



www.hightechcursos.com.br

Material de Apoio

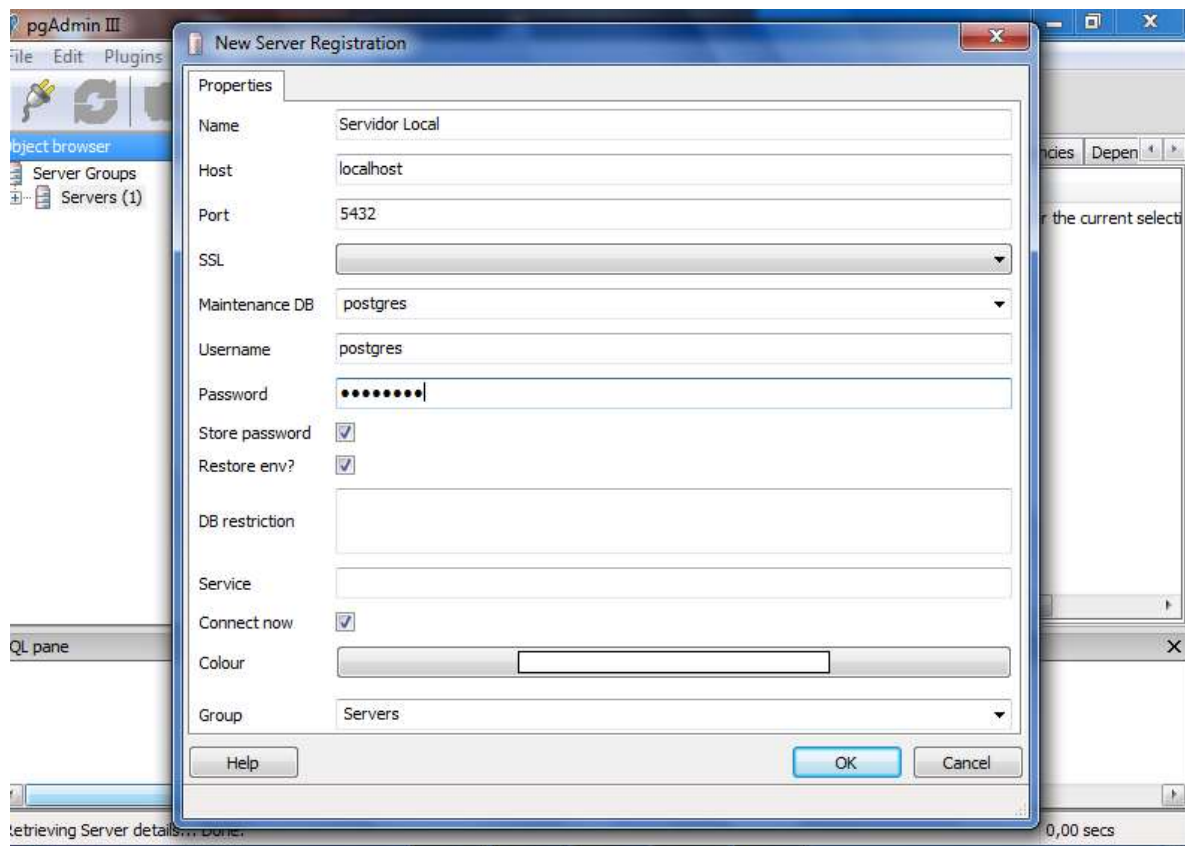
