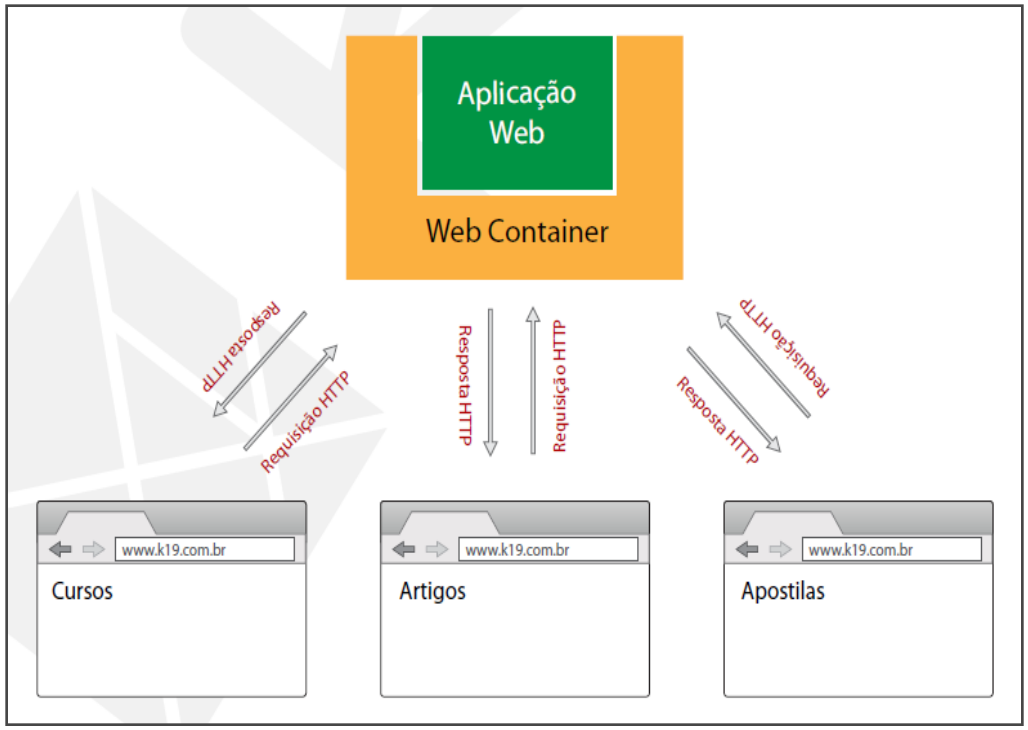
API	Função
JavaServer Pages (JSP) Java Servlets Java Server Faces (JSF)	Funcionalidades para web
Enterprise Javabeans (EJB) Java Persistence API (JPA)	Objetos distribuídos, clusters, acesso remoto.
Java API for XML Web Services (JAX-WS) Java API for XML Binding (JAX-B)	Trabalhar com arquivos XML.
Java Autenthication 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 – Servidor WEB**
  - Para usar o JEE na prática é necessário usar uma implementação das especificações.
  - Servir sua aplicação para auxiliá-la com serviços de infra-estrutura.
  - Exemplos:
    - *Jboss Application Server*
    - *Apache Geronino*
    - *GlassFish*

