

*Projeto Java Web*  
*arquitetura MVC e tecnologia*  
*JavaServer Pages - JSP*

(Teruel, 2009)

*profº Mauricio Conceição Mario*

# Projeto Java Web

## World Wide Web Consortium (W3C)

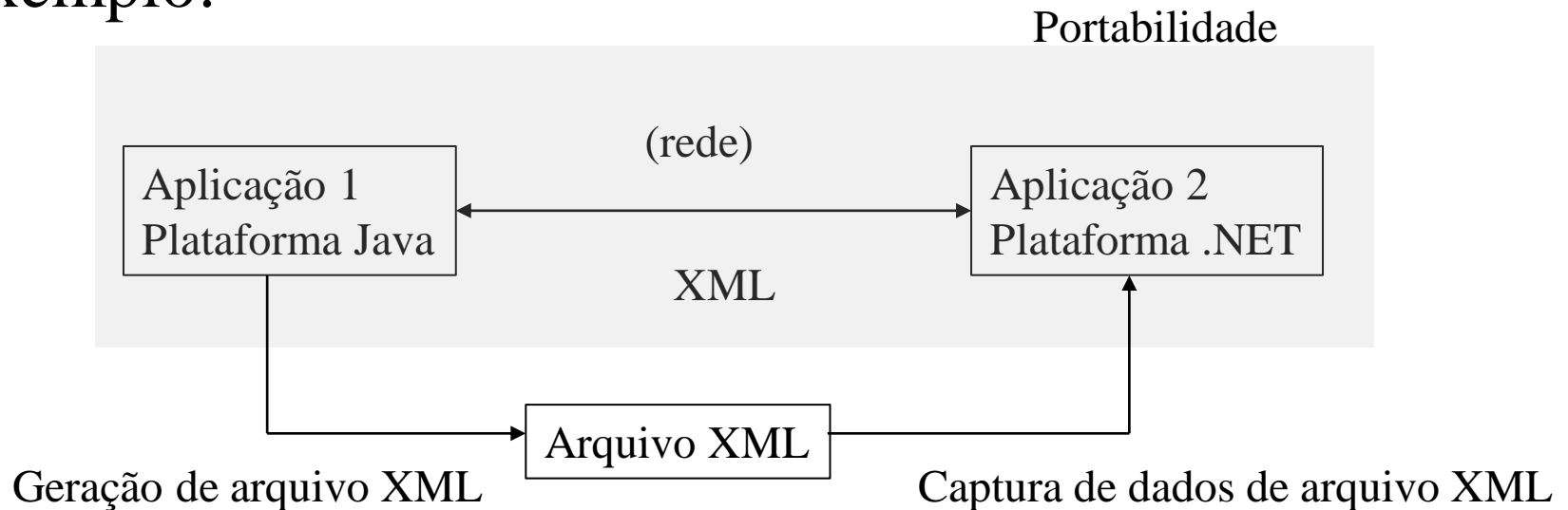
- Organização que inclui as maiores empresas mundiais de tecnologia, que tem por missão criar padrões para disponibilização de conteúdo na web. A criação de padrões é necessária para que os navegadores consigam interpretar um conjunto de elementos (*tags*) que seguem regras bem definidas.
- Para estabelecer padrões com o objetivo de dar suporte ao desenvolvimento de aplicações para a web, a W3C criou uma série de linguagens de marcação (*markup languages*), sendo as principais: *eXtensible Markup Language* – XML, *Hyper Text Markup Language* – HTML e *eXtensible Hyper Text Markup Language* - XHTML.

# Projeto Java Web

## Linguagem de marcação XML

Linguagem padronizada pela W3C que pode ser utilizada na comunicação entre aplicações executadas em diferentes plataformas de hardware ou sistema operacional.

Exemplo:



# Projeto Java Web

## Linguagem de marcação XML

### *Arquivos XML manipulados por CSS*

**Cascading Style Sheets – CSS:** CSS permite a visualização do conteúdo dos elementos XML, definindo a forma como os elementos são exibidos. O uso do CSS permite o mesmo estilo de formatação para exibição de vários documentos XML (é chamada de formatação em cascata).

Exemplo: O arquivo Pessoas5.xml faz uma referência ao arquivo **CSS** (TemplatePessoas5) que irá manipulá-lo através da linha de código: **<?xml-stylesheet type="text/css" href="TemplatePessoas5.css"?>**

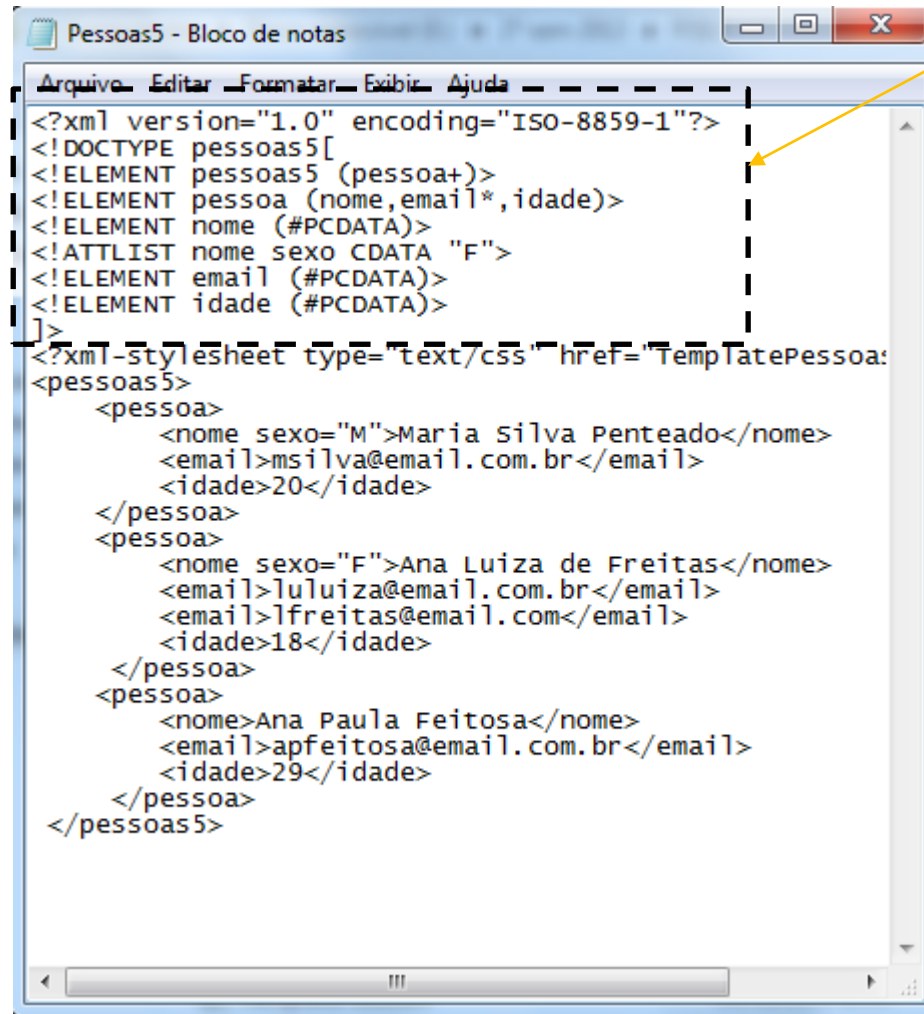
# Projeto Java Web

## Linguagem de marcação XML

*Arquivos XML manipulados por CSS*

**Declaração de Tipo de Documento – DTD:** valida um arquivo XML. A DTD contém um conjunto de instruções que descrevem os elementos usados no documento XML.

código Pessoas5.xml que será manipulado por arquivo CSS.



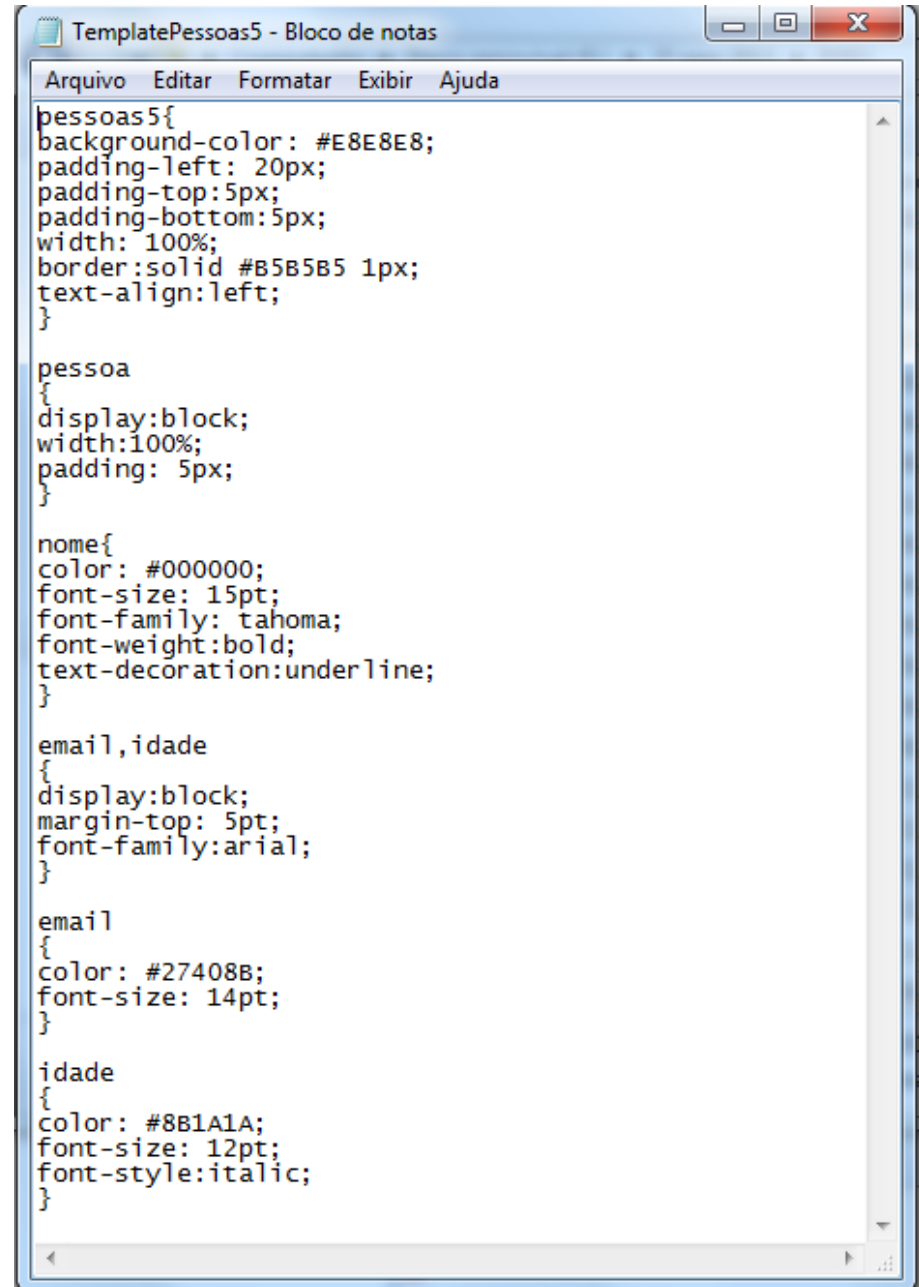
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE pessoas5[
<!ELEMENT pessoas5 (pessoa+)>
<!ELEMENT pessoa (nome,email*,idade)>
<!ELEMENT nome (#PCDATA)>
<!ATTLIST nome sexo CDATA "F">
<!ELEMENT email (#PCDATA)>
<!ELEMENT idade (#PCDATA)>
]>
<?xml-stylesheet type="text/css" href="TemplatePessoa:
<pessoas5>
  <pessoa>
    <nome sexo="M">Maria Silva Penteado</nome>
    <email>msilva@email.com.br</email>
    <idade>20</idade>
  </pessoa>
  <pessoa>
    <nome sexo="F">Ana Luiza de Freitas</nome>
    <email>luluiza@email.com.br</email>
    <email>lfreitas@email.com</email>
    <idade>18</idade>
  </pessoa>
  <pessoa>
    <nome>Ana Paula Feitosa</nome>
    <email>apfeitosa@email.com.br</email>
    <idade>29</idade>
  </pessoa>
</pessoas5>
```

# Projeto Java Web

## Linguagem de marcação XML

*Arquivos XML manipulados por CSS*

Código do arquivo  
**TemplatePessoas5.css**,  
responsável pela  
formatação de  
**Pessoas5.xml**:



```
TemplatePessoas5 - Bloco de notas
Arquivo  Editar  Formatar  Exibir  Ajuda

pessoas5{
background-color: #E8E8E8;
padding-left: 20px;
padding-top:5px;
padding-bottom:5px;
width: 100%;
border:solid #B5B5B5 1px;
text-align:left;
}

pessoa
{
display:block;
width:100%;
padding: 5px;
}

nome{
color: #000000;
font-size: 15pt;
font-family: tahoma;
font-weight:bold;
text-decoration:underline;
}