CURSO de JAVA WEB FUNDAMENTOS

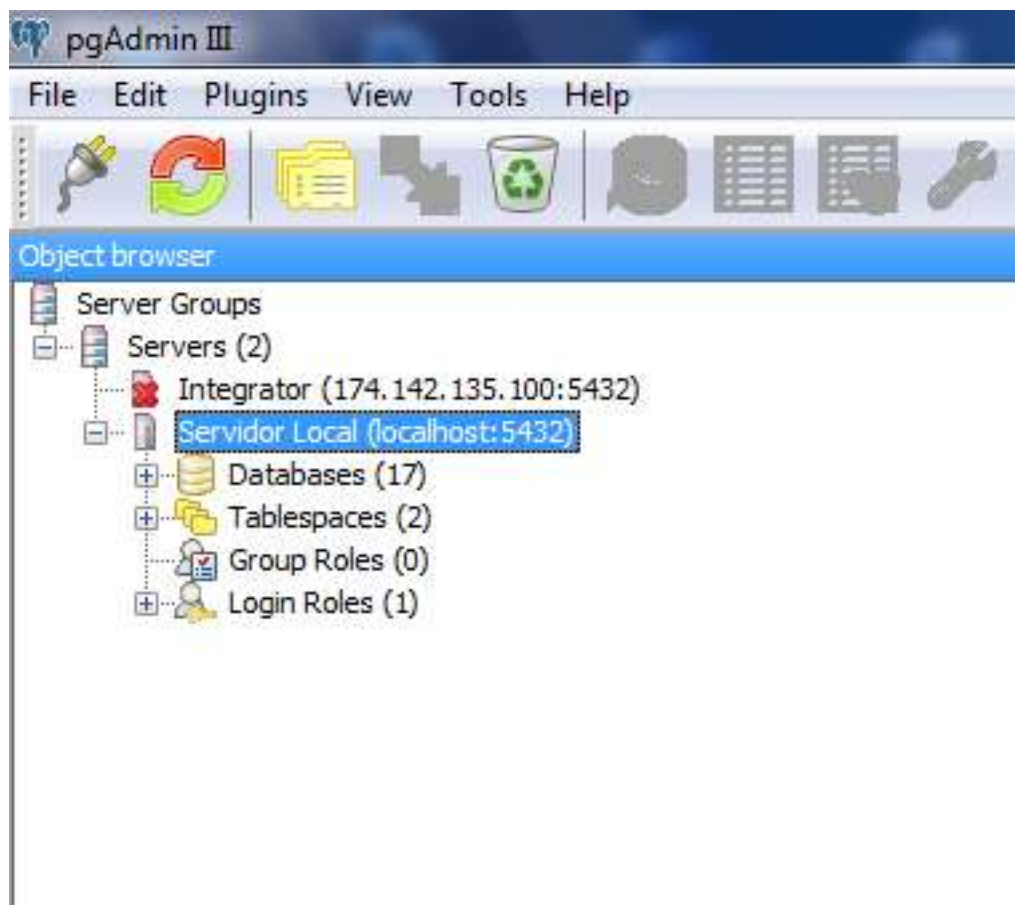
www.hightechcursos.com.br - contato@hightechcursos.com.br (67) 3387-2941

(CJWEB1)

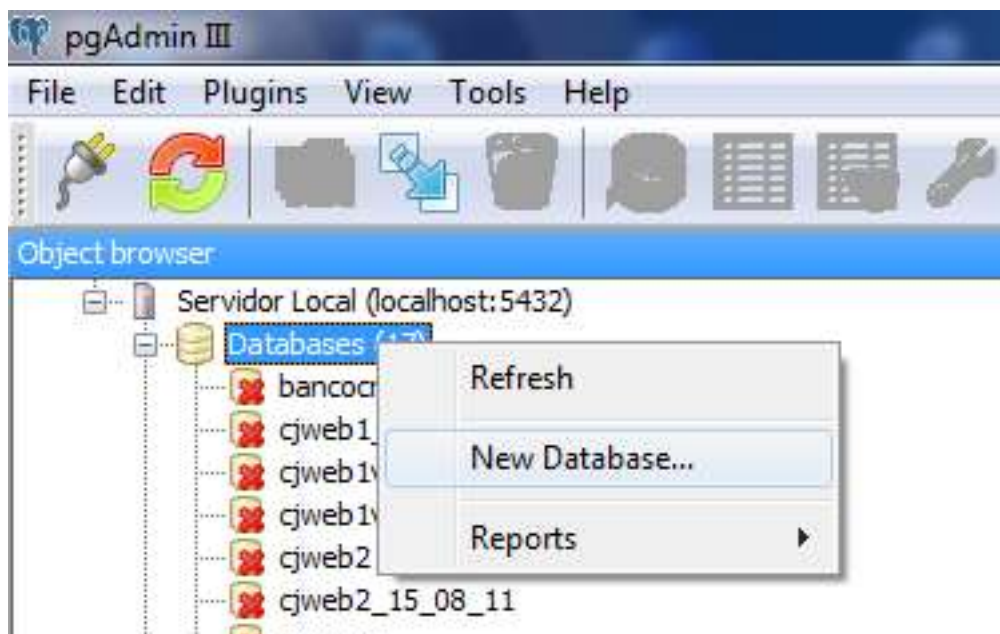