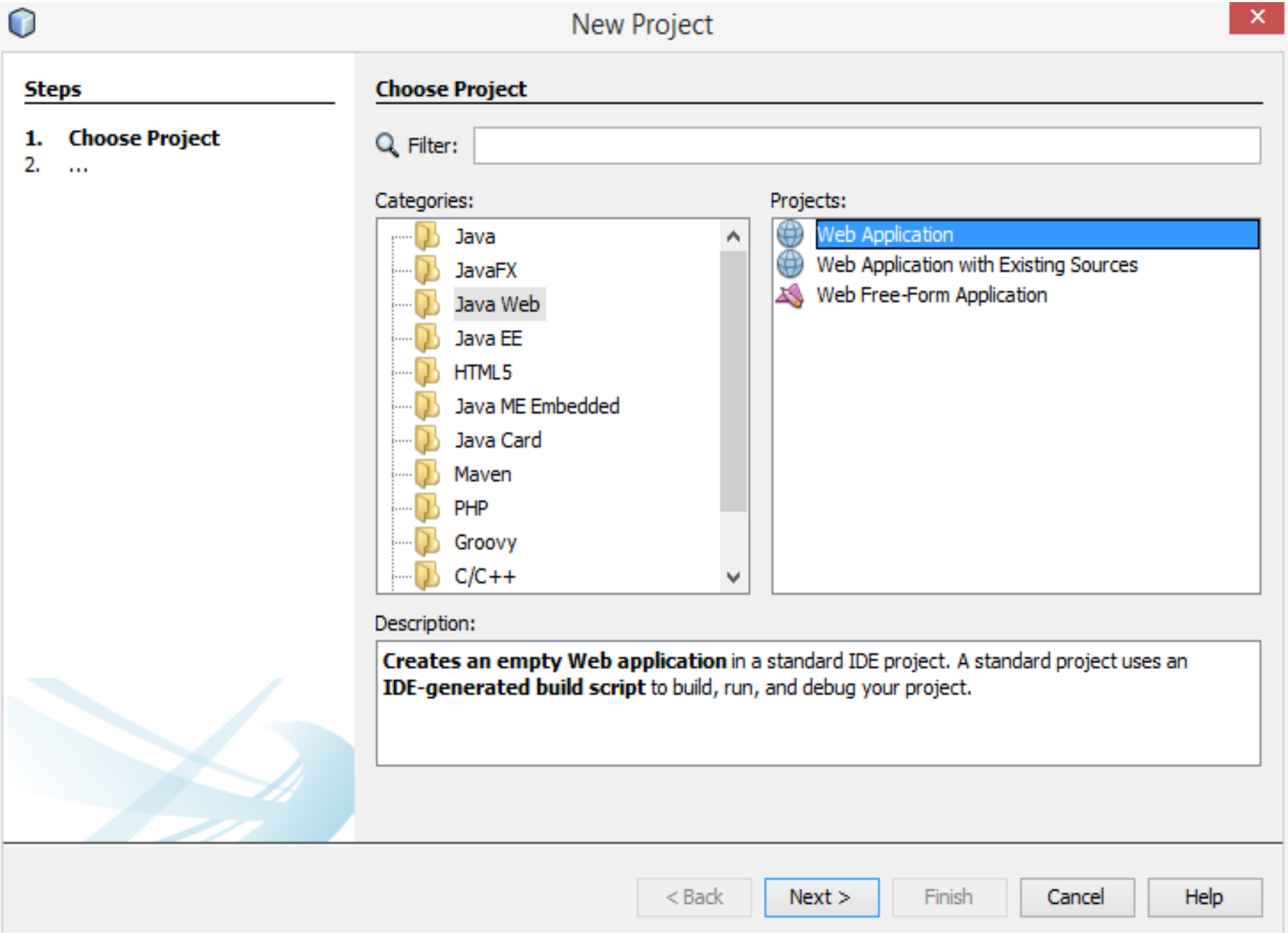
Java Enterprise Edition

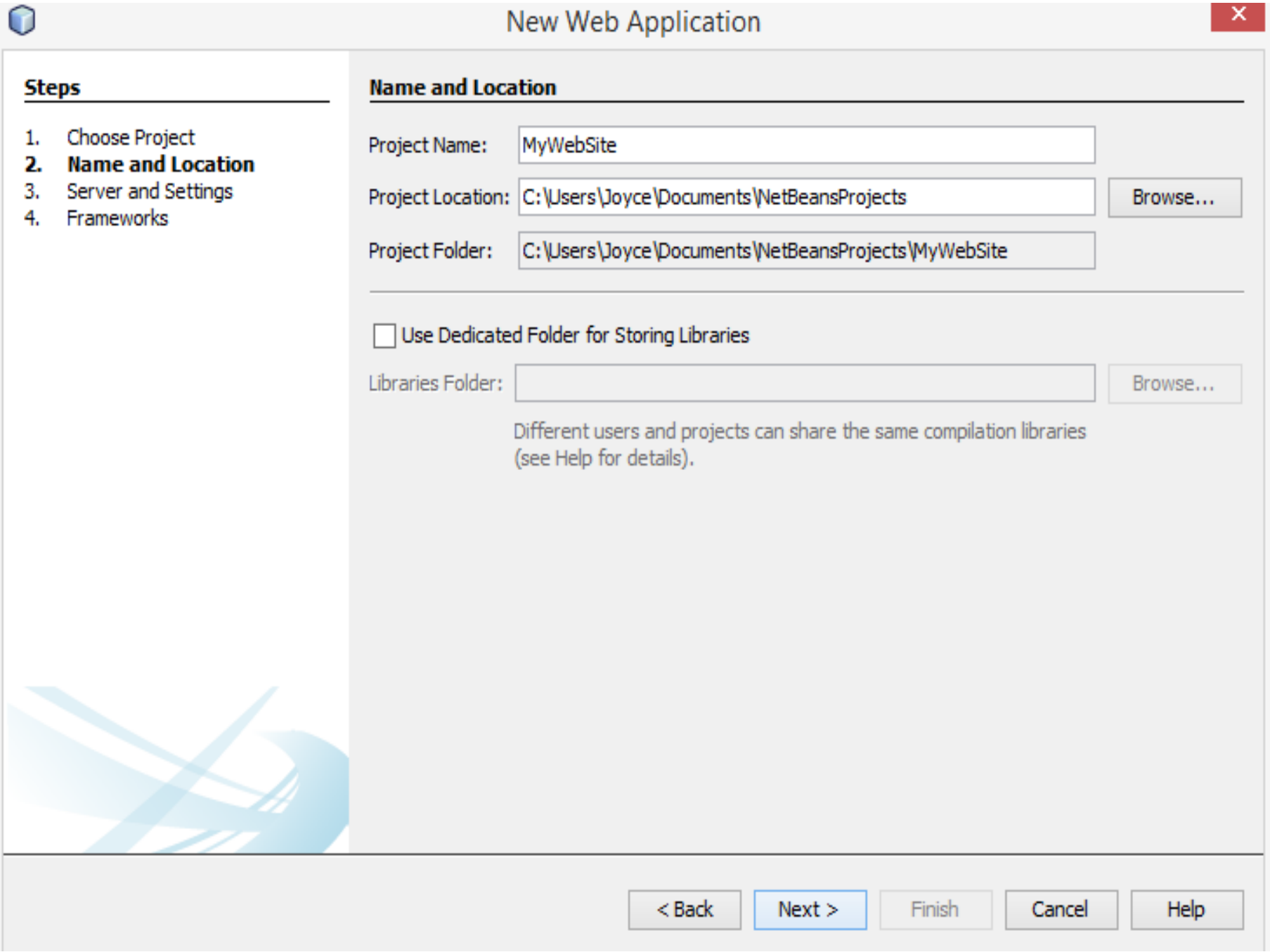
- ***Servlet Container (Web Container)***
  - **Responsabilidades:**
    - Envio e recebimento de mensagens HTTP;
    - Acesso simultâneo;
    - Conteúdo dinâmico.
  - Aplicações Web devem ser implantadas em um *web container* para obter os recursos fundamentais para seu funcionamento