email,idade
{
display:block;
margin-top: 5pt;
font-family:arial;
}

email
{
color: #27408B;
font-size: 14pt;
}

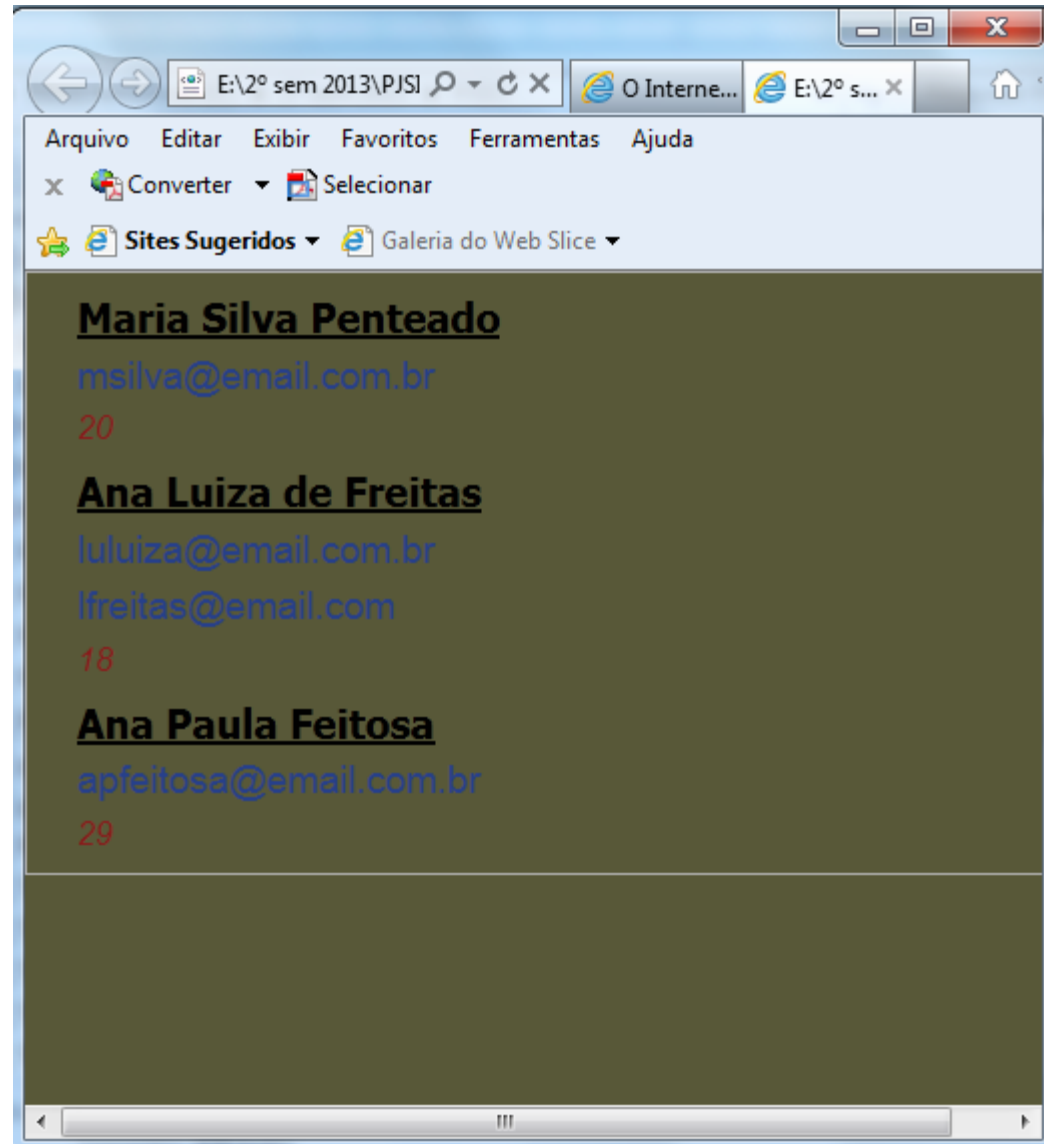
idade
{
color: #8B1A1A;
font-size: 12pt;
font-style:italic;
}
```

# Projeto Java Web

## Linguagem de marcação XML

### *Arquivos XML manipulados por CSS*

Para formatação do arquivo **Pessoas5.xml** com o arquivo **TemplatePessoas5.css** é necessário editar e salvar os arquivos com as respectivas extensões em uma mesma pasta. A seguir deve-se “clique” duas vezes com o mouse (esquerdo) sobre o arquivo **Pessoas5.xml**. O resultado da formatação do arquivo **Pessoas5.xml** é mostrado ao lado.



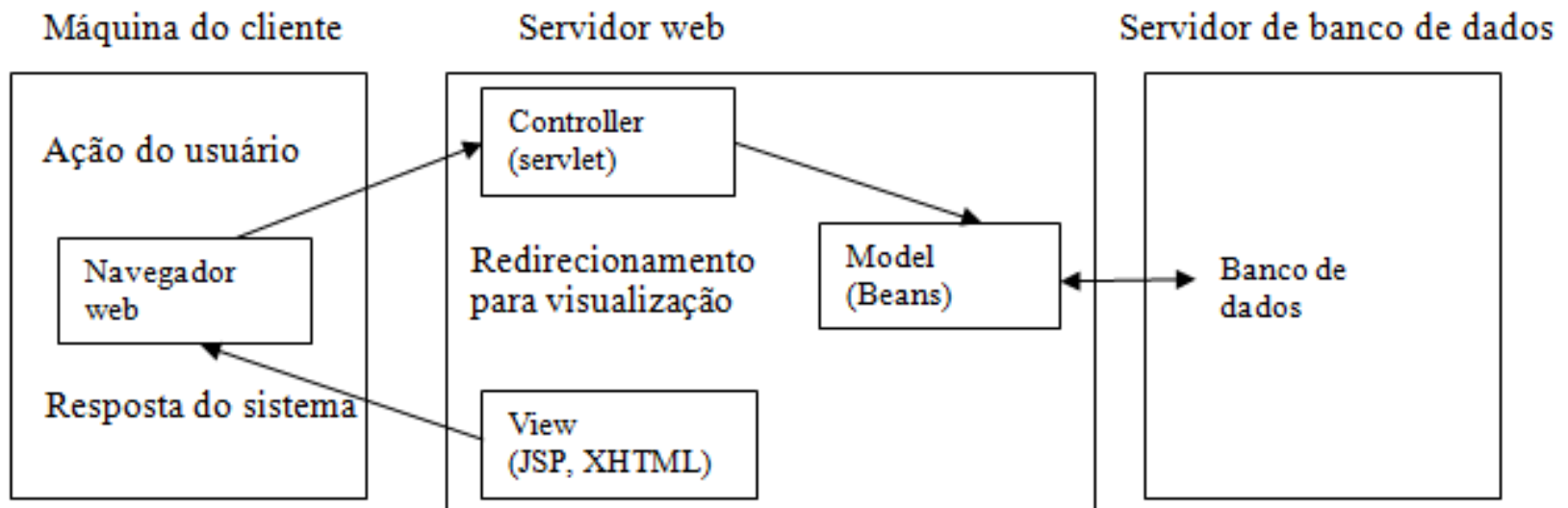
# Projeto Java Web

## Modelo de projeto de aplicações web – *Model-View-Control* –MVC

(Teruel, 2009)

camada	tecnologia	descrição
<i>View</i>	HTML, XHTML, CSS, <i>JavaScript</i> , JSP, JSTL, XML	Para apresentação dos dados a principal tecnologia é a JSP que gera conteúdo no formato HTML para a exibição dos dados no navegador.
<i>Controller</i>	<i>Servlets</i>	Classes Java no padrão Java EE que recebem requisições do cliente, processam e enviam para a camada <i>Model</i> . Essas classes retomam o conteúdo recebido da camada <i>Model</i> para o cliente.
<i>Model</i>	<i>beans</i>	Classes Java no padrão Java SE que manipulam atributos por meio de métodos <i>setters</i> e <i>getters</i> .

Tabela 1: tecnologias utilizadas nas camadas do modelo MVC [Teruel, 2009].

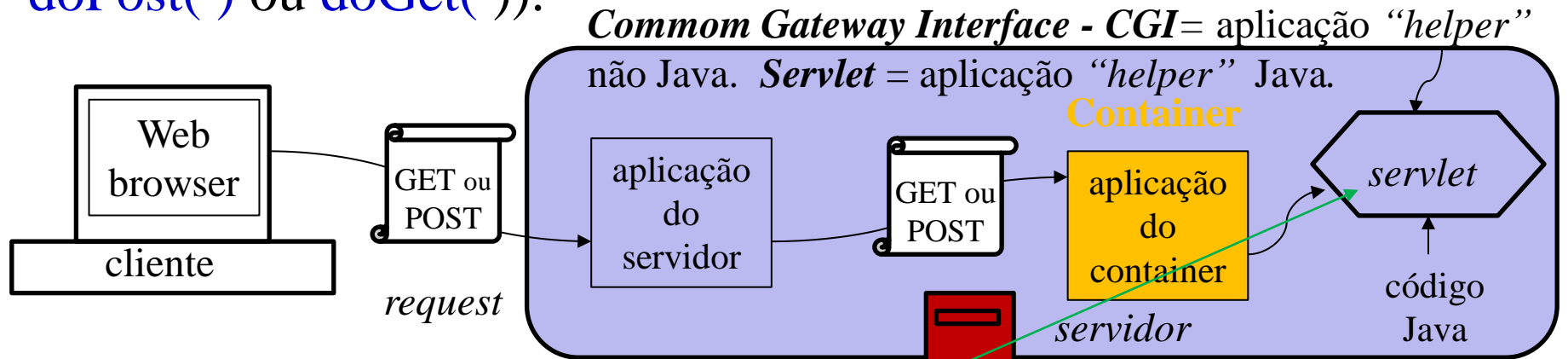


Representação do modelo MVC



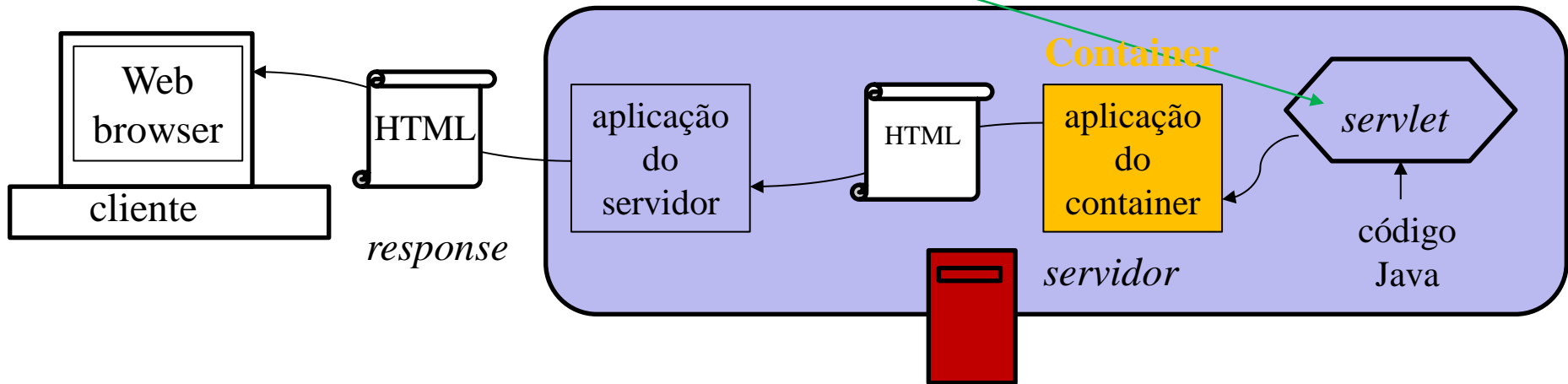
## Container – exemplos: Tomcat e Glassfish

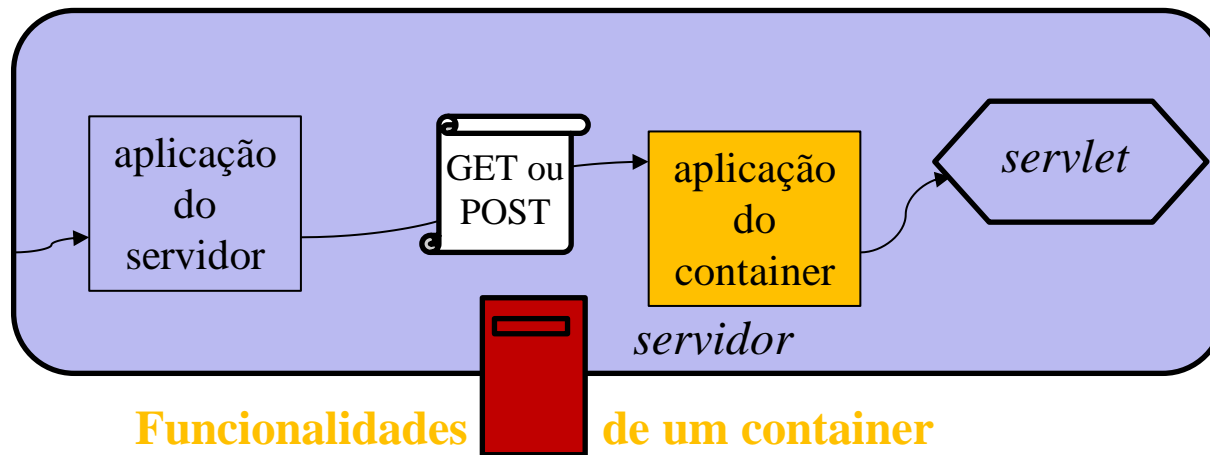
aplicação java, dentro de um servidor, que entrega a um *servlet* a *request* e a *response* HTTP. Chama os métodos do *servlet* (como `doPost()` ou `doGet()`).



*Servlet: é uma aplicação “helper” do tipo Java.*

*Não possui um método **main()**.*





**Suporte para comunicações:** proporciona a comunicação com o servidor através do conhecimento do protocolo do mesmo, proporcionando o acesso ao *servlet*. Evita desse modo a necessidade de construção de um *ServerSocket* e de monitoramento de uma porta.

**Gerenciamento do ciclo de vida:** controla o nascimento e a morte dos seus *servlets*. Carrega as classes, instancia e inicializa os *servlets*, chama os métodos do *servlet* e torna as instâncias do *servlet* aptas a coletar o lixo.

**Deployment Descriptor (DD):** O documento XML chamado *Deployment Descriptor* mapeia as *URLs* aos *servlets*. O *Deployment Descriptor* também informa como executar seus *servlets* e JSPs.

**Segurança:** O *Deployment Descriptor* é utilizado para configurar a segurança envolvendo um *servlet*.

**Suporte ao JSP:** traduz o código JSP para Java.

# Arquitetura MVC

## Exemplo de projeto de utilização de *Servlet* com acesso a banco de dados

- Na arquitetura MVC as *servlets* são usadas na camada *Controller*, na função de gerenciamento do fluxo de informações da aplicação. Todos os componentes da camada *View* (arquivos JSP, HTML, XHTML, CSS, *JavaScript*, etc) centralizam suas requisições nas *Servlets*, fornecendo dados que serão processados e que podem gerar operações no banco de dados, retornando conteúdo ao usuário no formato HTML.
- Criar páginas JSP implica necessariamente na geração de *servlets* no momento em que a aplicação é compilada e executada no servidor. Por exemplo, no *NetBeans*, quando uma página JSP é criada, durante a compilação é gerada uma classe *servlet* com a extensão **\_jsp.java**.
- No projeto *JavaWeb* construído, ao ser compilado o arquivo “Controle.jsp” é gerada a *servlet* “Controle\_jsp.java”, que passa a exercer a função de receber as requisições dos componentes da camada *view*.

(Teruel, 2009)

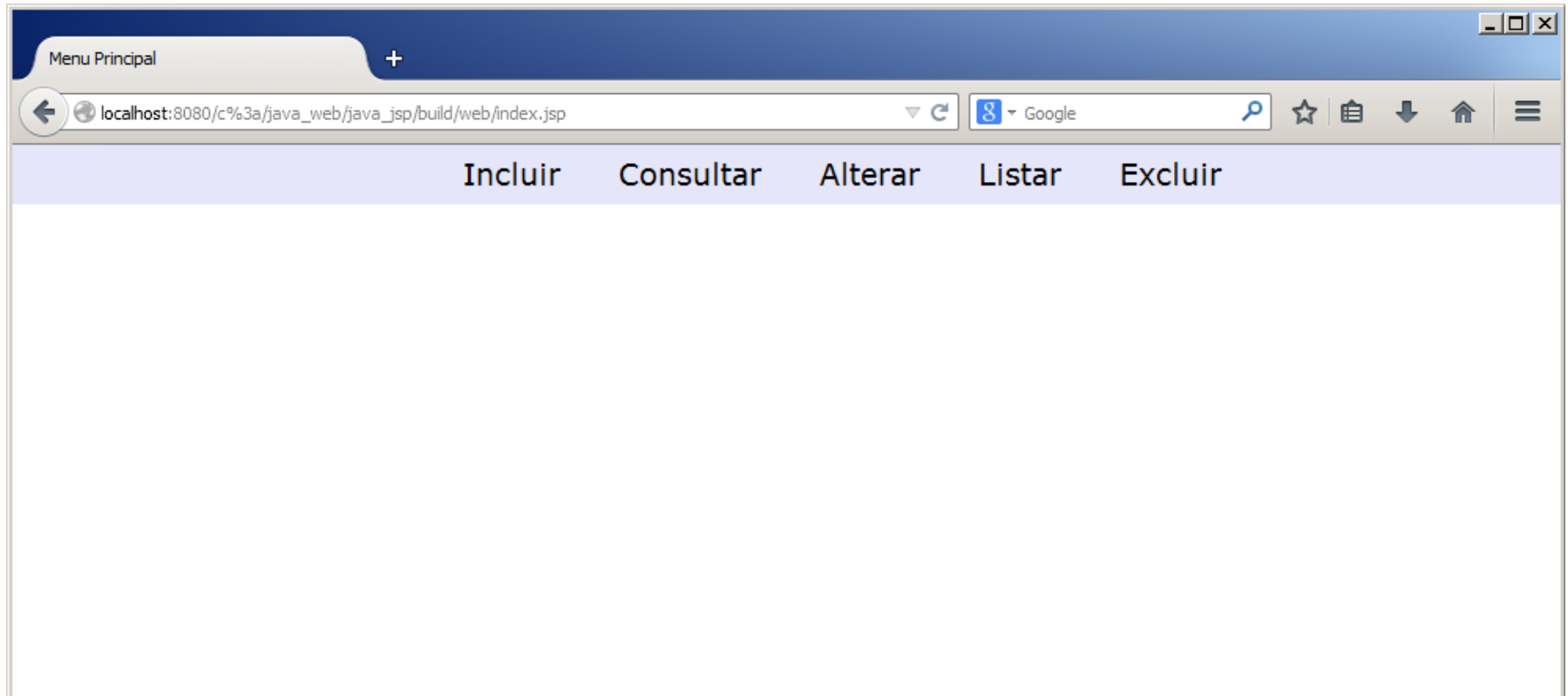
# Projeto Java Web

## Exemplo de projeto de aplicação JSP com acesso a banco de dados

- A tecnologia JSP é própria para o desenvolvimento de aplicações web em que as informações possam ser atualizadas através de acessos a base de dados. As páginas web construídas com tecnologia JSP podem conter elementos HTML, XHTML, XML, CSS, instruções *JavaScript* e Java. Conter instruções Java possibilita o acesso a banco de dados, manipulação de tipos de dados primitivos e de referência, laços de repetição, estruturas de seleção, tratamento de exceções, além da criação de métodos e objetos de classes no padrão Java SE. O projeto demonstrado como exemplo é uma aplicação web para cadastrar, alterar, excluir e pesquisar clientes utilizando JSP com banco de dados MySQL. A interface com o usuário da aplicação é mostrada na figura:

(Teruel, 2009)

# Projeto Java Web



Interface com usuário de aplicação web criada com tecnologia JSP a partir de exemplo original de (Teruel, 2009).

# Projeto Java Web

## Arquitetura da aplicação

A arquitetura escolhida para a aplicação é do tipo próximo ao MVC, com característica de conter código Java nos arquivos JSP de apresentação de conteúdo ao usuário [Teruel, 2009]. A distribuição dos componentes da aplicação através da arquitetura MVC é feita na figura 1.

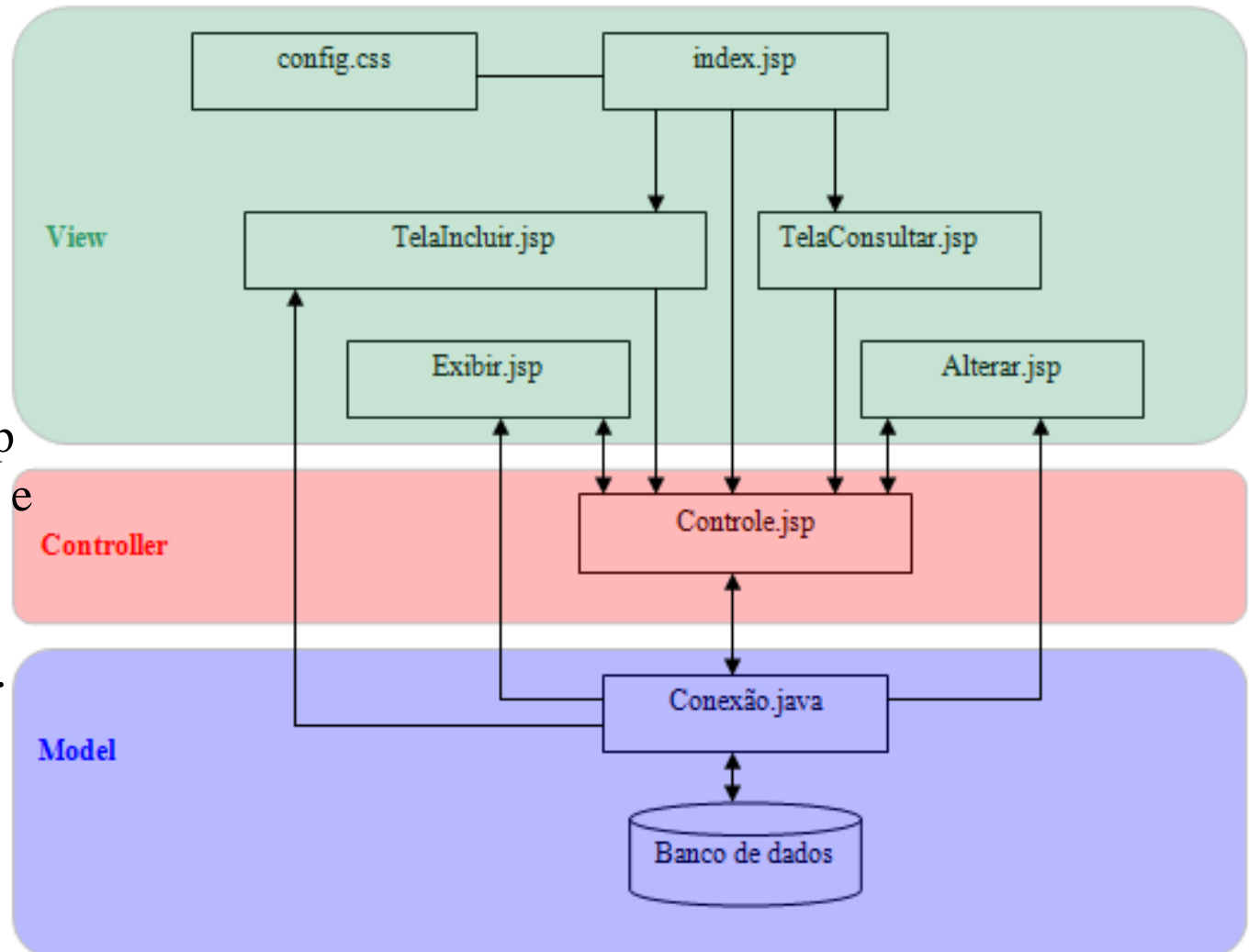


Figura 1 - arquitetura de aplicação web com tecnologia JSP a partir de exemplo original de [Teruel, 2009].

Componentes da camada *View* centralizam suas requisições em um componente da camada *Controller*. Arquivo `Controle.jsp` processa requisições e acessa o banco de dados através da classe `Conexão.java`.

# Projeto Java Web

## Componentes do projeto:

- Menu da aplicação **index.jsp** – está configurado no web.xml para ser o arquivo de inicialização da aplicação. Gera o menu abaixo:

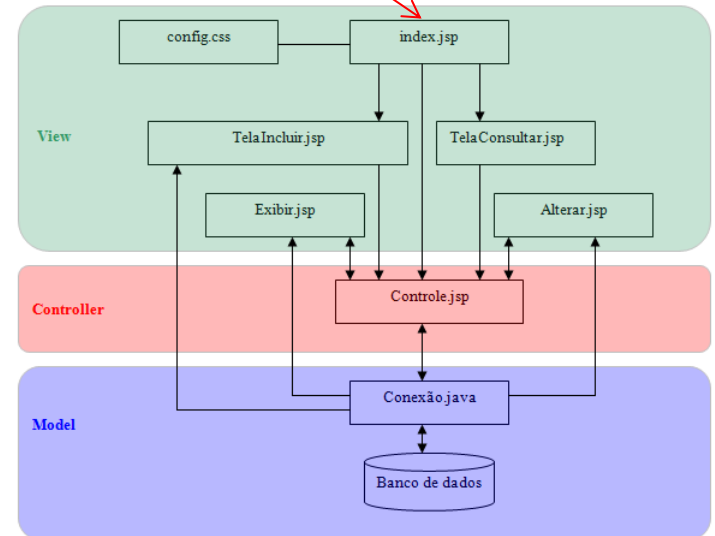


O menu é formado por uma lista de links que têm as funções:

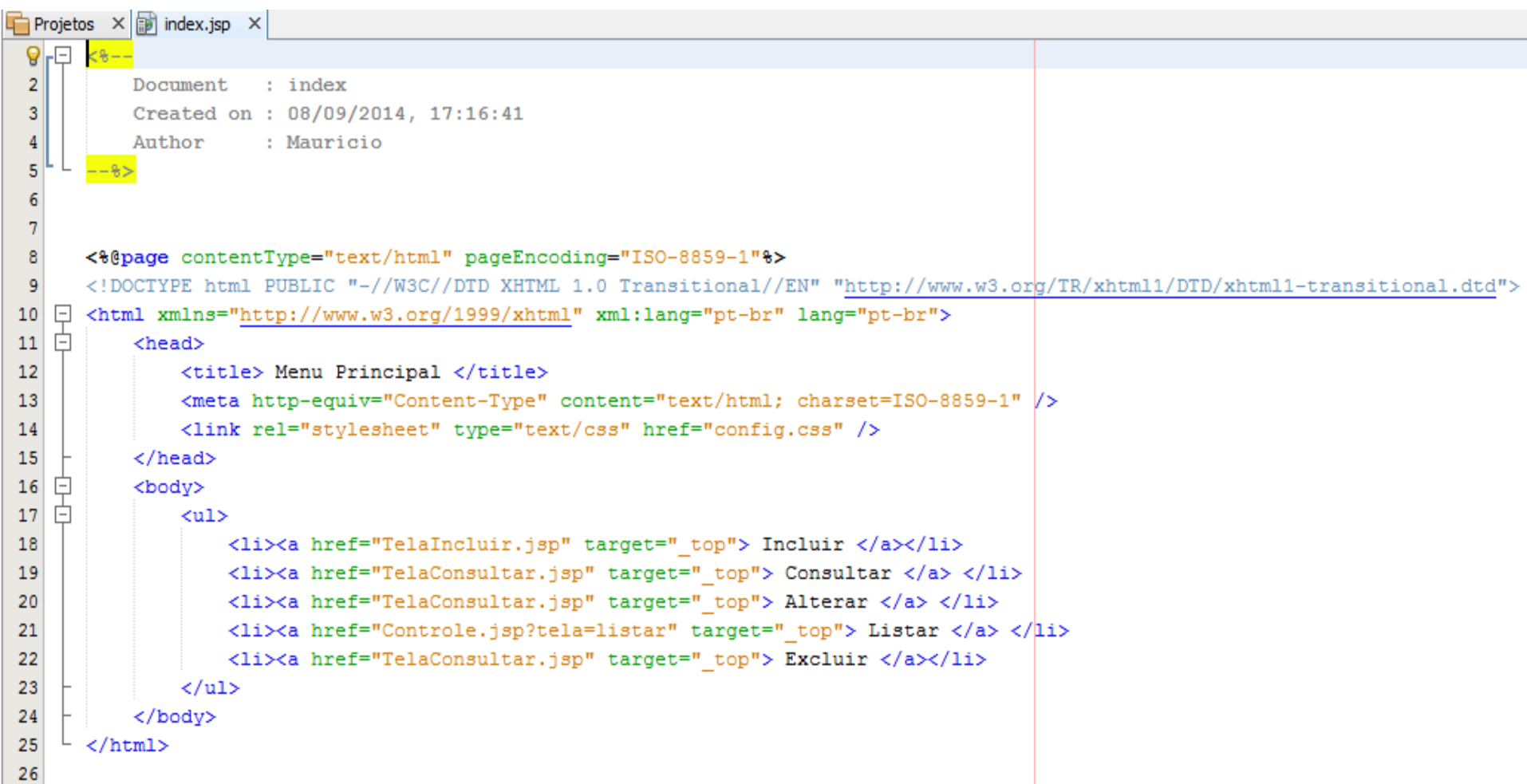
**Incluir:** requisita a execução do arquivo `TelaIncluir.jsp`.

**Consultar, Alterar, Excluir:** requisitam a execução do arquivo `TelaConsultar.jsp`.

**Listar:** requisita a execução do arquivo **Controle.jsp**.  
Listar passa para Controle, como parâmetro, o campo denominado `tela` que contém o valor `listar` por meio do método `get`. Esse parâmetro será recebido no arquivo **Controle.jsp** para identificar a requisição através do evento do mouse (clique) no link `listar`.



# index.jsp