Banco de Dados

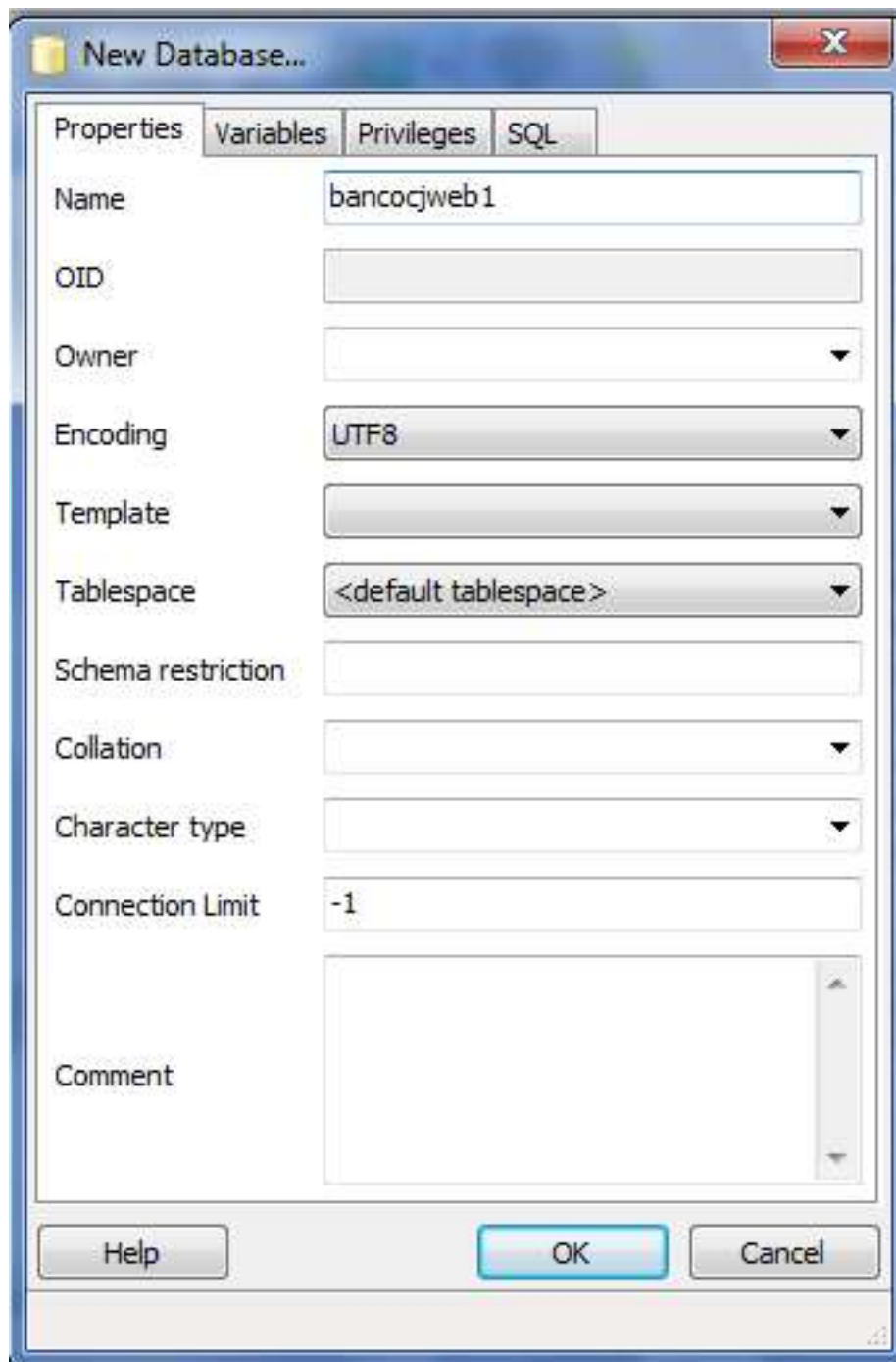
Criando Conexão com