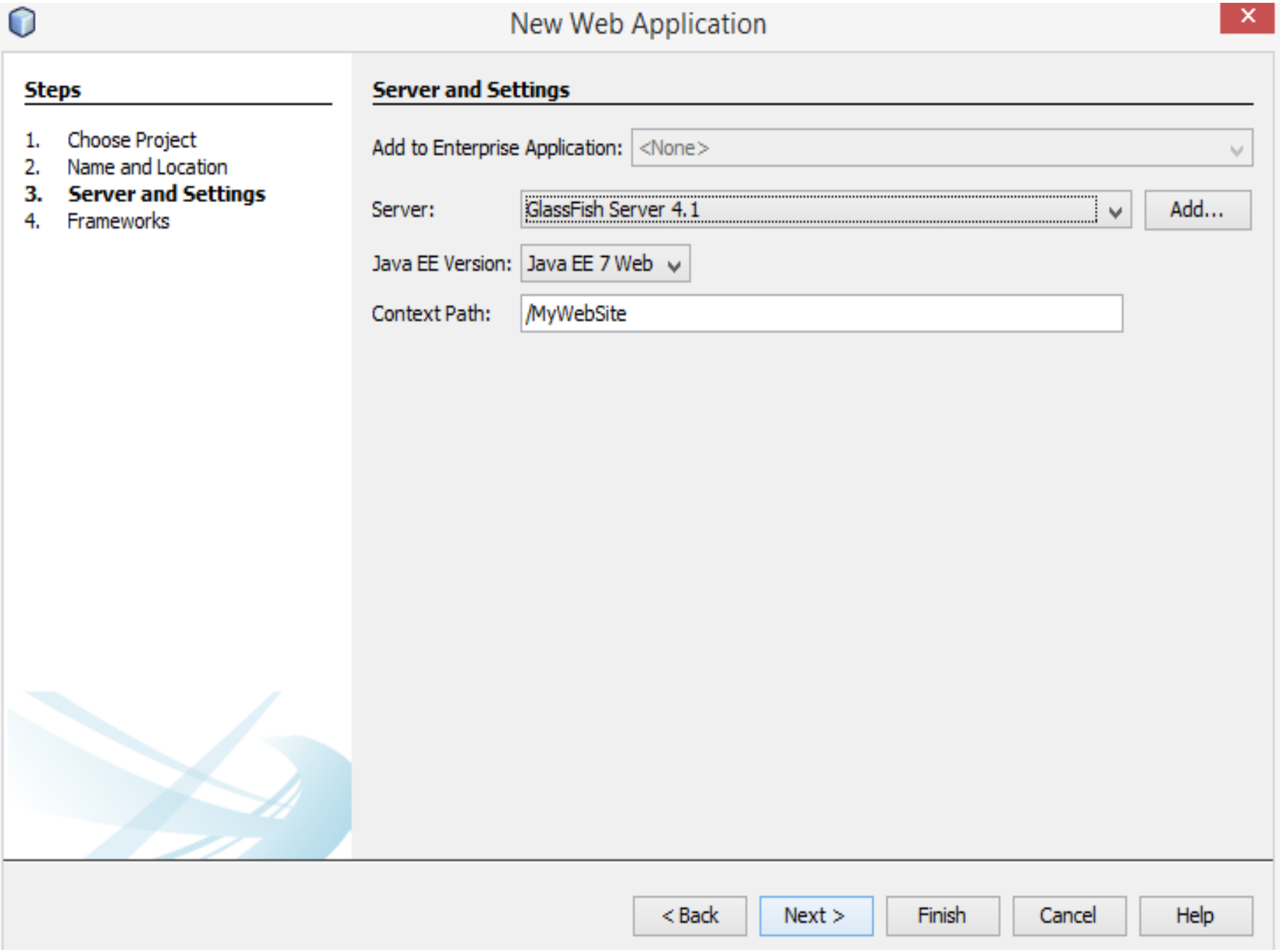


Java Enterprise Edition

- ***Servlet Container***
  - 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