```
1  <%--
2  Document : index
3  Created on : 08/09/2014, 17:16:41
4  Author : Mauricio
5  --%>
6
7
8  <%@page contentType="text/html" pageEncoding="ISO-8859-1"%>
9  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
10 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-br" lang="pt-br">
11 <head>
12 <title> Menu Principal </title>
13 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
14 <link rel="stylesheet" type="text/css" href="config.css" />
15 </head>
16 <body>
17 <ul>
18 <li><a href="TelaIncluir.jsp" target="_top"> Incluir </a></li>
19 <li><a href="TelaConsultar.jsp" target="_top"> Consultar </a> </li>
20 <li><a href="TelaConsultar.jsp" target="_top"> Alterar </a> </li>
21 <li><a href="Controle.jsp?tela=listar" target="_top"> Listar </a> </li>
22 <li><a href="TelaConsultar.jsp" target="_top"> Excluir </a></li>
23 </ul>
24 </body>
25 </html>
26
```



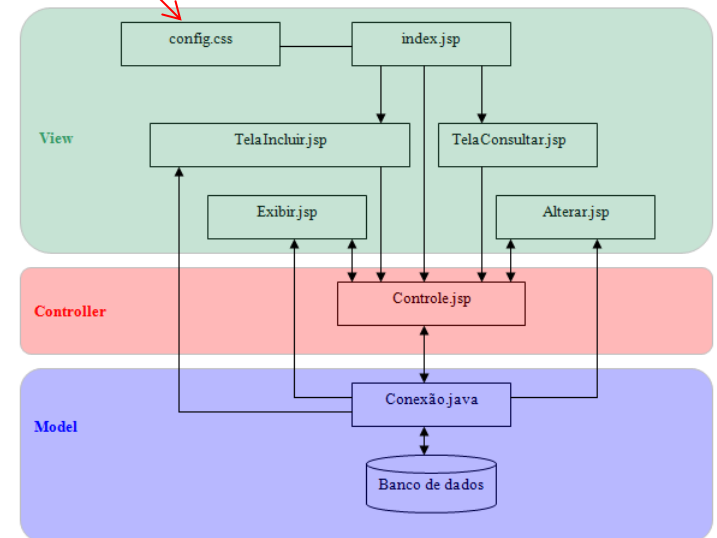
# Projeto Java Web

## Componentes do projeto:

- Folha de estilo **config.css** – é uma folha de estilo usada para formatar os elementos do menu **index.jsp** e demonstrar que o conteúdo HTML dos arquivos JSPs pode ser utilizado pela linguagem CSS.

### Trecho de código de **index.jsp**:

```
...<link rel = "stylesheet" type= "text/css" href= "config.css" />...
```



# config.css

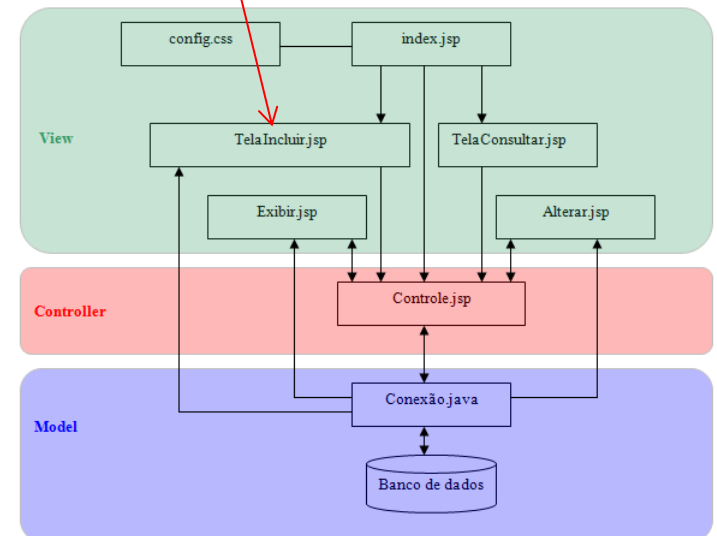
```
Projetos x config.css x
20      TODO customize this sample style
21      Syntax recommendation http://www.w3.org/TR/REC-CSS2/
22      */
23
24  *{
25      margin:0px;
26  }
27
28  ul {
29      list-style: none;
30      background-color:#E6E6FA;
31      padding-top:7px;
32      padding-bottom:7px;
33      text-align:center;
34  }
35
36  ul li {
37      padding-left:30px;
38      display:inline;
39  }
40
41  a {
42      text-decoration:none;
43      font-family:verdana;
44      font-size:14pt;
45      color:#000000;
46  }
47  a:hover {
48      text-decoration:underline;
49      color:#0000ff;
50  }
51
```

# Projeto Java Web

## Componentes do projeto:

- Formulário para realizar cadastro **TelaIncluir.jsp** – formulário de cadastro é exibido ao usuário quando é clicado o link Incluir. O ID que aparece é o da sequência ao maior cadastrado no banco de dados. Este formulário .JSP faz import da classe Conexão.java por utilizar seus métodos.

Área de cadastro	
ID:	2
Nome:	<input type="text"/>
Renda:	<input type="text"/>
<input type="button" value="Salvar"/> <input type="button" value="Limpar"/> <input type="button" value="Voltar"/>	

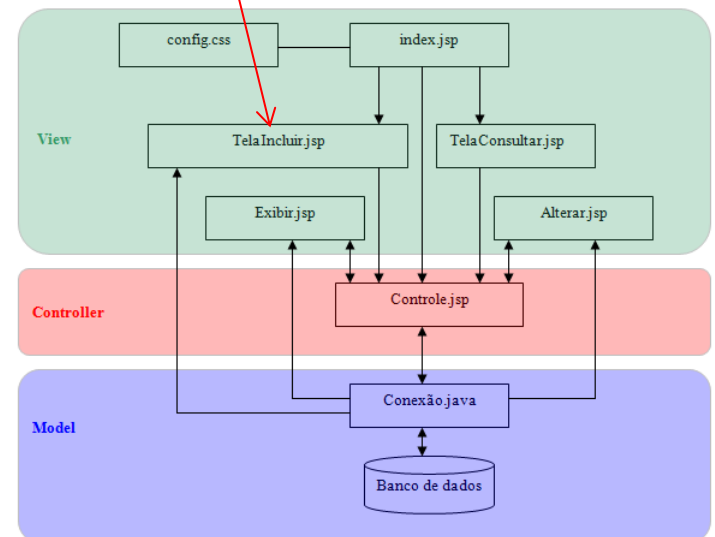


# Projeto Java Web

## Componentes do projeto:

- Formulário para realizar cadastro **TelaIncluir.jsp**.
- Através de objeto da classe Conexao.java, chama o método conectar( ) para fazer a conexão com o banco de dados. Também por intermédio deste objeto, é chamado o método consultar(sql) que fornece um parâmetro de comando sql para uma consulta de ID e depois atualização do ID de novo cliente.

Área de cadastro	
ID:	<input type="text" value="2"/>
Nome:	<input type="text"/>
Renda:	<input type="text"/>
<input type="button" value="Salvar"/> <input type="button" value="Limpar"/> <input type="button" value="Voltar"/>	



# TelaIncluir.jsp

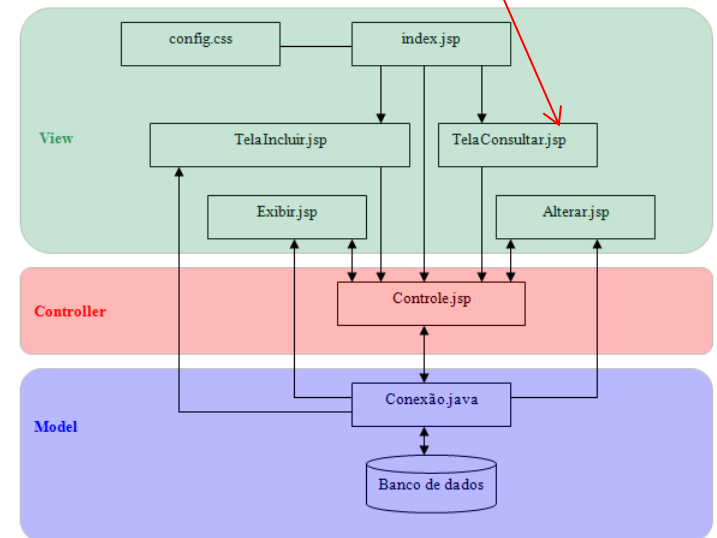
```
<%@page contentType="text/html" pageEncoding="ISO-8859-1"%>
<%@page import="ConexãoBD.Conexão,java.sql.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-br" lang="pt-br">
<head>
<title> Menu Principal </title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
<script type="text/JavaScript">
function valida()
{
    with (document.formulario)
    {
        if (nome.value == "")
        {
            alert("Por favor, digite o nome ");
        }
    }
}
</script>
</head>
<body>
<table border="1">
<tr>
<td><label>Nome:</label></td>
<td><input name="nome" type="text" size="45" /> </td>
</tr>
<tr>
<td><label>Renda:</label></td>
<td><input name="renda" type="text" size="15" /> </td>
</tr>
<tr>
<td colspan="2">
<input type="hidden" name="tela" value="incluir" />
<input type="submit" value="Salvar" />
<input type="reset" value="Limpar" />
<input type="button" value="Voltar" onclick="history.go(-1)" />
</td>
</tr>
</table>
</body>
</html>
```

# Projeto Java Web

## Componentes do projeto:

- Formulário para realizar consulta **TelaConsultar.jsp** – formulário de consulta é exibido ao usuário quando é clicado o link Consultar. Este formulário também atende as opções de alterar, limpar e excluir clientes através da identificação pelo ID.

Controle de Clientes	
ID:	<input type="text"/>
<input type="button" value="Consultar"/>	<input type="button" value="Excluir"/>
<input type="button" value="Alterar"/>	<input type="button" value="Limpar"/>
<input type="button" value="Voltar"/>	



# Projeto Java Web

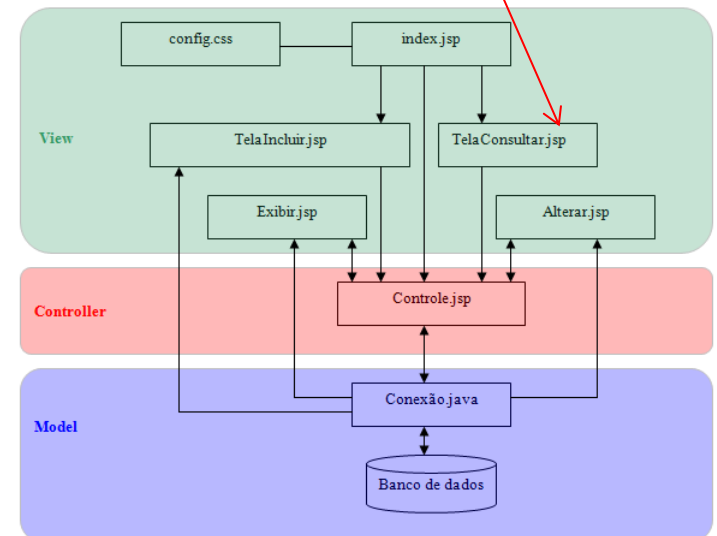
## Componentes do projeto:

- Formulário para realizar consulta **TelaConsultar.jsp**.
- Botões do tipo *submit* “submit( )” enviam os dados do formulário a um arquivo no servidor web utilizando os métodos do tipo *post* ou *get* (method=“post”). Este tipo de botão envia os dados dos campos do formulário utilizando elementos HTML. Os botões do formulário **Controle de Clientes**, quando clicados, submetem dados ao arquivo **Controle.jsp**.

Por exemplo, o botão Consultar submete o Formulário com o campo ID contendo um valor digitado e o campo oculto *hidden* **tela** contendo a palavra **consultar**.

```
...<input type="hidden" name="tela" value="" />...
```

Controle de Clientes	
ID:	<input type="text"/>
<input type="button" value="Consultar"/>	<input type="button" value="Excluir"/>
<input type="button" value="Alterar"/>	<input type="button" value="Limpar"/>
<input type="button" value="Voltar"/>	



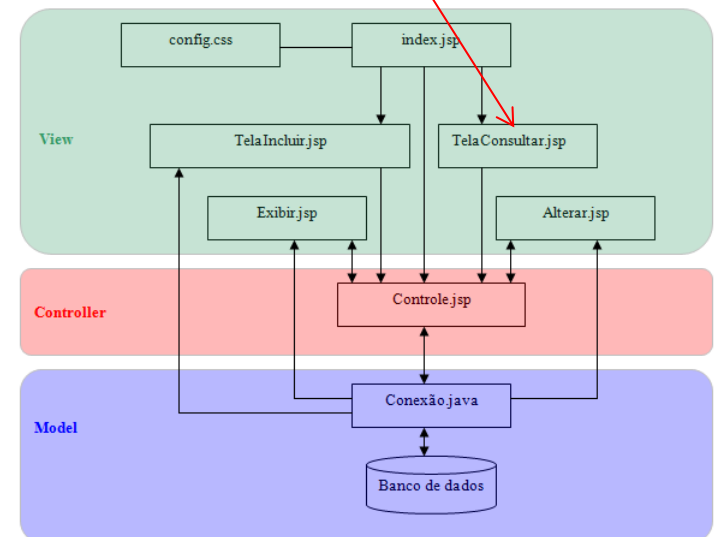
# Projeto Java Web

## Componentes do projeto:

- Formulário para listar clientes: **TelaConsultar.jsp** – formulário de lista é exibido ao usuário quando é clicado o link listar.

ID	Nome	Renda	Excluir	Alterar
1	Maria Clara	300.00	<a href="#">Excluir</a>	<a href="#">Alterar</a>

[Home](#)





# TelaConsultar.jsp

```
<%@page contentType="text/html" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-br" lang="pt-br">
  <head>
    <title> Menu Principal </title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <script type="text/JavaScript">
      function valida()
      {
        with (document.formulario)
        {
          if (id.value == "")
            .
            .
            .
        }
      }
    </script>
  </head>
  <body>
    <table border="1">
      <tr>
        <td><label>ID:</label></td>
        <td><input name="id" type="text" size="10" /> </td>
      </tr>
      <tr>
        <td colspan="2">
          <input type="hidden" name="tela" value="" />
          <input type="button" value="Consultar" onclick="Consultar()" />
          <input type="button" value="Excluir" onclick="Excluir()" />
          <input type="button" value="Alterar" onclick="Alterar()" />
          <input type="reset" value="Limpar" />
          <input type="button" value="Voltar" onclick="history.go(-1)" />
        </td>
      </tr>
    </table>
    </form>
  </body>
</html>
```

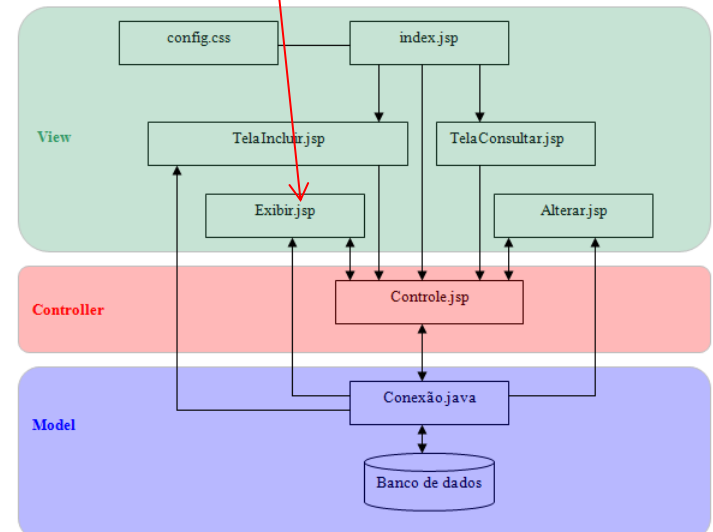
# Projeto Java Web

## Componentes do projeto:

- Apresentação dos dados das pesquisas: **Exibir.jsp** – O arquivo é responsável pela apresentação dos dados retornados de pesquisas no banco de dados. Todo fluxo que se inicia ao clicar nos links **Consultar** ou **Listar** do menu principal **index.jsp** termina nesse arquivo. A opção **Listar** exibe todos os dados dos clientes em ordem alfabética por nome e a opção **Consultar** exibe os dados de apenas um cliente. Para que esse arquivo identifique qual operação deve ser realizada, o conteúdo do campo **ID** recebido na requisição feita por meio do arquivo **Controle.jsp** é verificado.

Para cada linha do formulário resultante da pesquisa estão disponíveis os links **Excluir** e **Alterar**, que respondem respectivamente com requisições ao arquivo **Controle.jsp** com as passagens dos parâmetros **tela** com o valor **excluir** e **ID** contendo o valor id da linha.

ID	Nome	Renda	Excluir	Alterar
1	Maria Clara	300.00	<a href="#">Excluir</a>	<a href="#">Alterar</a>



# Exibir.jsp