Servidor PostgreSQL Local



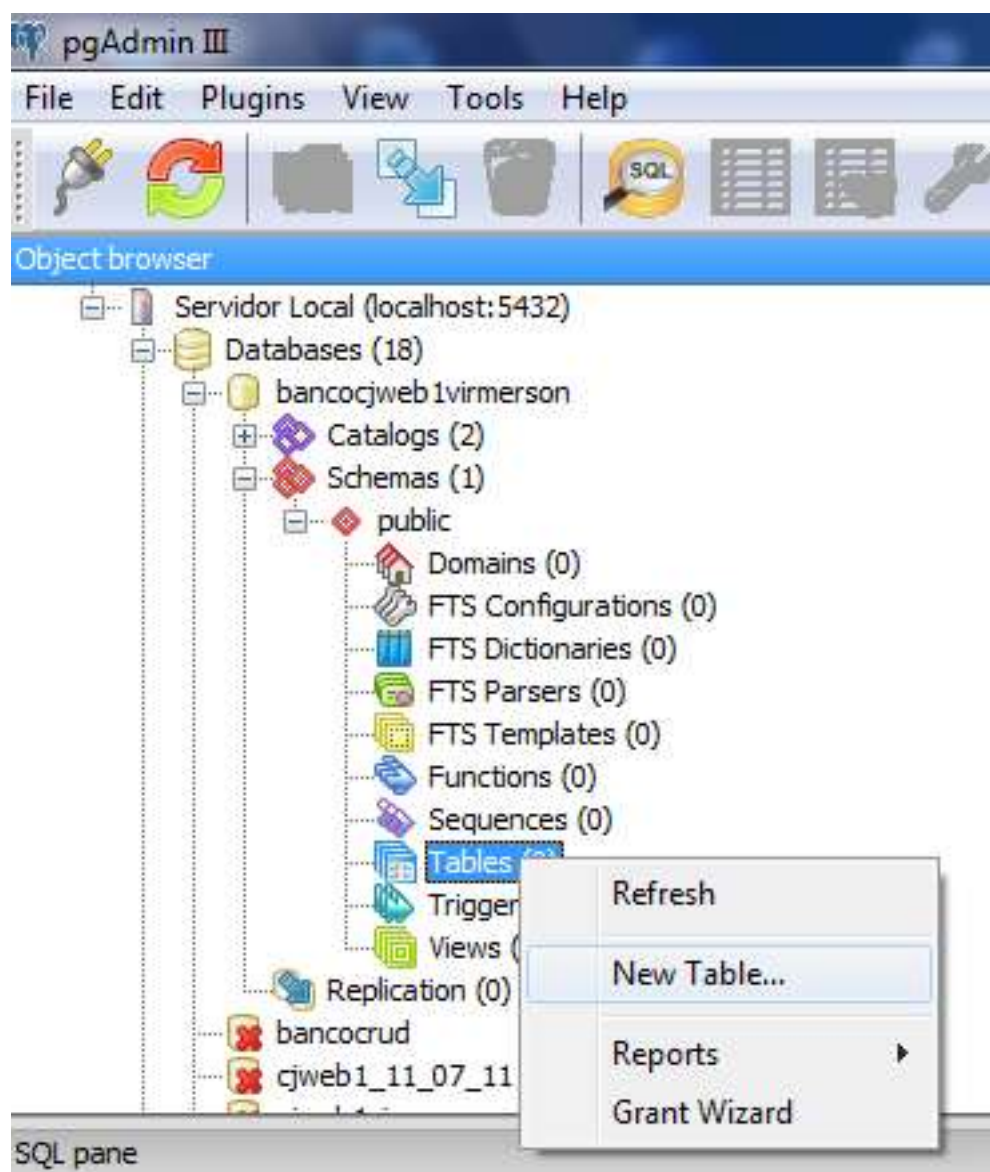


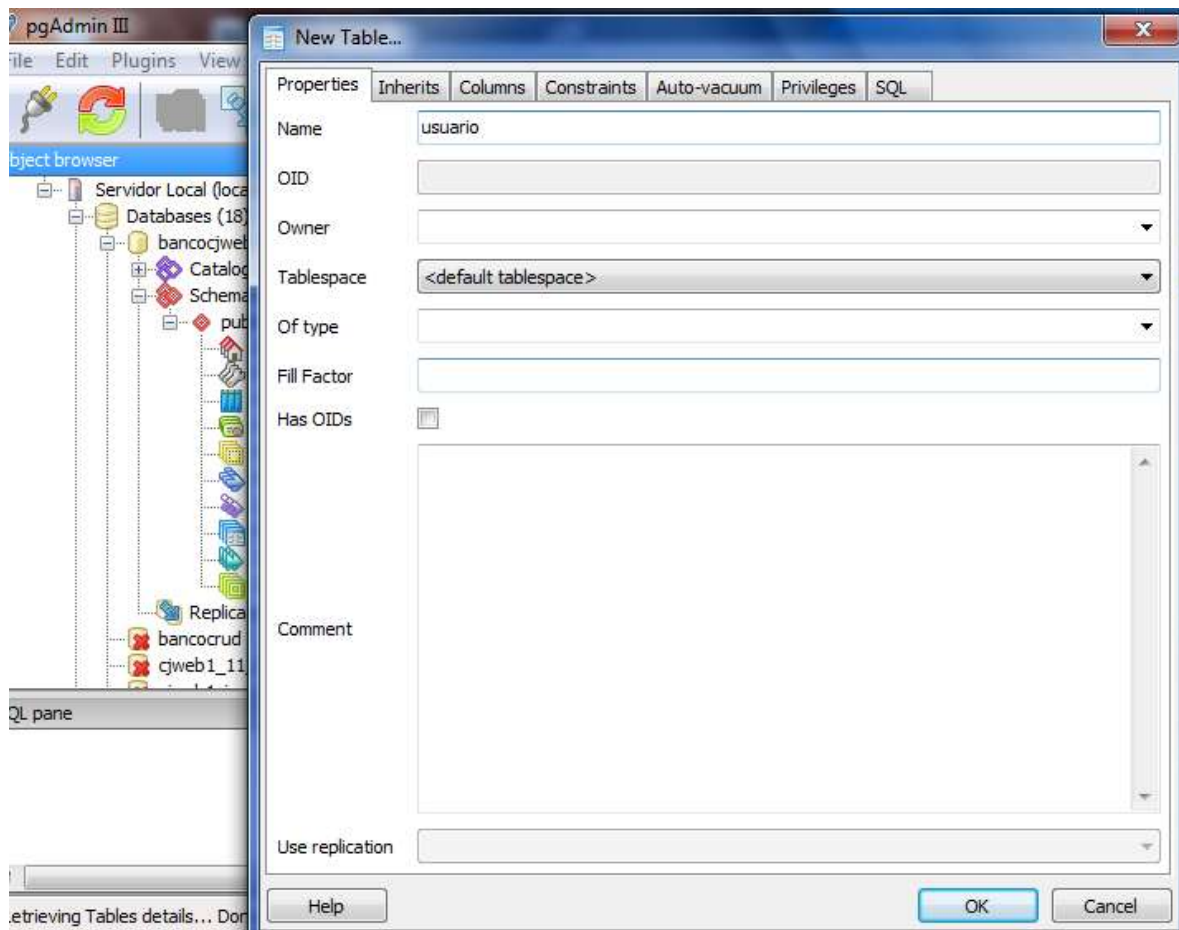