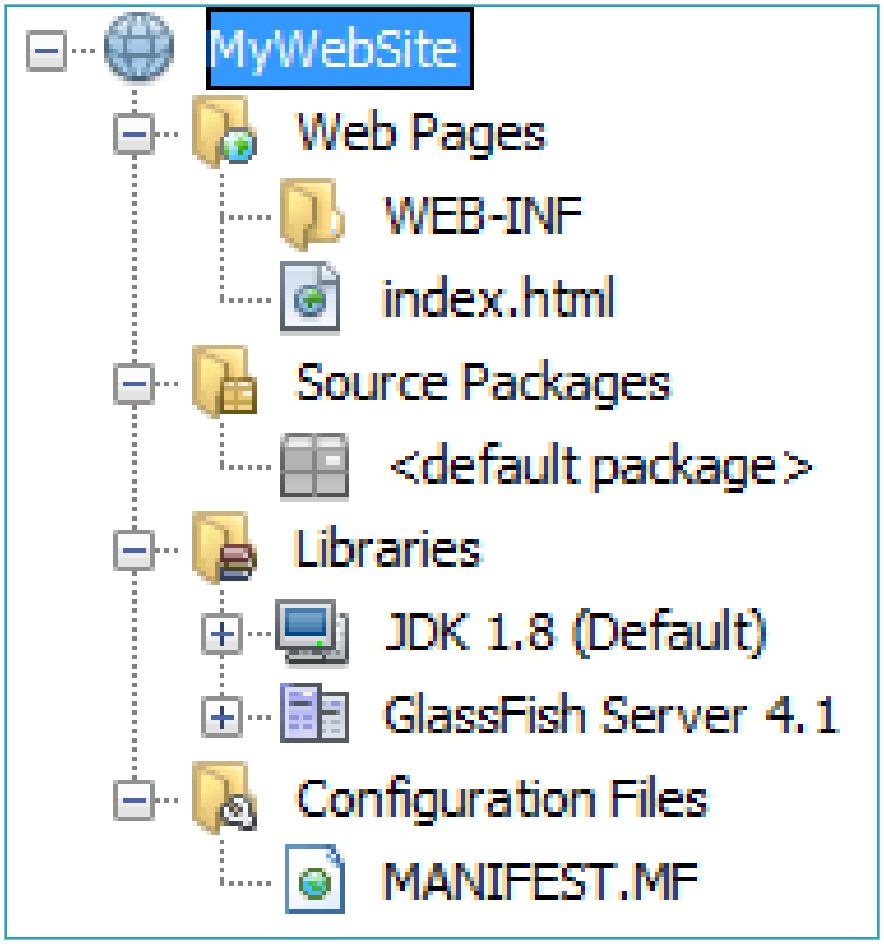

New Project
Steps
1. Choose Project
2. ...
Choose Project
Filter:
Categories:
Java
JavaFX
Java Web
Java EE
HTML5
Java ME Embedded
Java Card
Maven
PHP
Groovy
C/C++
Projects:
Web Application
Web Application with Existing Sources
Web Free-Form Application
Description:
Creates an empty Web application in a standard IDE project. A standard project uses an IDE-generated build script to build, run, and debug your project.
< Back
Next >
Finish
Cancel
Help


New Web Application
Steps
1. Choose Project
2. Name and Location
3. Server and Settings
4. Frameworks
Name and Location
Project Name: MyWebSite
Project Location: C:\Users\Joyce\Documents\NetBeansProjects
Browse...
Project Folder: C:\Users\Joyce\Documents\NetBeansProjects\MyWebSite
Use Dedicated Folder for Storing Libraries
Libraries Folder:
Browse...
Different users and projects can share the same compilation libraries (see Help for details).
< Back
Next >
Finish
Cancel
Help


New Web Application
Steps
1. Choose Project
2. Name and Location
3. Server and Settings
4. Frameworks
Server and Settings
Add to Enterprise Application:
<None>
Server:
GlassFish Server 4.1
Add...
Java EE Version:
Java EE 7 Web
Context Path:
/MyWebSite
< Back
Next >
Finish
Cancel
Help