```
<%@page contentType="text/html" pageEncoding="ISO-8859-1"%>
<%@page import="ConexãoBD.Conexão,java.sql.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1"
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-br" lang="pt-br">
  <head>
    <title> Controle de fluxo </title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
  </head>
  <body>
    <%
      String sql=null;
      int varId = Integer.parseInt(request.getParameter("id"));
      if (varId==0) {
        sql = "select * from cliente order by nome";
      } else {
        sql = "select * from cliente where id=" + varId + "";
      }
      Conexão conbd = new Conexão();
      conbd.conectar();
      if (conbd.getRetorno() == 0) {
        out.println("Erro na conexão com o banco de dados");
      } else {
        ResultSet rs = conbd.consultar(sql);
        if (conbd.getRetorno() == 0) {
          out.println("<h2 align='center'>Erro na pesquisa</h2>");
          out.println("<br /> <br />");
          out.println("<p align='center'><a href='index.jsp'>Home</a></p>");
        } else {
          if (!rs.next()) {
            out.println("<h2 align='center'>Nenhum registro encontrado</h2>");
          }
        }
      }
    %>
  </body>
</html>
```

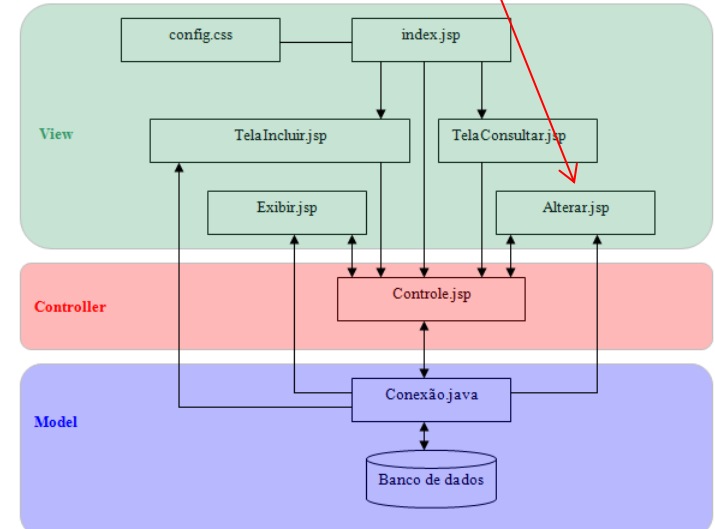
# Projeto Java Web

## Componentes do projeto:

- Apresentação dos dados dos clientes: **Alterar.jsp** – O arquivo não altera os dados do cliente no banco de dados, mas apenas realiza uma pesquisa a partir do ID informado, exibe os dados retornados no formulário e permite que o usuário modifique esses dados. Ao clicar no botão Salvar, os dados do formulário são submetidos ao arquivo Controle.jsp que, pelo método incAltExc da classe Conexão.java, realiza a alteração no banco de dados.

Controle de Clientes	
ID:	<input type="text" value="1"/>
<input type="button" value="Consultar"/> <input type="button" value="Excluir"/> <input type="button" value="Alterar"/> <input type="button" value="Limpar"/> <input type="button" value="Voltar"/>	

Área de alteração	
ID:	<input type="text" value="1"/>
Nome:	<input type="text" value="Maria Clara"/>
Renda:	<input type="text" value="300.00"/>
<input type="button" value="Salvar"/> <input type="button" value="Limpar"/> <input type="button" value="Voltar"/>	



# Alterar.jsp

```
<%@page contentType="text/html" pageEncoding="ISO-8859-1"%>
<%@page import="ConexãoBD.Conexão,java.sql.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-br" lang="pt-br">
  <head>
    <title> Controle de fluxo </title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <script type="text/JavaScript">
      function valida()
      {
        with (document.formulario)
        {
          if (nome.value == "")
```

•  
•  
•

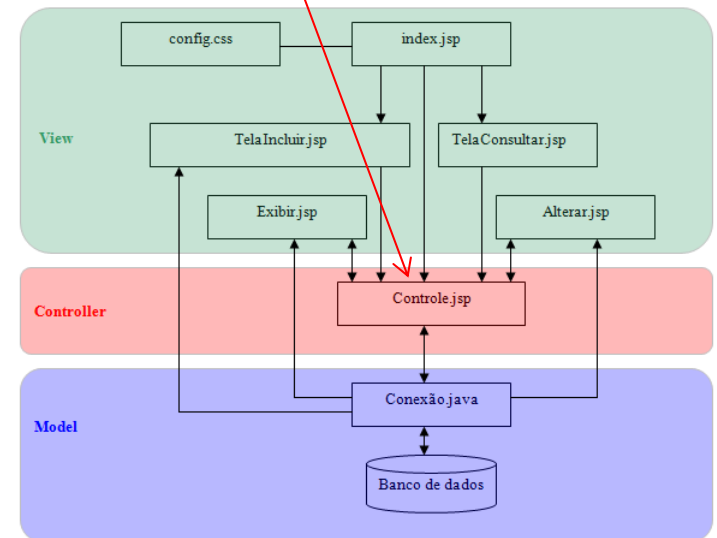
```
      ----
      <td><label>Renda:</label></td>
      <td><input name="renda" type="text" size="15" value="<%out.print(rs.getString(3));%>" /> </td>
    </tr>
    <tr>
      <td colspan="2">
        <input type="hidden" name="tela" value="telaalterar" />
        <input type="submit" value="Salvar" />
        <input type="reset" value="Limpar" />
        <input type="button" value="Voltar" onclick="history.go(-1)" />
      </td>
    </tr>
  </table>
```

•  
•  
•

# Projeto Java Web

## Componentes do projeto:

- Controlador de fluxo da aplicação: **Controle.jsp** – gerencia o fluxo de informações da aplicação. Todas as interações que o usuário faz no sistema passam por esse arquivo antes de chegar à classe **Conexão.java**, que acessa o banco de dados. Faz o import da classe **Conexão.java**, e consequentemente todas as classes do pacote java.sql.



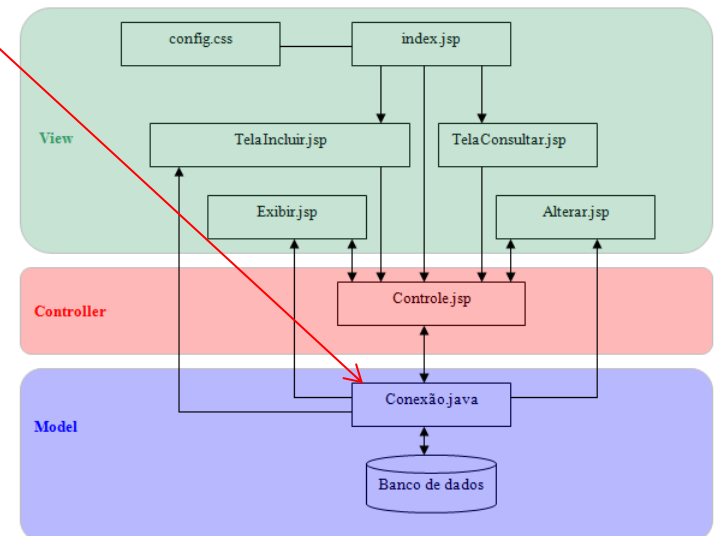
# Controle.jsp

```
<%@page contentType="text/html" pageEncoding="ISO-8859-1"%>
<%@page import="ConexãoBD.Conexão,java.sql.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-br" lang="pt-br">
  <head>
    <title> Controle de fluxo </title>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
  </head>
  <body>
    <%
      String sql = null;
      int varId = 0;
      String varTela = request.getParameter("tela");
      if (varTela.equals("incluir") || varTela.equals("alterar") || varTela.equals("excluir") || varTela.equals("consultar"))
        varId = Integer.parseInt(request.getParameter("id"));
      }
      if (varTela.equals("incluir")) {
        String varNome = request.getParameter("nome");
        double varRenda = Double.parseDouble(request.getParameter("renda"));
        sql = "insert into cliente values(" + varId + ", '" + varNome + "', " + varRenda + ")";
      }
      if (varTela.equals("telaalterar")) {
        String varNome = request.getParameter("nome");
        double varRenda = Double.parseDouble(request.getParameter("renda"));
        sql = "update cliente set nome='" + varNome + "', renda=" + varRenda + " where id=" + varId + ";
      }
      if (varTela.equals("excluir")) {
        sql = "delete from cliente where id=" + varId + ";
      }
    %>
    <%
      Conexão conbd = new Conexão();
      conbd.conectar();
      if (conbd.getRetorno() == 0) {
        out.println("Erro na conexão com o banco de dados");
      }
      if (varTela.equals("consultar")){
        response.sendRedirect("Exibir.jsp?id="+ varId + "");
      }
      if (varTela.equals("listar")){
        response.sendRedirect("Exibir.jsp?id=0");
      }
      conbd.desconectar();
    %>
  </body>
</html>
```

# Projeto Java Web

## Componentes do projeto:

- Classe **Conexão.java** – disponibiliza métodos e atributos que podem ser acessados pelos arquivos JSP da aplicação. Essa classe possui um método para fazer a conexão com o banco de dados “conectar( )”, um método para incluir, alterar ou excluir registros “incAlcExc( )”; possui também um método para pesquisar registros “consultar( )” e outro método para encerrar a conexão com o banco de dados “desconectar( )”.





# Conexao.java

- 
- 
- 

```
public void incAltExc(String sql) {  
  
    try {  
        st.executeUpdate(sql);  
        retorno = 1;  
    }  
    catch (SQLException e) {  
        if (e.getErrorCode() == 1582) {  
            retorno = 2;  
        } else{ retorno = 0; }  
    }  
}  
  
public ResultSet consultar(String sql) {  
    ResultSet rs = null;  
    try {  
        rs = st.executeQuery(sql);  
        retorno = 1;  
    } catch (SQLException ex) {  
        retorno = 0;  
    }  
    return rs;  
}
```

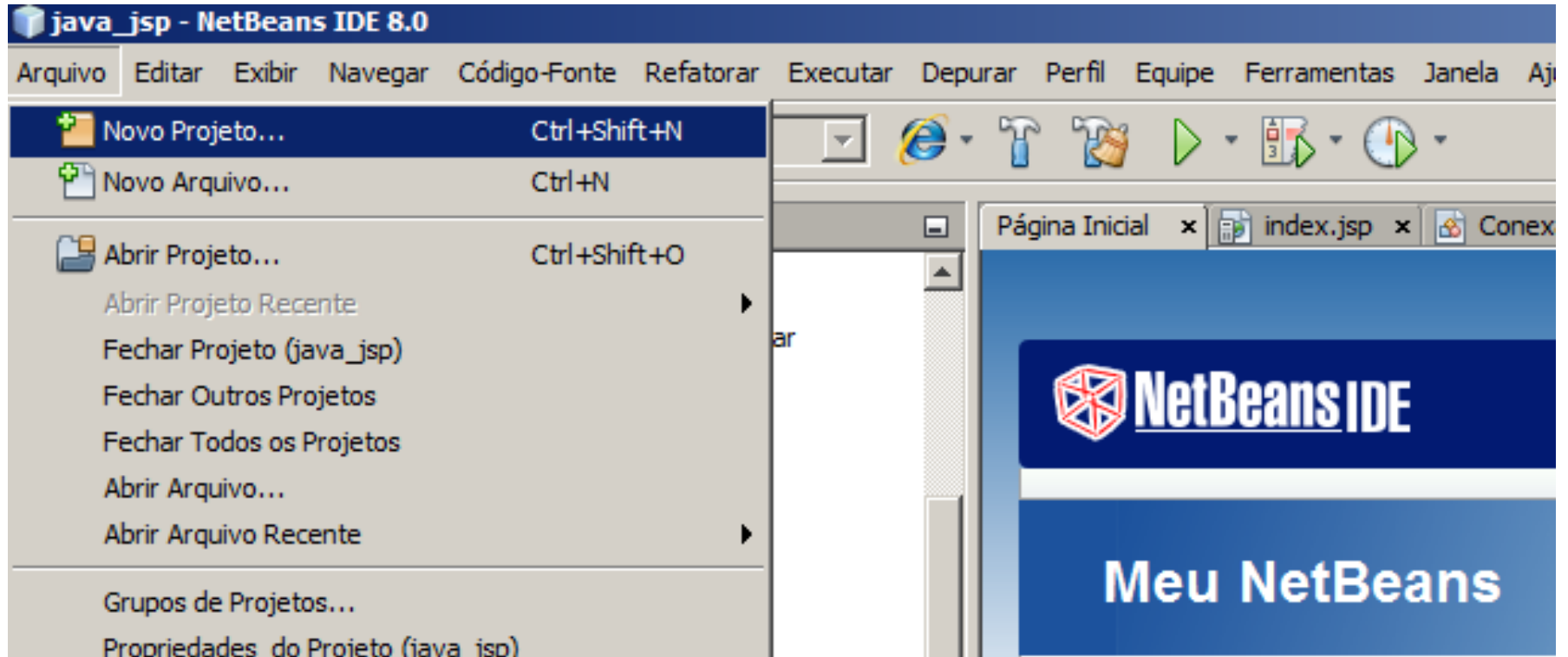
- 
- 
-

# Projeto Java Web