Criando novo Banco de Dados



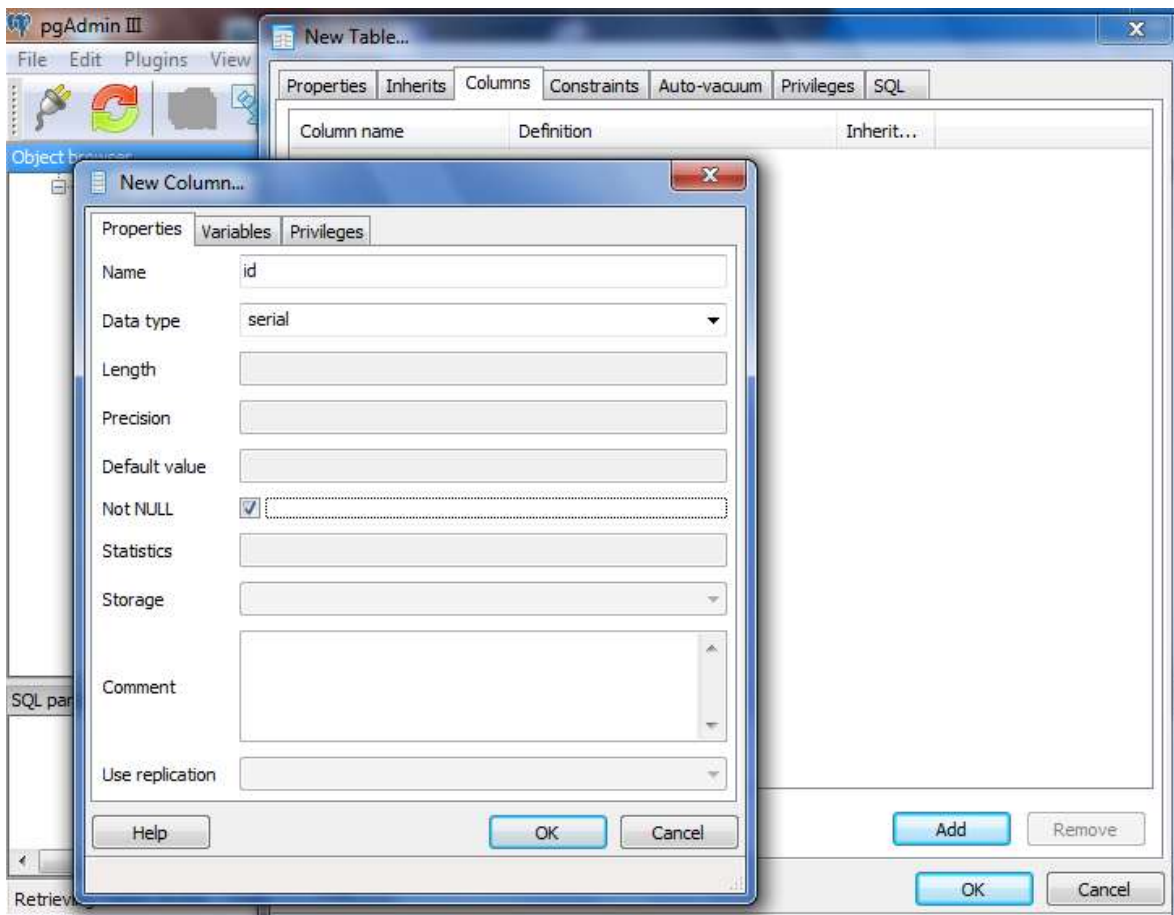