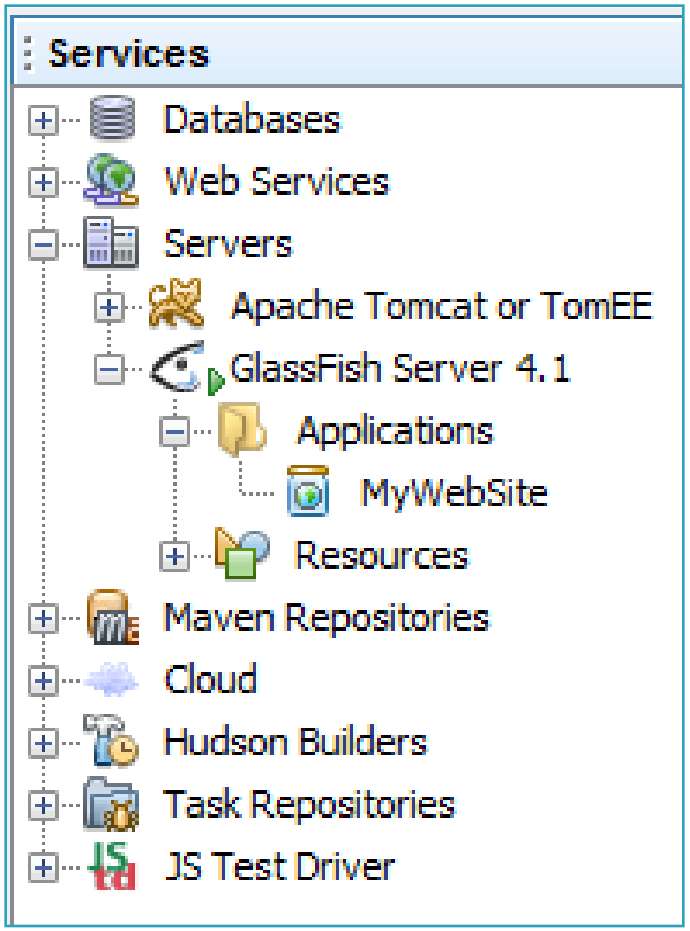
Java Enterprise Edition

- Estrutura do Projeto



Java Enterprise Edition

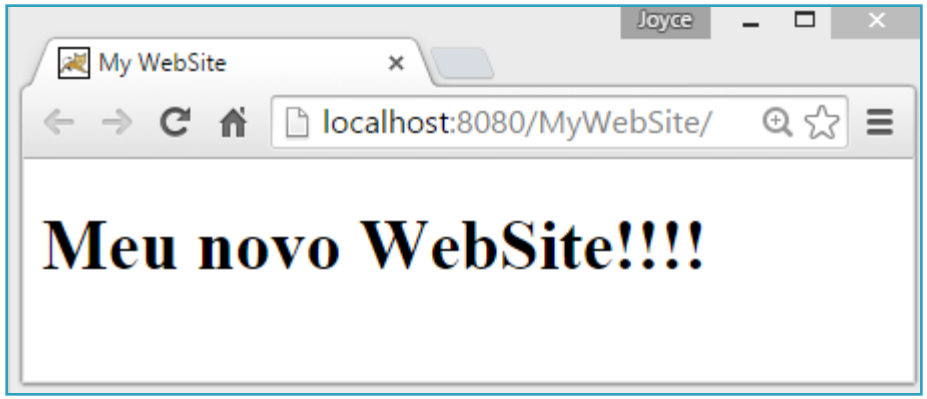
- Estrutura do Projeto



Java Enterprise Edition

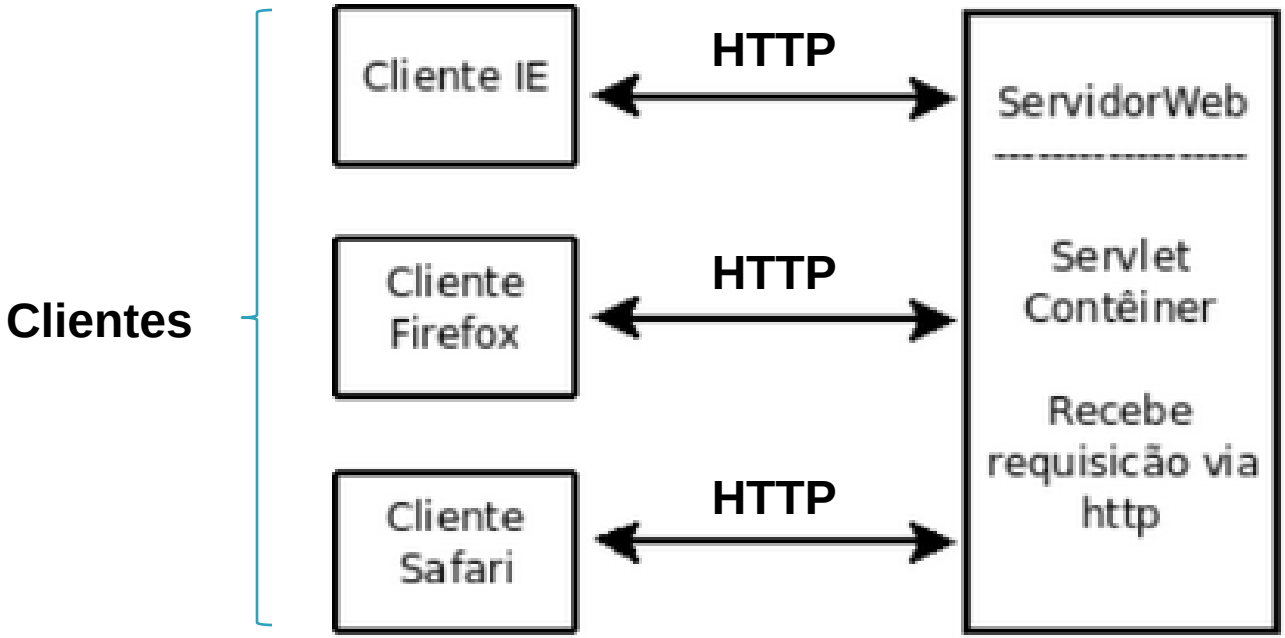
PROGRAMAÊ!