```
public void conectar() {  
    try {  
        Class.forName("com.mysql.jdbc.Driver");  
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/bd007", "mauricio", "brunoa");  
        st = con.createStatement();  
        retorno = 1;  
    } catch (ClassNotFoundException ex) {  
        retorno = 0;  
    } catch (Exception ex1) {  
        retorno = 0;  
    }  
}
```

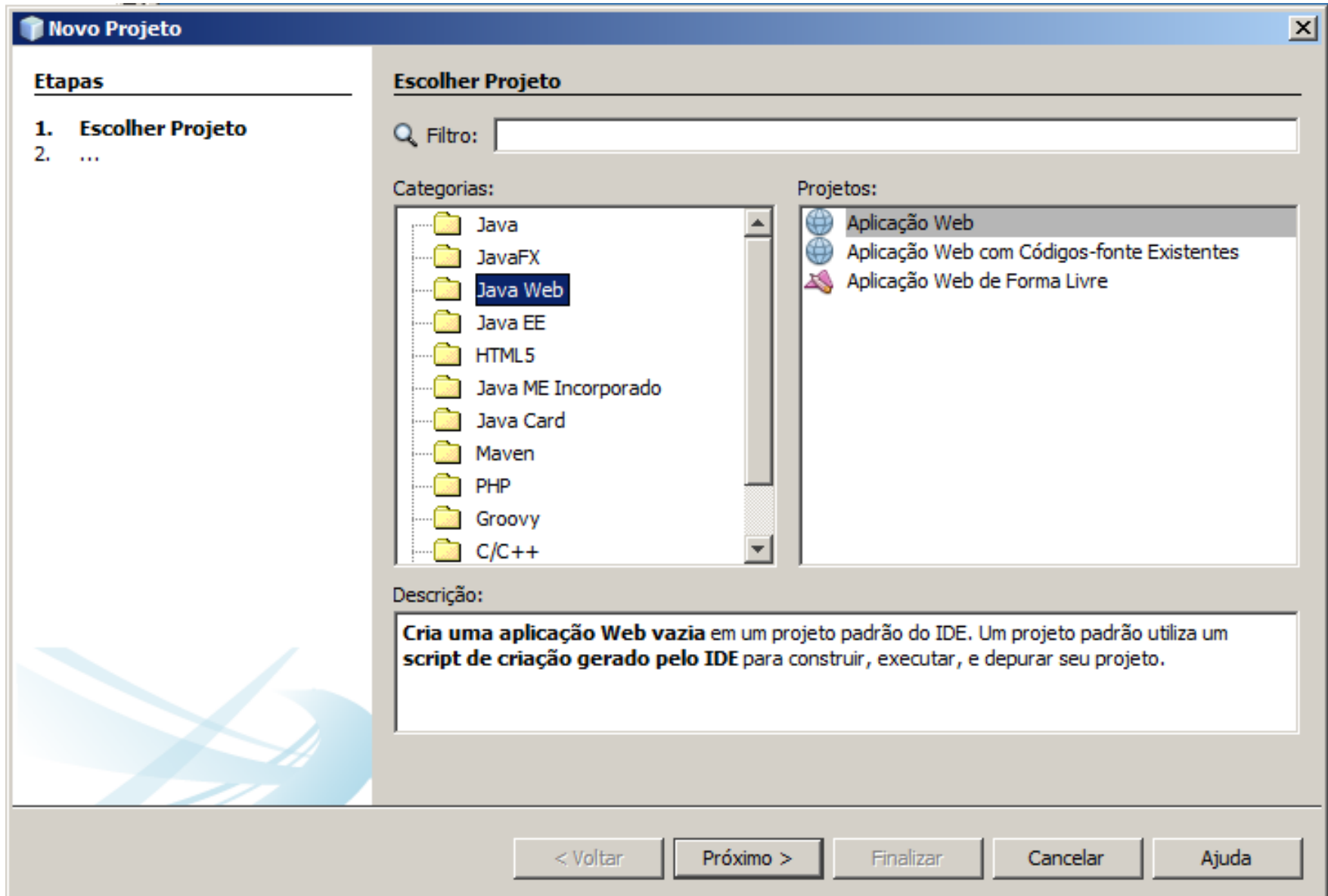
# Projeto Java Web

Abrir o *NetBeans* e criar novo projeto...



# Projeto Java Web

... fazer a opção de Aplicação Web e “clique” – próximo...



# Projeto Java Web

... criar uma pasta para o projeto e nomeá-lo; no exemplo nome = java\_web\_I ...

Propriedades do Projeto - java\_web\_I

**Categorias:**

- Códigos-fonte
- Frameworks
- Spring Framework
- Bibliotecas
- Arquivos JavaScript
- Pré-processadores de CSS
- Construir
  - Compilação
  - Encapsulamento
  - Documentação
- Executar
- Cabeçalhos da Licença
- Formatação

Pasta do Projeto: C:\Users\cmari\_000\Documents\java\_web\java\_web\_I

Pasta das Páginas Web: web Procurar...

Pasta WEB-INF: web/WEB-INF Procurar...

Pastas de Pacotes de Códigos-fonte:

Pasta de Pacotes	Label
src/java	Pacotes de Códigos-fonte

Adicionar Pasta...  
Remover  
Mover para Cima  
Mover para Baixo

Pastas de Pacotes de Teste:

Pasta de Pacotes	Label
test	Pacotes de Teste

Adicionar Pasta...  
Remover  
Mover para Cima  
Mover para Baixo

Formato Código-fonte/Binário: JDK 8 Inclui/Exclui...

Codificação: UTF-8

OK Cancelar Ajuda

# Projeto Java Web

... selecionar o servidor web: no exemplo é utilizado o *GlassFish*...

**Novo Aplicação Web**

**Passos**

1. Escolha o projeto
2. Nome e local
- 3. Servidor e configurações**
4. Frameworks

**Servidor e configurações**

Adicionar ao aplicativo corporativo: <Nenhuma>

Servidor: GlassFish Server 3.x

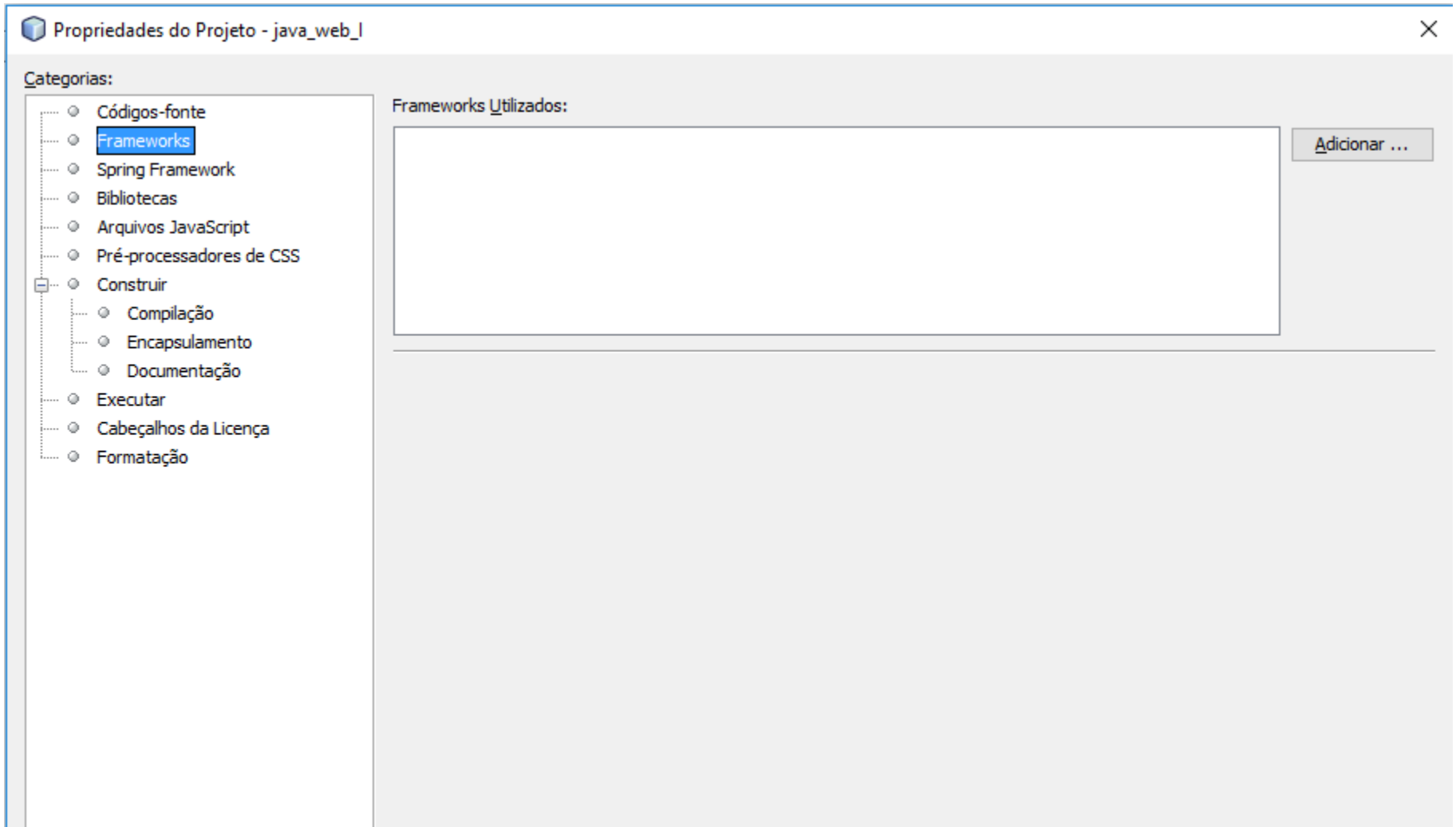
Versão do Java EE: Java EE 6 Web ☐ Habilitar injeção de contextos e dependências

Caminho do contexto: /WebApplication1

< Voltar   Próximo >   Finalizar   Cancelar   Ajuda

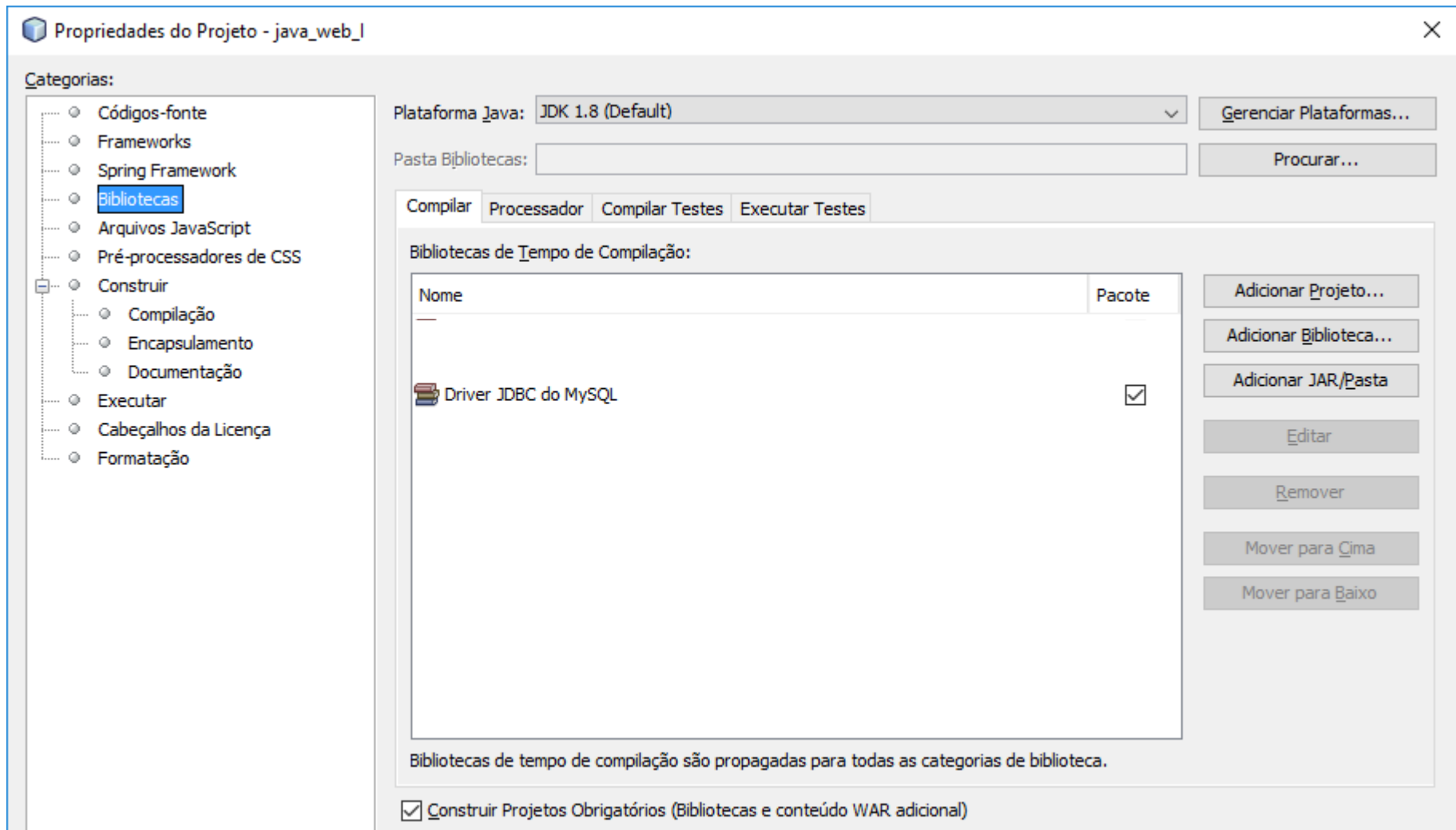
# Projeto Java Web

... no projeto não estarão sendo utilizados *frameworks* específicos...



# Projeto Java Web

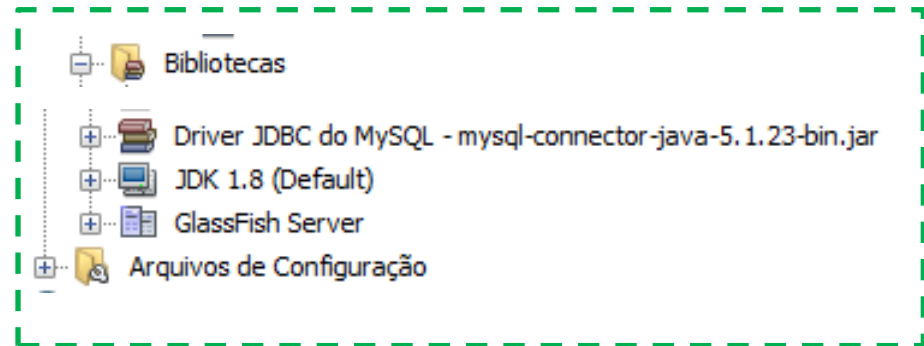
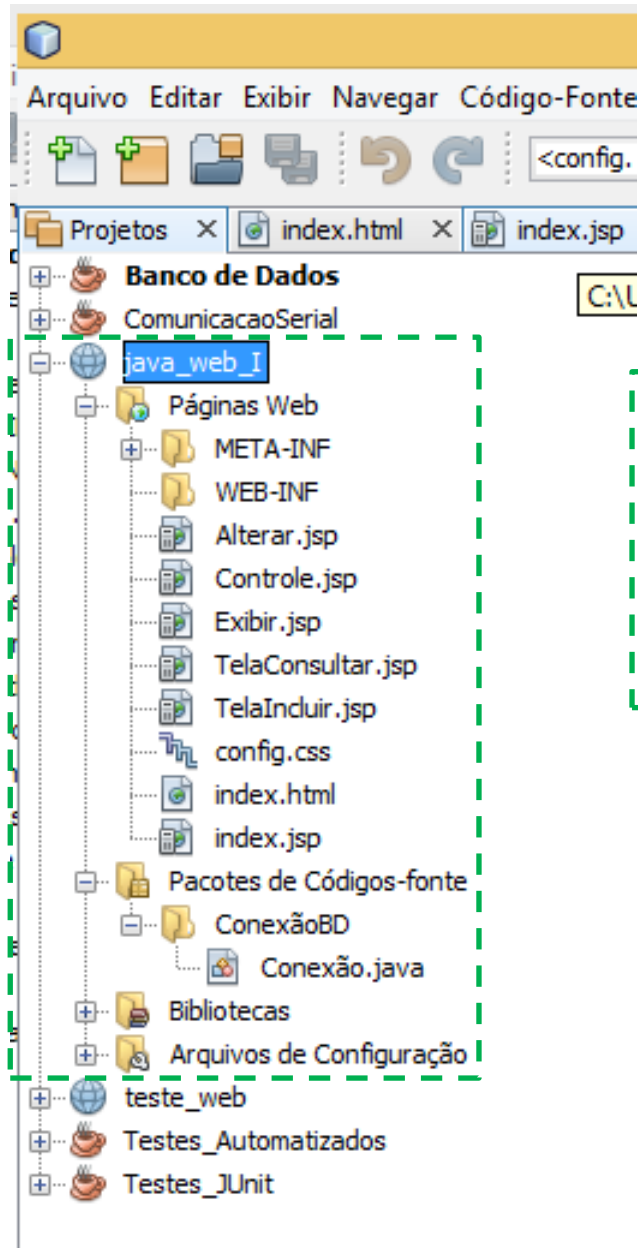
... será inserido o *driver* JDBC do banco de dados *MySQL* em Bibliotecas...





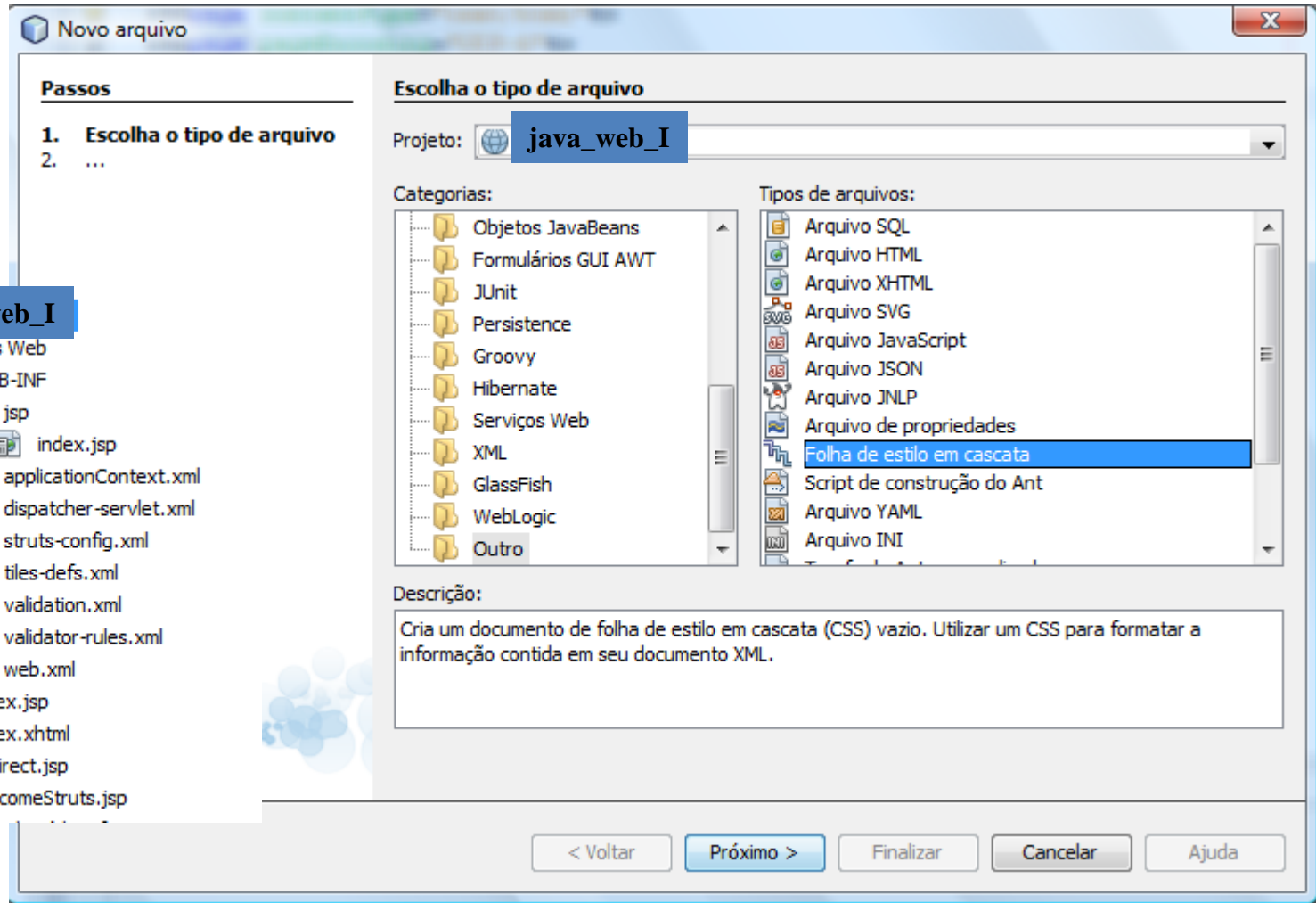
# Projeto Java Web

... visão geral dos componentes que farão parte do projeto ...



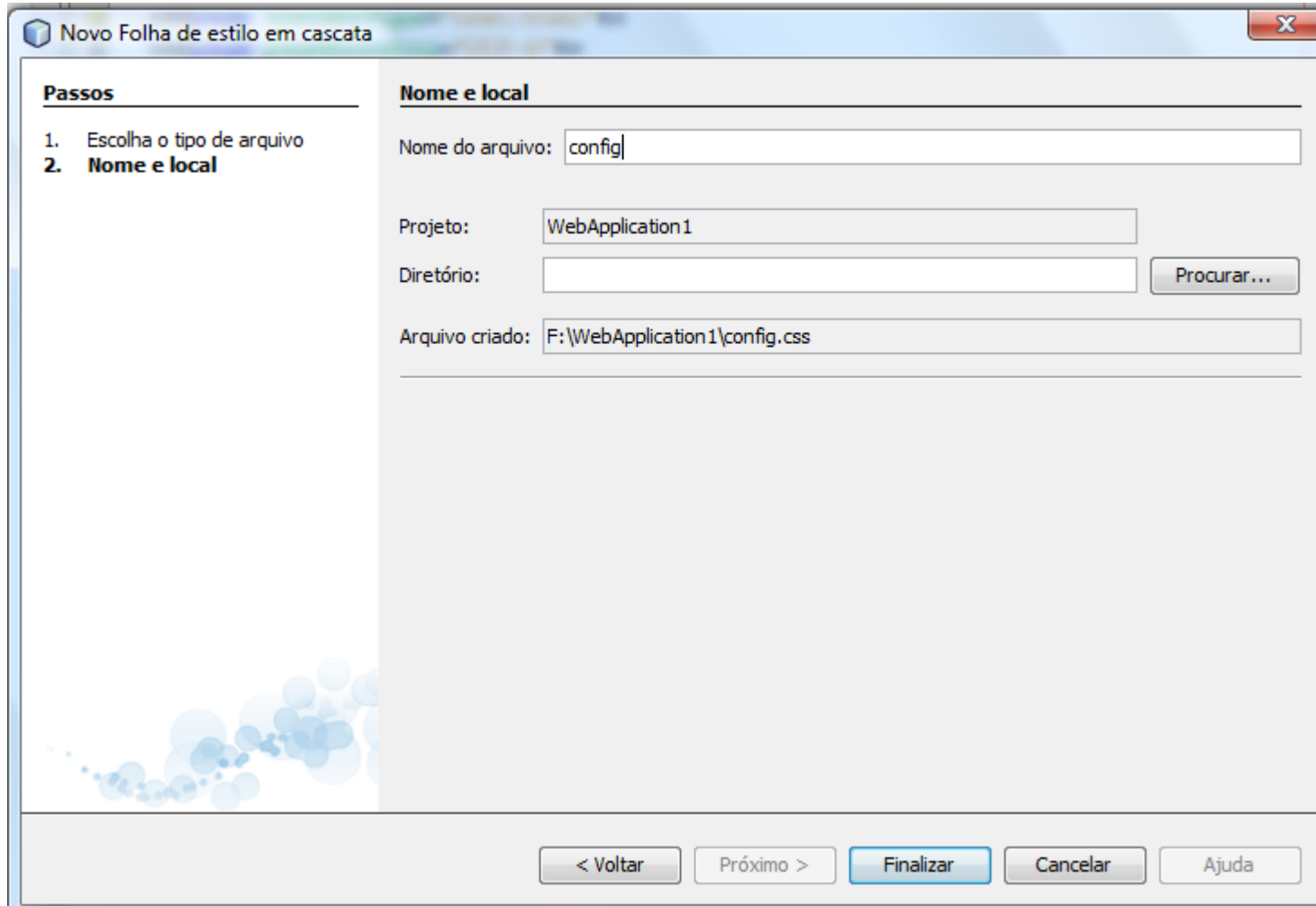
# Projeto Java Web

... inserir o arquivo config.css no projeto...



# Projeto Java Web

... inserir o arquivo config.css no projeto...



**Novo Folha de estilo em cascata**

**Passos**

1. Escolha o tipo de arquivo
- 2. Nome e local**

**Nome e local**

Nome do arquivo:

Projeto:

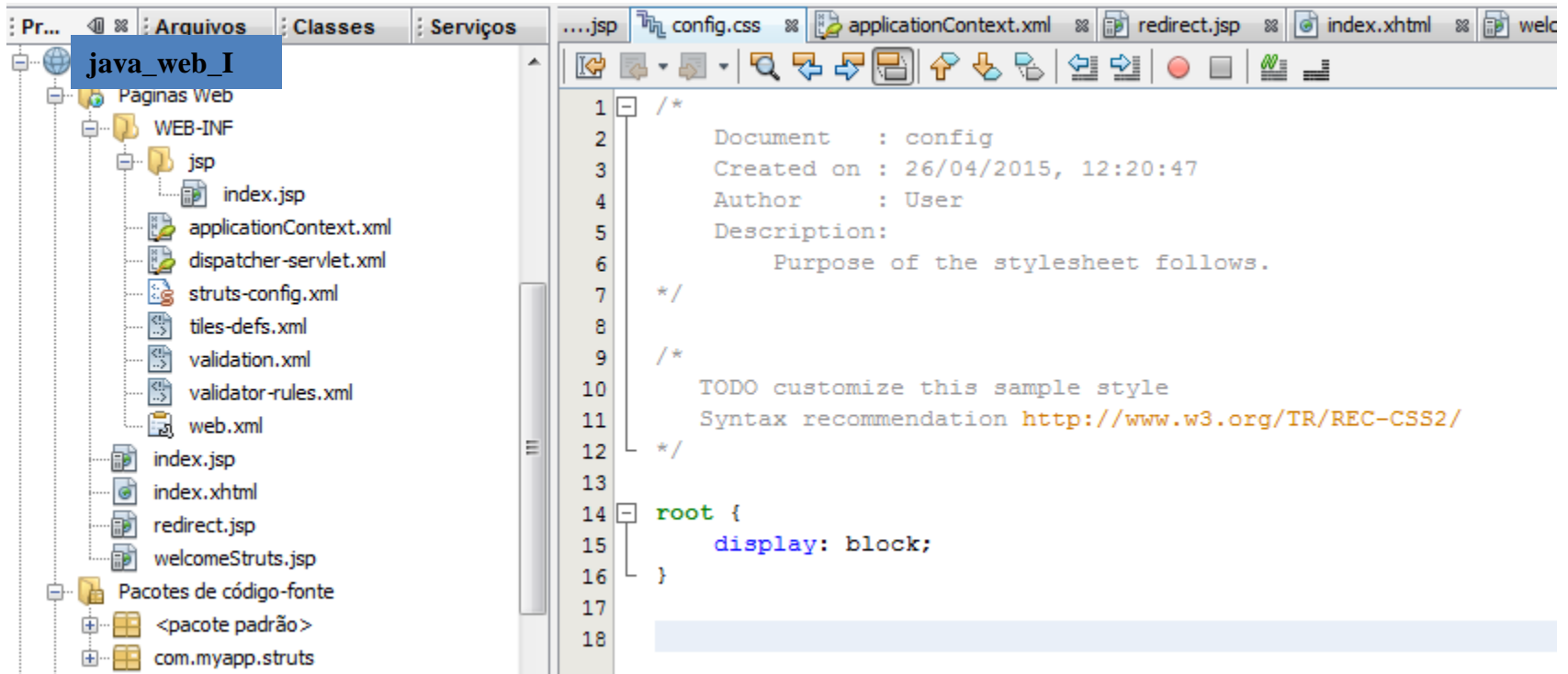
Diretório:

Arquivo criado:

< Voltar   Próximo >   **Finalizar**   Cancelar   Ajuda

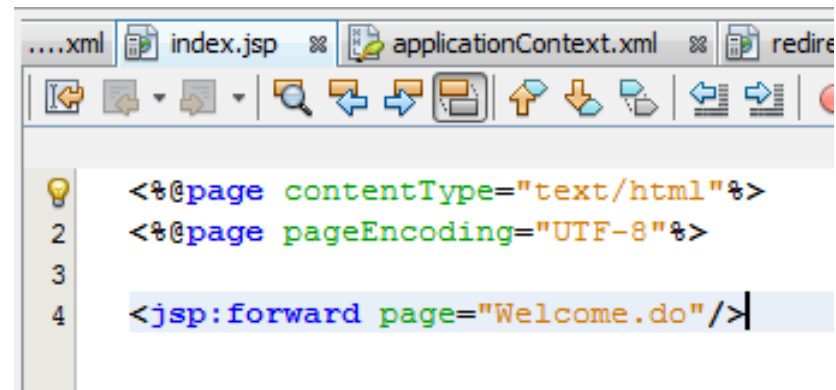
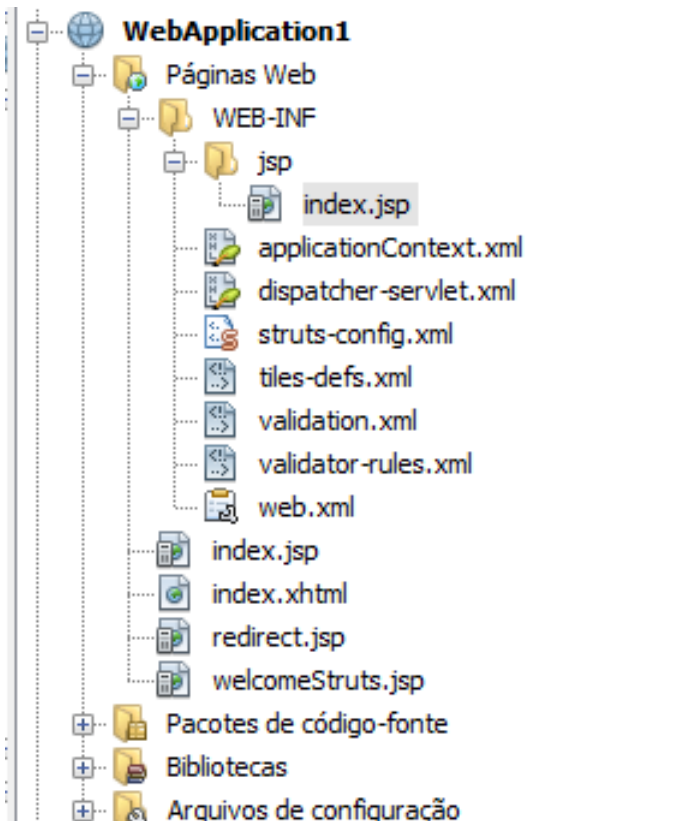
# Projeto Java Web

... arquivo config.css no projeto...



# Projeto Java Web

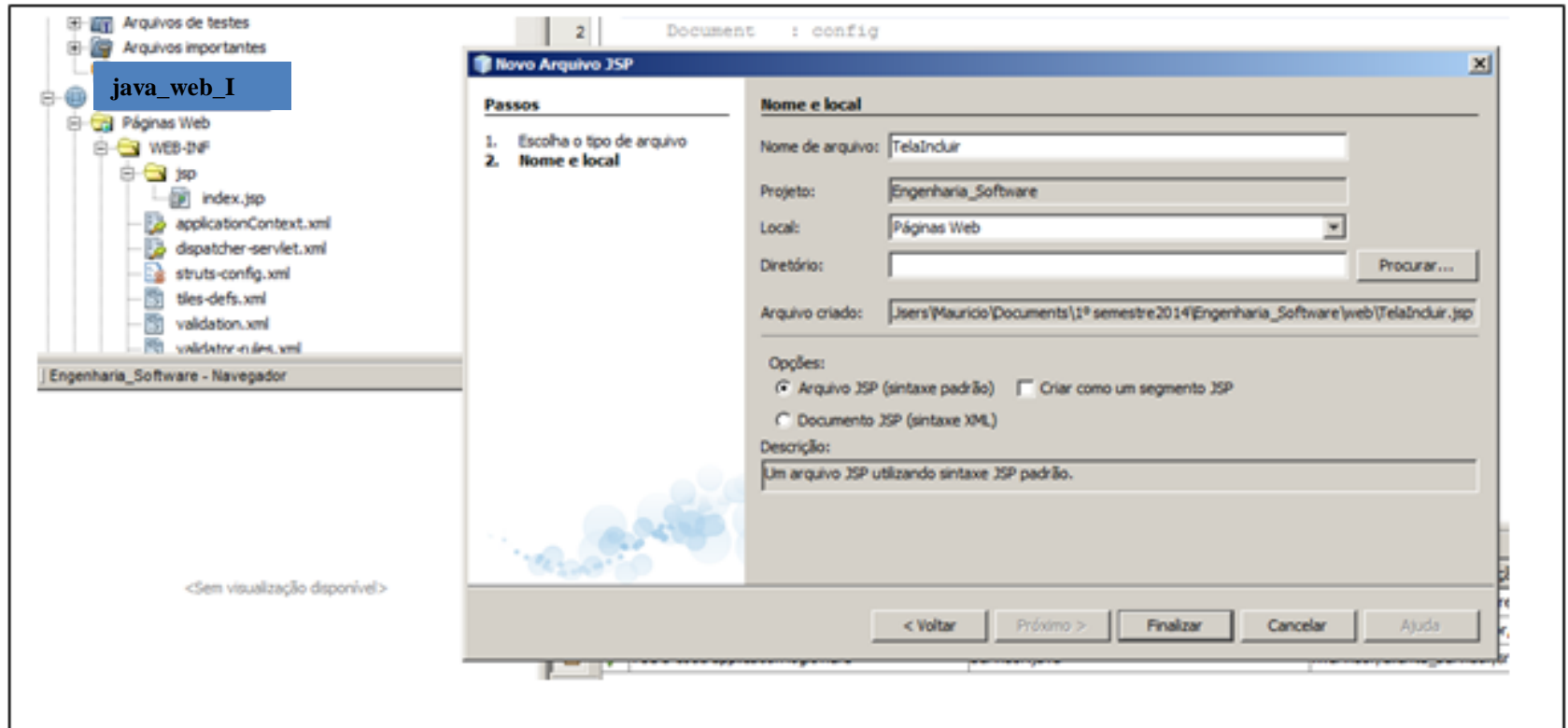