- Primeira Página
  - Crie|Edite index.html
  - Inicie|Reinicie o servidor de aplicação (GlassFish)
  - Acesse
    - http://localhost:8080/MyWebSite/



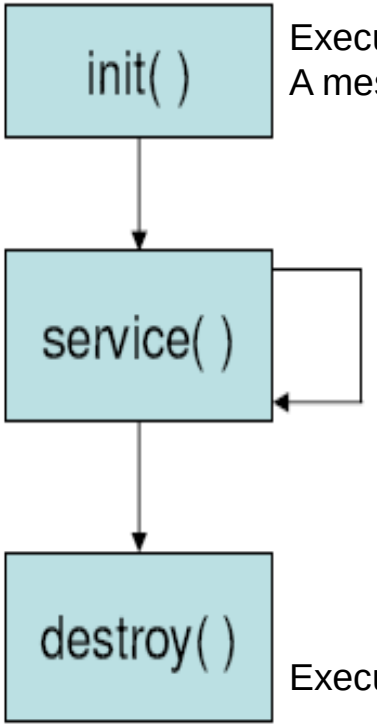
Java Enterprise Edition

- *Servlet - "Pequeno Servidor"*
  - São módulos de software que são executados em um servidor web para atender as requisições de aplicações cliente e prestar-lhes algum tipo de serviço



Java Enterprise Edition

- *Servlet*
  - Funcionamento



Executado uma única vez quando o *servlet* é carregado.
A mesma instância pode ser chamada simultaneamente para diferentes requisições

Executado toda vez que o *servlet* recebe uma solicitação

void service(HttpServletRequest req, HttpServletResponse res);

Executado toda vez quando o *servlet* édescarregado ou o container for desligado.

Java Enterprise Edition

- *Servlet*

```
@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

- Uma única instância de cada *Servlet*

```java
@WebServlet("/contador")
public class Contador extends HttpServlet {
    private int contador = 0; //variavel de instância

    protected void service(HttpServletRequest request,
                           HttpServletResponse response)
                           throws ServletException, IOException {
        contador++; // a cada requisição a mesma variável é incrementada

        // recebe o writer
        PrintWriter out = response.getWriter();

        // escreve o texto
        out.println("<html>");
        out.println("<body>");
        out.println("Contador agora é: " + contador);
        out.println("</body>");
        out.println("</html>");
    }
}
```

Java Enterprise Edition

- *Servlets*
  - HttpServlet - javax.servlet.http.*
  - Escrevendo uma Servlet
    - Criar classe que estenda HttpServlet
    - Sobrescrever o método service

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

Java Enterprise Edition

- *Servlets*
  - Nossa Primeira 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

- *Servlets*
  - Mapeando uma 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

- *Servlets*
  - Mapeando uma Servlet
  - Alternativas

```
<servlet-mapping>
  <servlet-name>primeiraServlet</servlet-name>
  <url-pattern>/oi/*</url-pattern>
</servlet-mapping>
```

```
<servlet-mapping>
    <servlet-name>primeiraServlet</servlet-name>
    <url-pattern>*.php</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>");
    }

}
```

Java Enterprise Edition

- *Servlets*
  - Alternativas

```
@WebServlet(name = "helloServlet", urlPatterns = {"/hello", "/ola"})
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

- *Servlets*
  - Enviando Parâmetros na Requisição
    - formAdicionaUsuario.html

```
7  <html>
8      <body>
9          <form action="./adicionaUsuario" method="post">
10             Email: <input type="text" name="email">
11             <br><br>
12             Senha: <input type="password" name="senha">
13             <br><br>
14             <input type="submit" value="Adicionar">
15         </form>
16     </body>
17 </html>
```

Java Enterprise Edition

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

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

Java Enterprise Edition

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

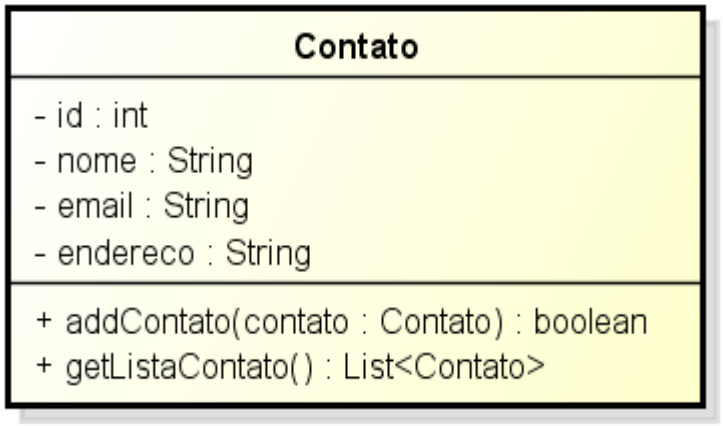
```
@WebServlet("/adicionaUsuario")
public class AdicionaUsuario extends HttpServlet{

    @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);
    }
}
```

- *Servlets – Exercício*
  - *Crie as classes Contato, ContatoDAO*



  - *Crie um formulário HTML para receber as informações de um contato e enviar para um servlet.*
  - *Crie o servlet AdicionaContato para:*
    - *receber as informações de um contato;*
    - *executar o método addContato()*

Java Enterprise Edition

- *Servlets – Exercício*

```
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) {...6 lines }

    public int getId() {...3 lines }
    public void setId(int id) {...3 lines }
    public String getNome() {...3 lines }
    public void setNome(String nome) {...3 lines }
    public String getEmail() {...3 lines }
    public void setEmail(String email) {...3 lines }
    public String getEndereco() {...3 lines }
    public void setEndereco(String endereco) {...3 lines }
}
```

Java Enterprise Edition

- *Servlets – Exercício*
  - *ContatoDAO*

```
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;
    }

}
```

Java Enterprise Edition

- *Java Server Page – JSP*
  - Página baseada em HTML onde pode ser inserido código Java para adicionar comportamento dinâmico.
  - Páginas JSP são transformadas em Servlets.
  - Scriptlet
    - código escrito entre <% *code* %>

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

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

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

```
<%-- comentário em jsp --%>
```

Java Enterprise Edition

- *Java Server Page – JSP*
  - Nossa Primeira JSP: localhost:8080/MyWebSite/wellcome.jsp

```
<html>
    <body>
        <%-- Comentário: My First JSP Page --%>
        <%
            String message = "Seja bem-Vindo(a)!!";
        %>
        <h1><%=message%></h1>
        <% out.println("Desenvolvido por Joyce Miranda"); %>
    </body>
</html>
```

Java Enterprise Edition

- *Java Server Page – JSP*
  - localhost:8080/MyWebSite/listaAlunos.jsp

```
<%@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

- *Java Server Page – JSP*
  - *Expression Language*
    - Remover um pouco do JAVA das páginas JSP

```
<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>
```

Java Enterprise Edition

- *Java Server Page – JSP*
  - *Expression Language*

Arithmetic Operation	Operator
Addition	`+`
Subtraction	`-`
Multiplication	`*`
Division	`/ and div`
Remainder	`% and mod`

```
${4 * 3 / 2 + 1}
```

Java Enterprise Edition

- *Java Server Page – JSP*
  - *Expression Language*

Logical and Relational Operator	Operator
Equals	`== and eq`
Not equals	`!= and ne`
Less Than	`< and lt`
Greater Than	`> and gt`
Greater Than or Equal	`>= and ge`
Less Than or Equal	`<= and le`
and	`&& and and`
or	`|| and or`
not	`! and not`

```
${9 mod 7 eq 0}
```

Java Enterprise Edition

- *Java Server Page – JSP*
  - *TagLibs*
    - Uso de tags para substituir trechos de código
  - 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

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

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

Java Enterprise Edition

- *Java Server Page – JSP*
  - Definindo Cabeçalho
    - Define qual tagLib será usada e define um prefixo para ela.

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

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

Java Enterprise Edition

- *Java Server Page – JSP*
  - *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}

<a href="<c:url value='/produtos/produto1.jsp'/>">Produto 1</a>
```

/MyWebSite/produtos/produto1.jsp

Java Enterprise Edition

- *Java Server Page – JSP*
  - *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

- *Java Server Page – JSP*
  - *JSTL Core*
    - c:choose || c:when || c:otherwise