Todos os arquivos JSP e Java que serão criados têm os códigos-fontes fornecidos; depois de criá-los, copiar os respectivos códigos nos arquivos, começando pelo **index.jsp**.



# Projeto Java Web

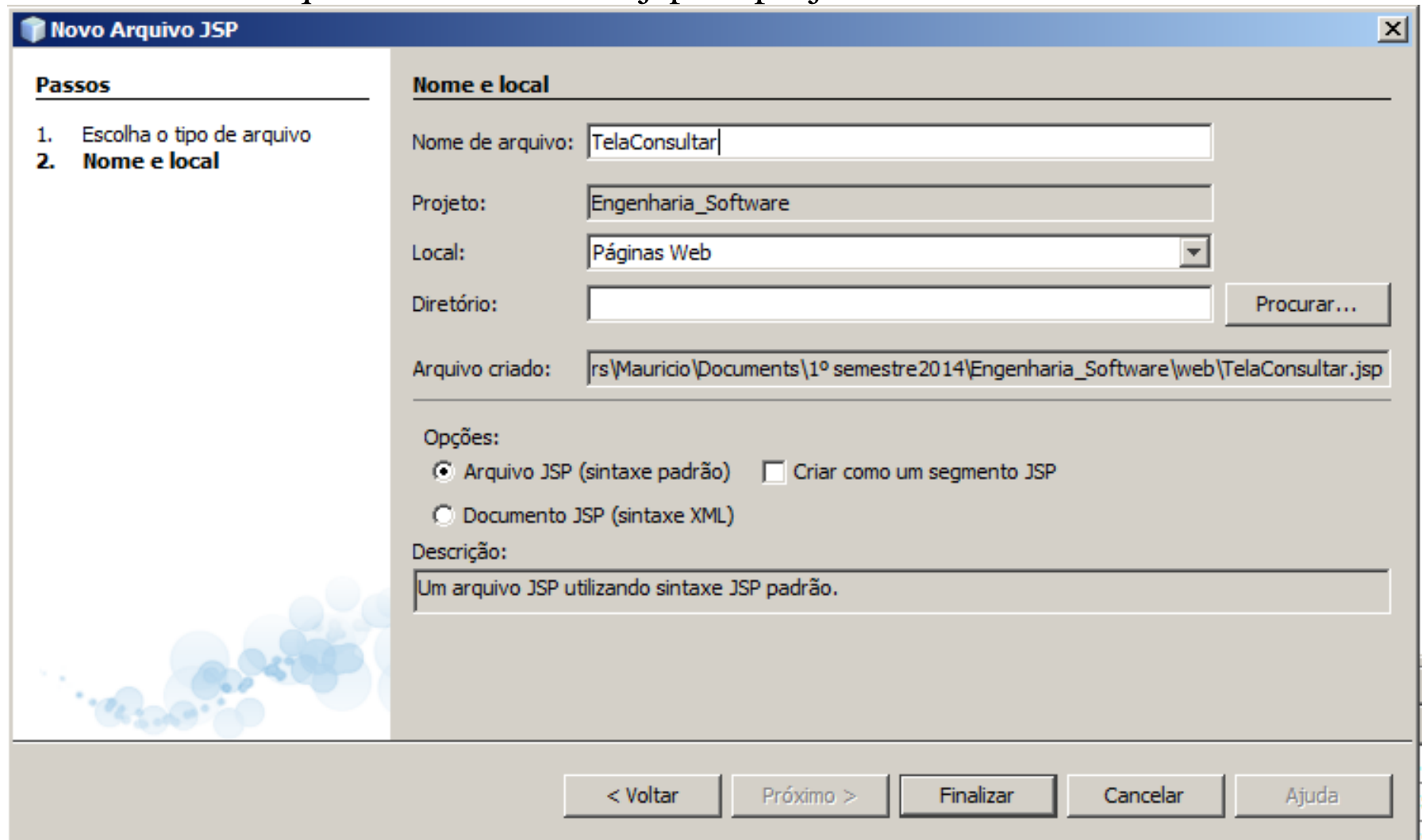
... Inserir arquivo TelaIncluir.jsp no projeto...

Criação de arquivos JSP



# Projeto Java Web

... Inserir arquivo TelaConsultar.jsp no projeto...



**Novo Arquivo JSP**

**Passos**

1. Escolha o tipo de arquivo
2. **Nome e local**

**Nome e local**

Nome de arquivo:

Projeto:

Local:

Diretório:

Arquivo criado:

**Opções:**

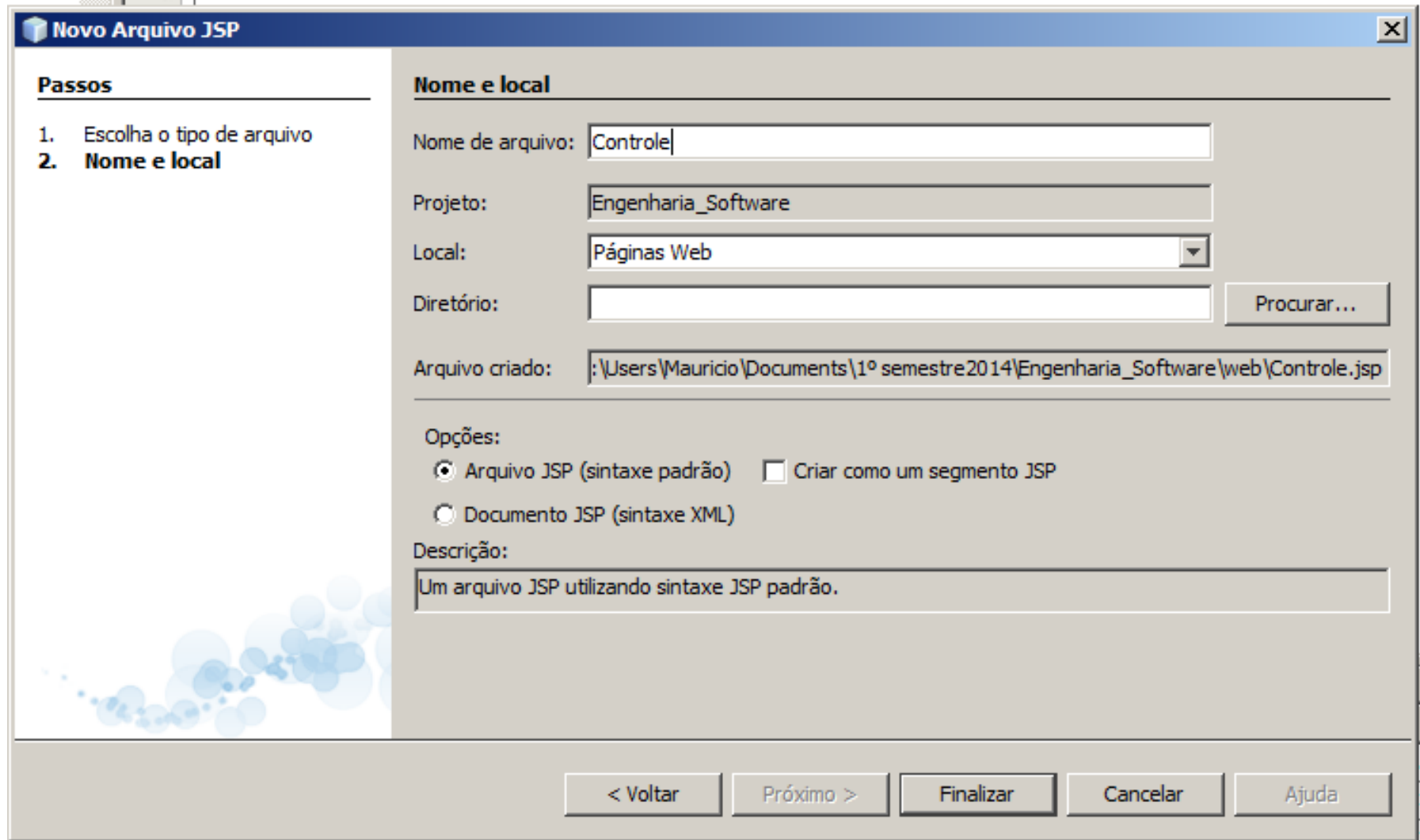
☒ Arquivo JSP (sintaxe padrão) ☐ Criar como um segmento JSP

☐ Documento JSP (sintaxe XML)

**Descrição:**

# Projeto Java Web

... Inserir arquivo Controle.jsp no projeto...



**Novo Arquivo JSP**

**Passos**

1. Escolha o tipo de arquivo
2. **Nome e local**

**Nome e local**

Nome de arquivo:

Projeto:

Local:

Diretório:

Arquivo criado:

**Opções:**

☒ Arquivo JSP (sintaxe padrão) ☐ Criar como um segmento JSP

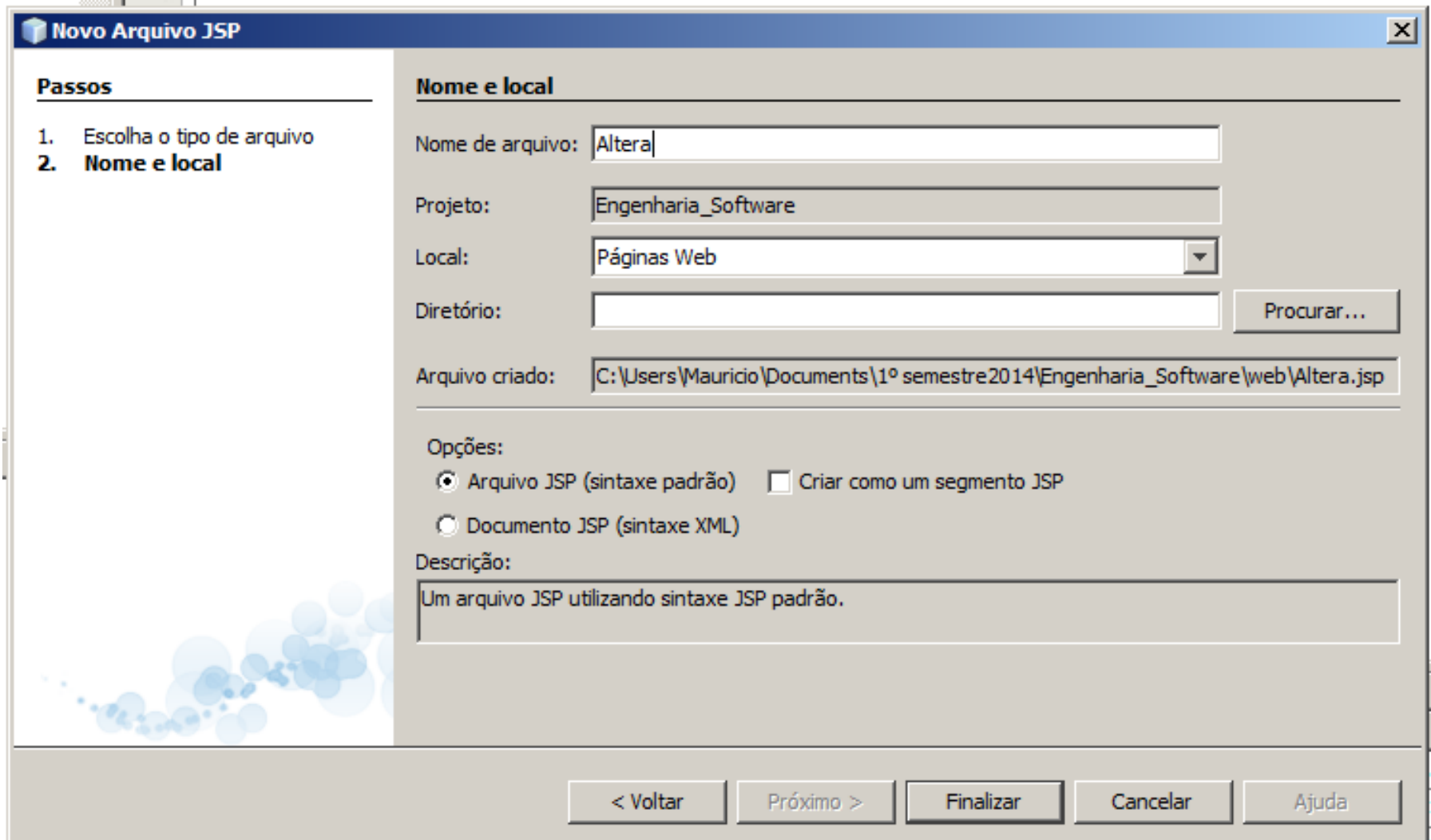
☐ Documento JSP (sintaxe XML)

**Descrição:**



# Projeto Java Web

... Inserir arquivo Alterar.jsp no projeto...



**Novo Arquivo JSP**

**Passos**

1. Escolha o tipo de arquivo
2. **Nome e local**

**Nome e local**

Nome de arquivo:

Projeto:

Local:

Diretório:

Arquivo criado:

Opções:

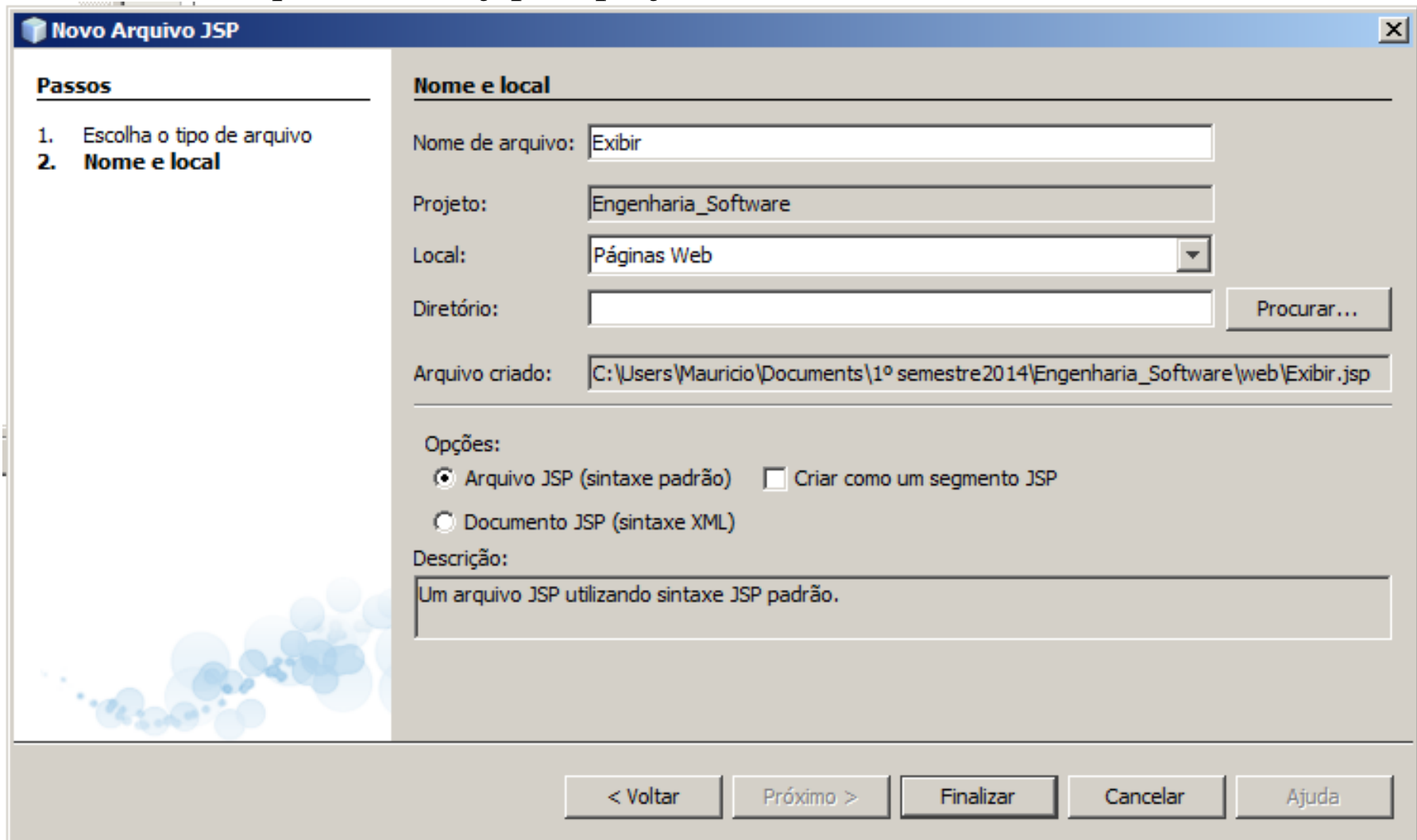
☒ Arquivo JSP (syntaxe padrão) ☐ Criar como um segmento JSP

☐ Documento JSP (syntaxe XML)

Descrição:

# Projeto Java Web

... Inserir arquivo Exibir.jsp no projeto...



**Novo Arquivo JSP**

**Passos**

1. Escolha o tipo de arquivo
2. **Nome e local**

**Nome e local**

Nome de arquivo:

Projeto:

Local:

Diretório:

Arquivo criado:

Opções:

☒ Arquivo JSP (sintaxe padrão) ☐ Criar como um segmento JSP

☐ Documento JSP (sintaxe XML)

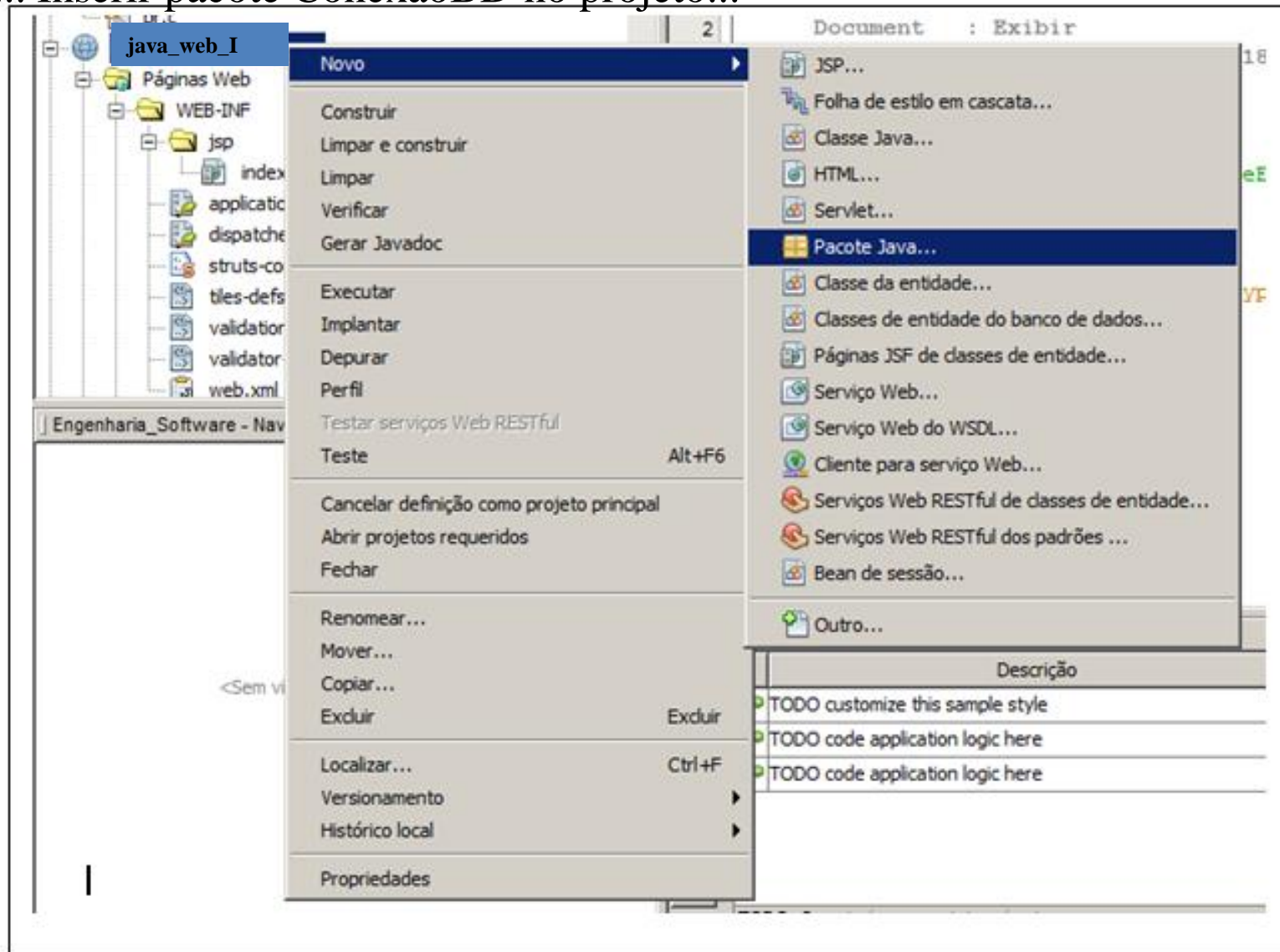
Descrição:

< Voltar   Próximo >   Finalizar   Cancelar   Ajuda

# Projeto Java Web

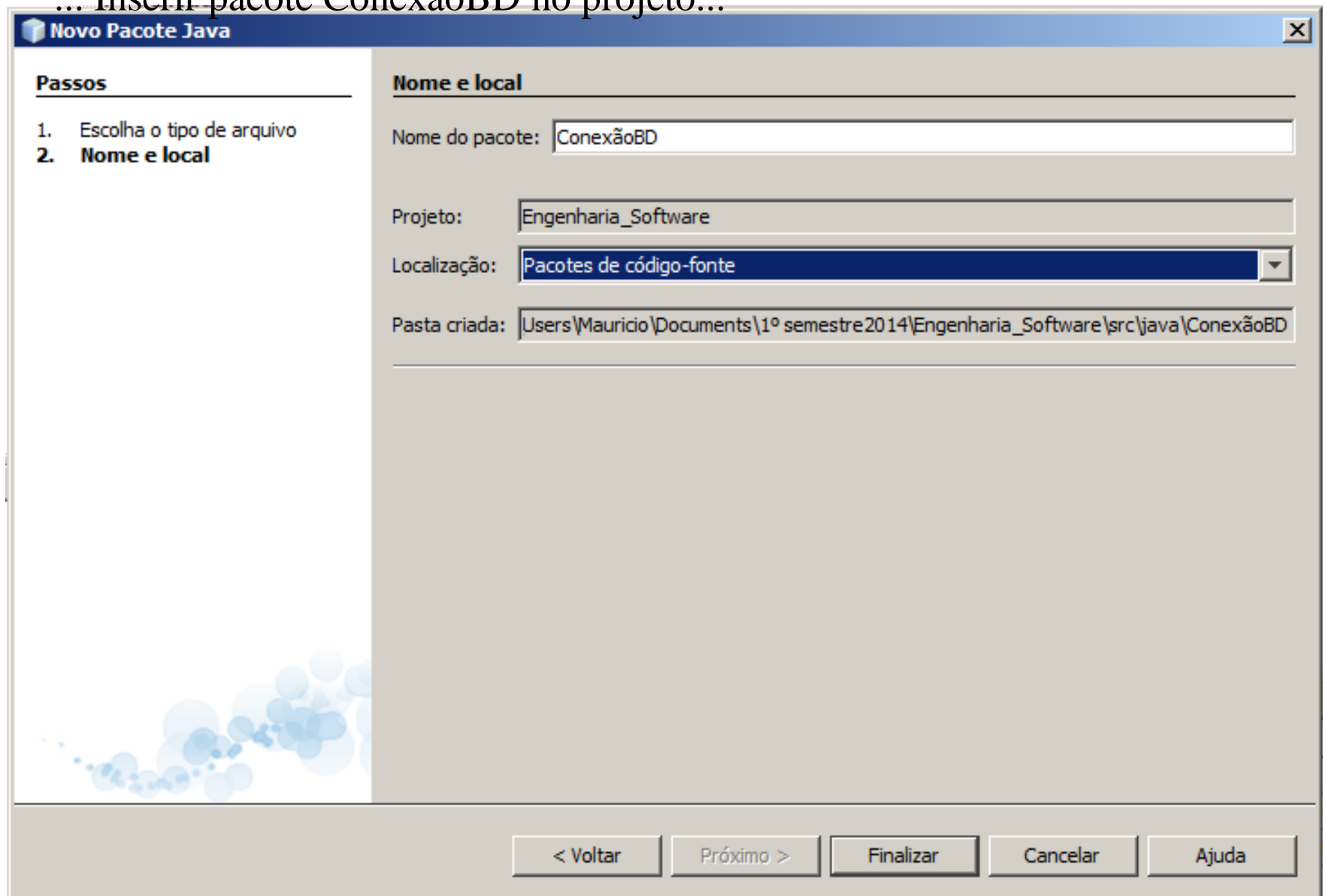
Criação de pacote com classes Java:

... Inserir pacote ConexãoBD no projeto...



# Projeto Java Web

... Inserir pacote ConexãoBD no projeto...



**Novo Pacote Java**

**Passos**

1. Escolha o tipo de arquivo
2. **Nome e local**

**Nome e local**

Nome do pacote:

Projeto:

Localização:

Pasta criada:

< Voltar    Próximo >    Finalizar    Cancelar    Ajuda

# Projeto Java Web

... Inserir classe Conexao.java no projeto...

The screenshot shows an IDE interface with a project named 'pr.c'. The project structure on the left includes a folder 'java\_web\_I' containing a 'Paginas web' folder, which in turn contains a 'WEB-INF' folder. Inside 'WEB-INF', there is a 'jsp' folder and several files: 'index', 'applicatio', 'dispatche', 'struts-co', 'tiles-defs', 'validation', 'validator', and 'web.xml'. Below these are 'Altera.jsp' and 'Controle.jsp'. A context menu is open over the 'java\_web\_I' folder, listing various actions. The 'Novo' (New) option is selected, which has opened a sub-menu. In this sub-menu, 'Classe Java...' (Java Class...) is highlighted. Other options in the sub-menu include 'Pacote Java...', 'JSP...', 'Folha de estilo em cascata...', 'HTML...', 'Servlet...', 'Classe da entidade...', 'Classes de entidade do banco de dados...', 'Páginas JSF de classes de entidade...', 'Serviço Web...', 'Serviço Web do WSDL...', 'Cliente para serviço Web...', 'Serviços Web RESTful de classes de entidade...', 'Serviços Web RESTful dos padrões ...', and 'Bean de sessão...'. At the bottom of the sub-menu is an 'Outro...' (Other...) option.

Engenharia\_Software - Nav

Descrição	
* TODO customize this sample style	co
* TODO code application logic here	Cli
* TODO code application logic here	Se


# Projeto Java Web

Novo Classe Java

Passos

1. Escolha o tipo de arquivo

2. **Nome e local**



Nome e local

Nome da classe:

Projeto:

Localização:

Pacote:

Arquivo criado:

< Voltar

Próximo >

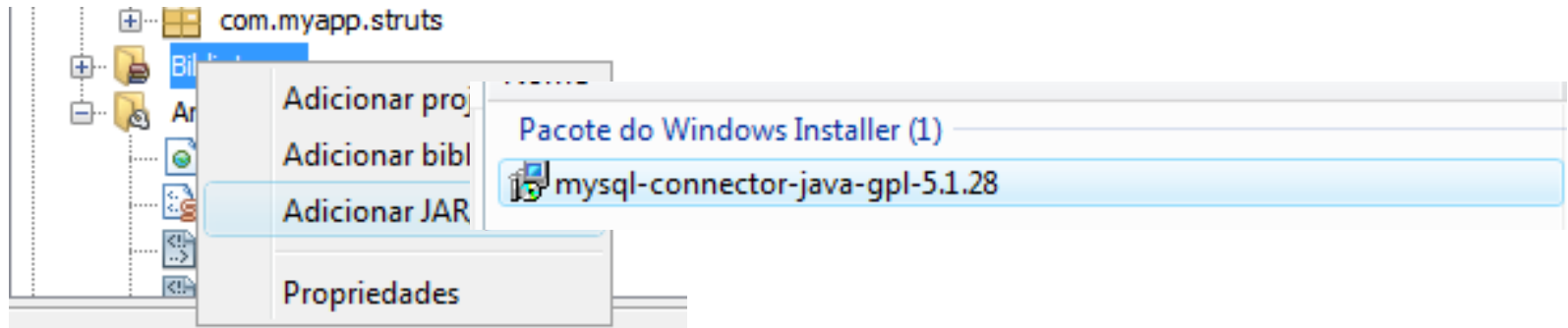
Finalizar

Cancelar

Ajuda

# Projeto Java Web

Banco de dados *MySQL* – inserção de conector J (driver) no projeto Java web: adicionar como arquivo JAR em Bibliotecas.

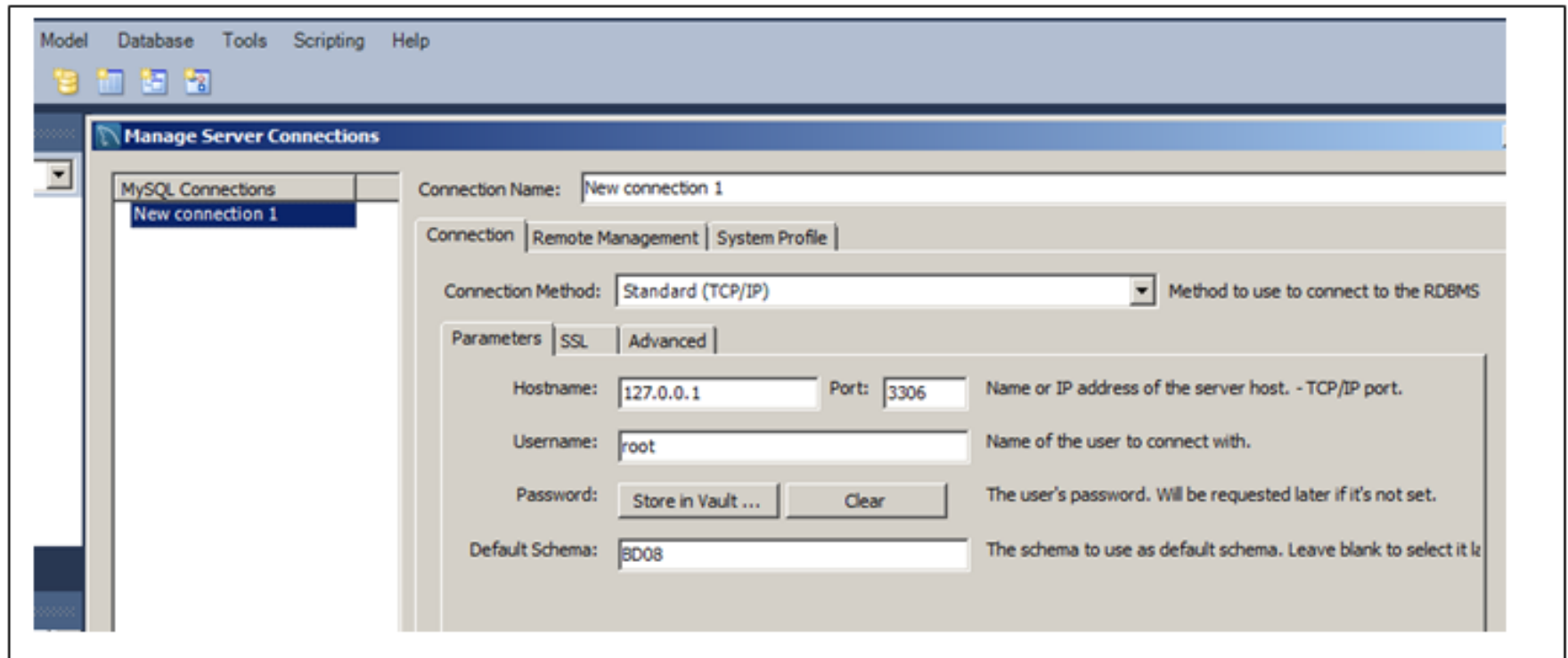


# Projeto Java Web

Criação do banco de dados com MySQL

29-07-2014

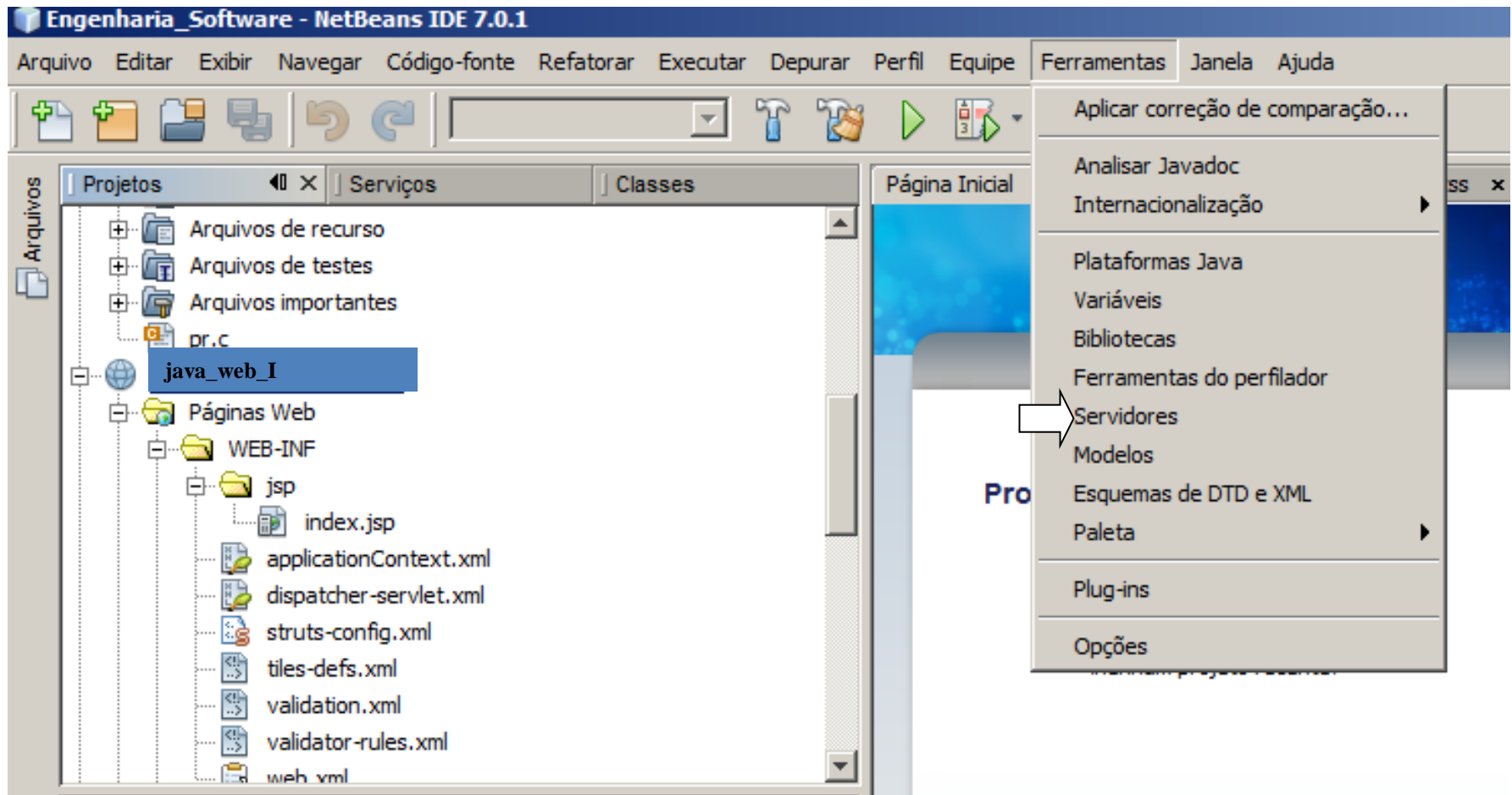
*Configurações de conexão*





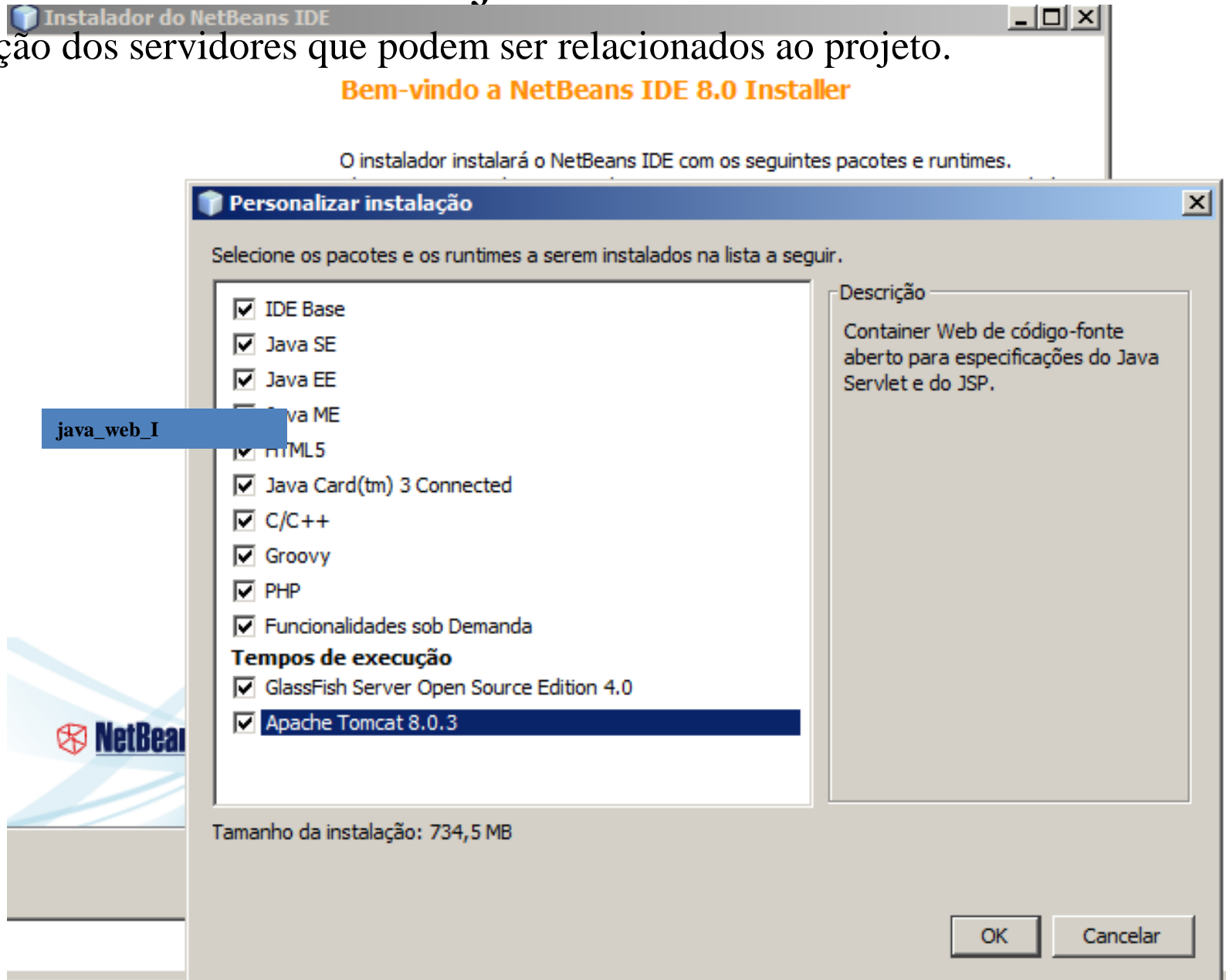
# Projeto Java Web

Verificação dos servidores que podem ser relacionados ao projeto.



# Projeto Java Web

Verificação dos servidores que podem ser relacionados ao projeto.



# Projeto Java Web

Conexão com banco de dados *MySQL*.

**Assistente de Nova Conexão**

**Personalizar Conexão**

Nome do Driver: MySQL (Driver Connector/J) em MySQL (Connector/J driver)

Host: localhost Porta: 3306

Banco de dados: mysql

Nome do Usuário: root

Senha:

☐ Lembrar senha

Propriedades da Conexão Testar Conexão

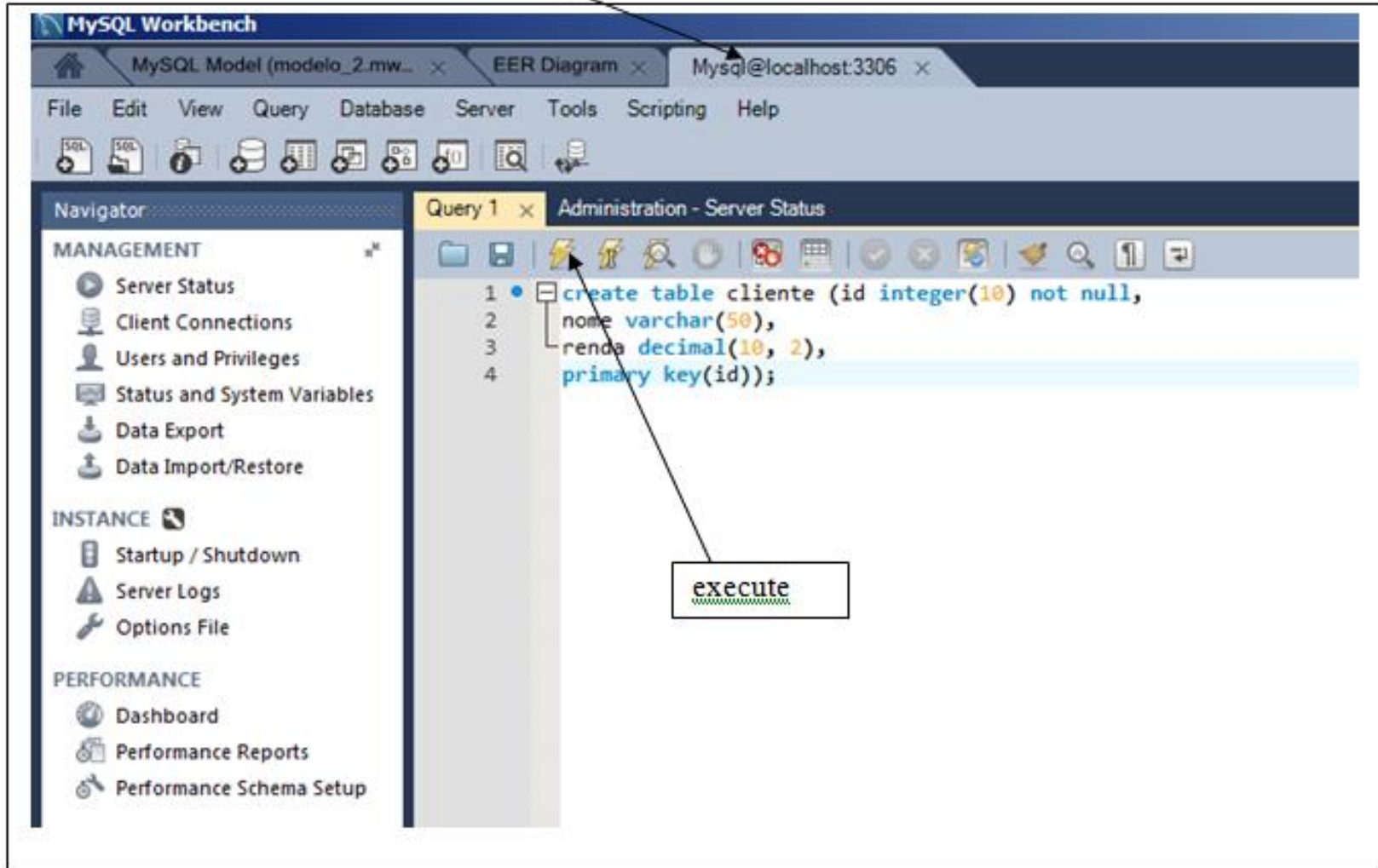
JDBC URL: jdbc:mysql://localhost:3306/mysql?zeroDateTimeBehavior=convertToNull

< Voltar Próximo > Finalizar Cancelar Ajuda

# Projeto Java Web

Edição da tabela no banco de dados *MySQL*.

tela principal do Query Browser



# Projeto Java Web

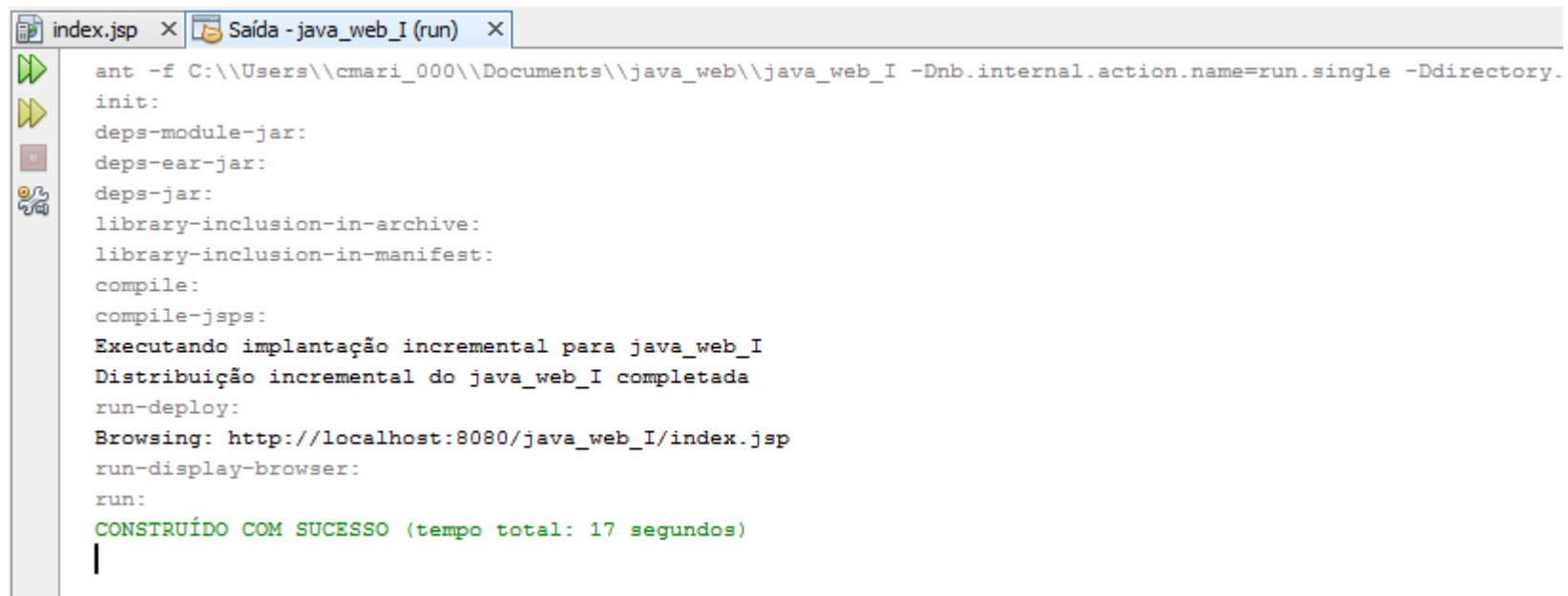
Configuração da conexão, usuário, senha e nome do banco de dados no *MySQL*.

The screenshot shows the 'Manage Server Connections' dialog box. On the left, a tree view shows 'MySQL Connections' and 'Local instance MySQL56'. The main area has tabs for 'Connection', 'Remote Management', and 'System Profile'. The 'Connection' tab is active, showing 'Connection Name: Sem nome' and 'Connection Method: Standard (TCP/IP)'. Below this are sub-tabs for 'Parameters', 'SSL', and 'Advanced'. The 'Parameters' sub-tab is active, showing fields for 'Hostname' (localhost), 'Port' (3306), 'Username' (mauricio), 'Password' (Store in Vault ...), and 'Default Schema' (bd007). Each field has a description to its right. At the bottom of the dialog, there is a separate box containing the text 'senha = brunoa'.

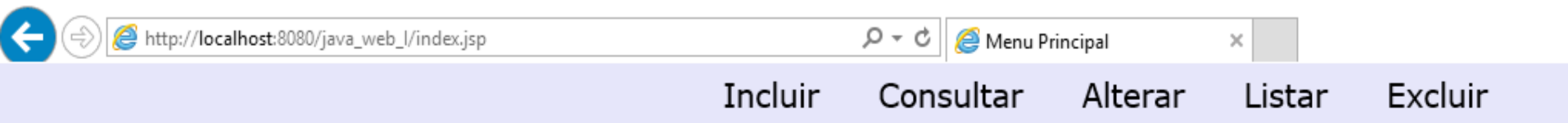
Field	Value	Description
Connection Name	Sem nome	
Connection Method	Standard (TCP/IP)	Method to use to connect to the RDBMS
Hostname	localhost	Name or IP address of the server host. - TCP/IP port.
Port	3306	
Username	mauricio	Name of the user to connect with.
Password	Store in Vault ...	The user's password. Will be requested later if it's not set.
Default Schema	bd007	The schema to use as default schema. Leave blank to select it later.





senha = brunoa

# Execução do arquivo index.jsp