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

<c:choose>
    <c:when test="${numero > 0}">
        Número Positivo
    </c:when>
    <c:otherwise>
        Número Negativo
    </c:otherwise>
</c:choose>
```

Java Enterprise Edition

- *Java Server Page – JSP*
  - *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

- *Java Server Page – JSP*
  - *JSTL Core*
    - c:forEach

```
<c:forEach begin="0" end="5" var="cont">
    <c:out value="${cont}"/>
</c:forEach>
```

```
<c:forEach begin="0" end="5" var="cont" step="2">
    <c:out value="${cont}"/>
</c:forEach>
```

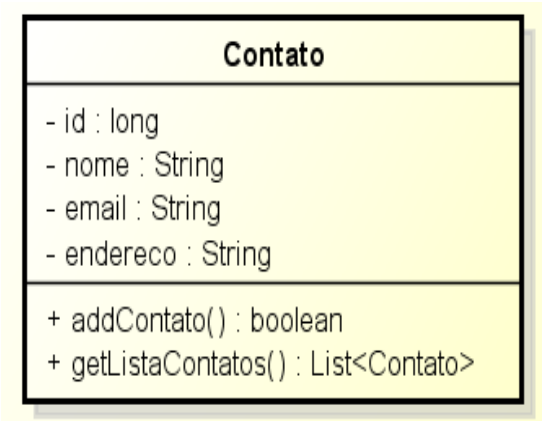
Java Enterprise Edition

- *Java Server Page – JSP*
  - *JSTL Core*
    - c:forEach – Acessando Métodos

```
<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>
```

Java Enterprise Edition

- *Java Server Page – JSP*
  - *JSTL Core*
    - localhost:8080/MyWebSite/contatos.jsp
      - Invoque o método "getListaContatos()" da classe ContatoDAO e exiba seu resultado usando c:forEach




```
public List<Contato> getListaContatos(){
    //simulando consulta ao BD
    List<Contato> lista = new ArrayList<Contato>();
    lista.add(new Contato(0, "João", "joao@email", "AM"));
    lista.add(new Contato(0, "Ruth", "ruth@email", "RJ"));
    lista.add(new Contato(0, "Sara", "sara@email", "SP"));
    return lista;
}
```

      - Exiba os resultados em uma tabela, destacando com cores diferentes as linhas pares e as linhas impares.

Java Enterprise Edition

- *Java Server Page – JSP*
  - *Tratamento de ERROS de servidor*
    - Com JSTL - <c:catch>

```
<c:catch  var="erro" >
    <jsp:useBean id="dao" class="ContatoDAO"/>
    <%-- simulando erro --%>
    <jsp:setProperty name="dao" property="null" />
    <%-- --%>
    <c:forEach items="${dao.listaContatos}">
        nome: ${contato.nome}
    </c:forEach>
</c:catch>
<c:if test="${not empty erro}">
    Ocorreu um erro! <br>
    ${erro}
</c:if>
```

Um erro ocorreu
java.lang.InstantiationException: ContatoDAO

Java Enterprise Edition

- *Java Server Page – JSP*
  - *Tratamento de ERROS de servidor*
    - Com diretiva - <%@page errorPage="" %>


```jsp
<%@page isErrorPage="true"%>
<!DOCTYPE html>
<html>
    <!-- página de erro padrão -->
    Um erro ocorreu <br>
    ${pageContext.errorData.throwable}
</html>
```

```jsp
<%@page errorPage="erro.jsp" %>
<jsp:useBean id="dao" class="ContatoDAO"/>
<%-- simulando erro --%>
<jsp:setProperty name="dao" property="null" />
<%-- --%>
<c:forEach items="${dao.listaContatos}">
    nome: ${contato.nome}
</c:forEach>
```

```
Um erro ocorreu
java.lang.InstantiationException: ContatoDAO
```

Java Enterprise Edition

- *Java Server Page – JSP*
  - *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!

Java Enterprise Edition

- *Java Server Page – JSP*
  - *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

- *Java Server Page – JSP*
  - *JSTL Format - <fmt:message>*
    - Internacionalização do seu sistema

Java Enterprise Edition

PROGRAMAÊ!

- *Java Server Page – JSP*
  - *JSTL Format - <fmt:message>*
    - Internacionalização do seu sistema

formAdicionaUsuarioInter.jsp?lang=pt

formAdicionaUsuarioInter.jsp?lang=eng

Formulário de Cadastro

Email:

Senha:

Enviar Cancelar

Registration Form

Email:

Password:

Send Cancel

Java Enterprise Edition

- *Java Server Page – JSP*
  - *JSTL Format - <fmt:message>*
    - Internacionalização do seu sistema
      - 1º: Criar arquivos 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

- *Java Server Page – JSP*
  - *JSTL Format - <fmt:message>*
    - Internacionalização do seu sistema
      - 2º: Colocar tokens nas páginas

```
<fmt:setLocale value="${param.lang}" />
<fmt:setBundle basename="properties.messages" />

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

<form action="adicionarusuario" method="post">
    <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>
```