```
index.jsp x Saída - java_web_I (run) x
>> ant -f C:\\Users\\cmari_000\\Documents\\java_web\\java_web_I -Dnb.internal.action.name=run.single -Ddirectory.
>> init:
>> deps-module-jar:
>> deps-ear-jar:
>> deps-jar:
>> library-inclusion-in-archive:
>> library-inclusion-in-manifest:
>> compile:
>> compile-jsp:
>> Executando implantação incremental para java_web_I
>> Distribuição incremental do java_web_I completada
>> run-deploy:
>> Browsing: http://localhost:8080/java_web_I/index.jsp
>> run-display-browser:
>> run:
>> CONSTRUÍDO COM SUCESSO (tempo total: 17 segundos)
```



← →  http://localhost:8080/java\_web\_I/index.jsp    Menu Principal x

Incluir Consultar Alterar Listar Excluir

# Projeto Java Web

Lista do 1º bimestre: **exercício**

Construir o projeto Java Web e demonstrar o funcionamento através da apresentação dos *prints* do acesso ao banco de dados através dos formulários. Mostrar também a tabela no banco de dados *MySQL* mostrando dados inseridos e/ou modificados.

## Referências Bibliográficas:

- Teruel, Evandro Carlos. *Web Total - desenvolva sites com tecnologias de uso livre: prático e avançado*. Editora Érica. São Paulo, 2009.
- Servlets & JSP
- Bryan Basham, Kathy Sierra e Bert Bates
- Ed. O'Reilly / Alta Books -