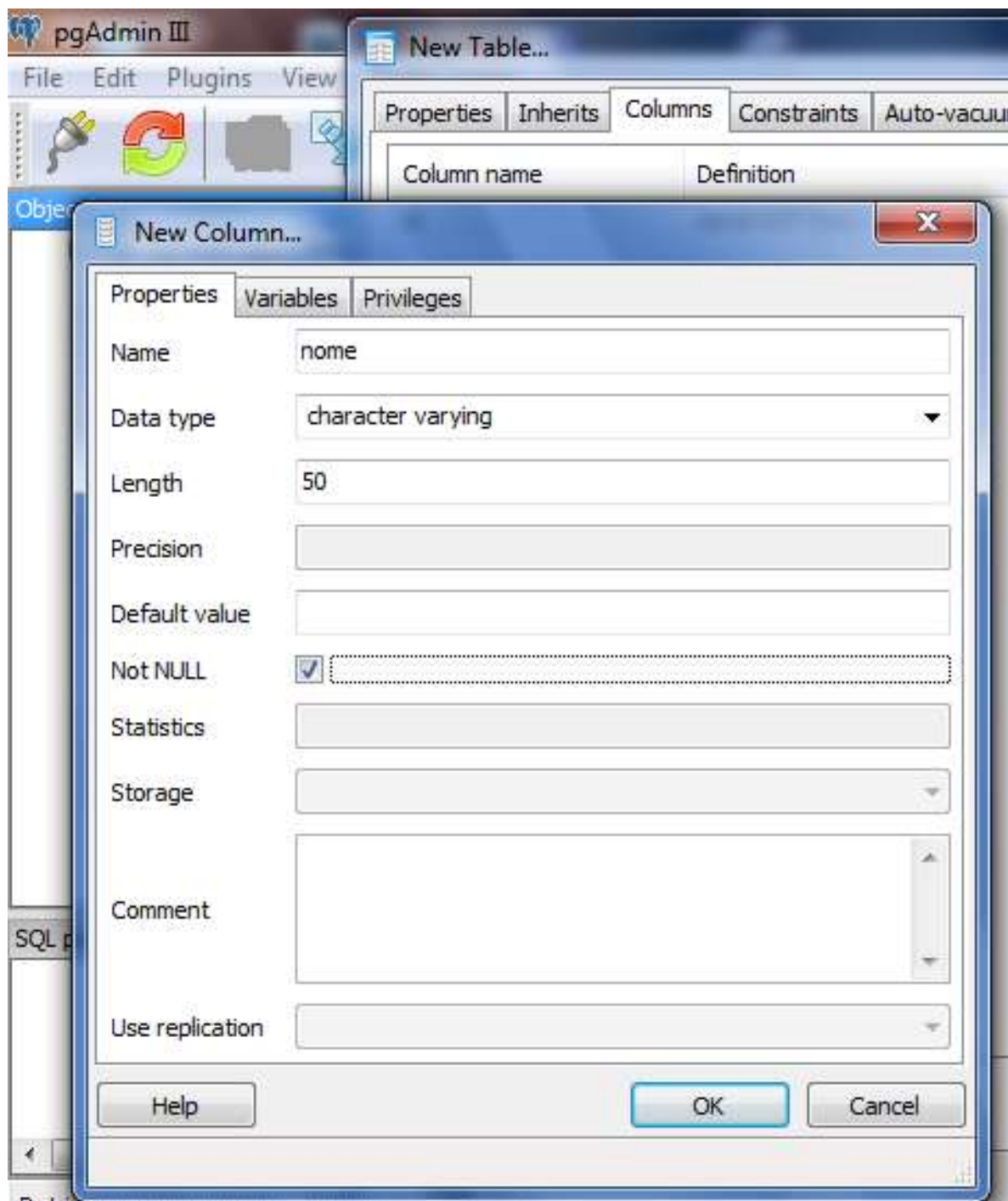
Criando nova Tabela "usuario" no Banco de Dados

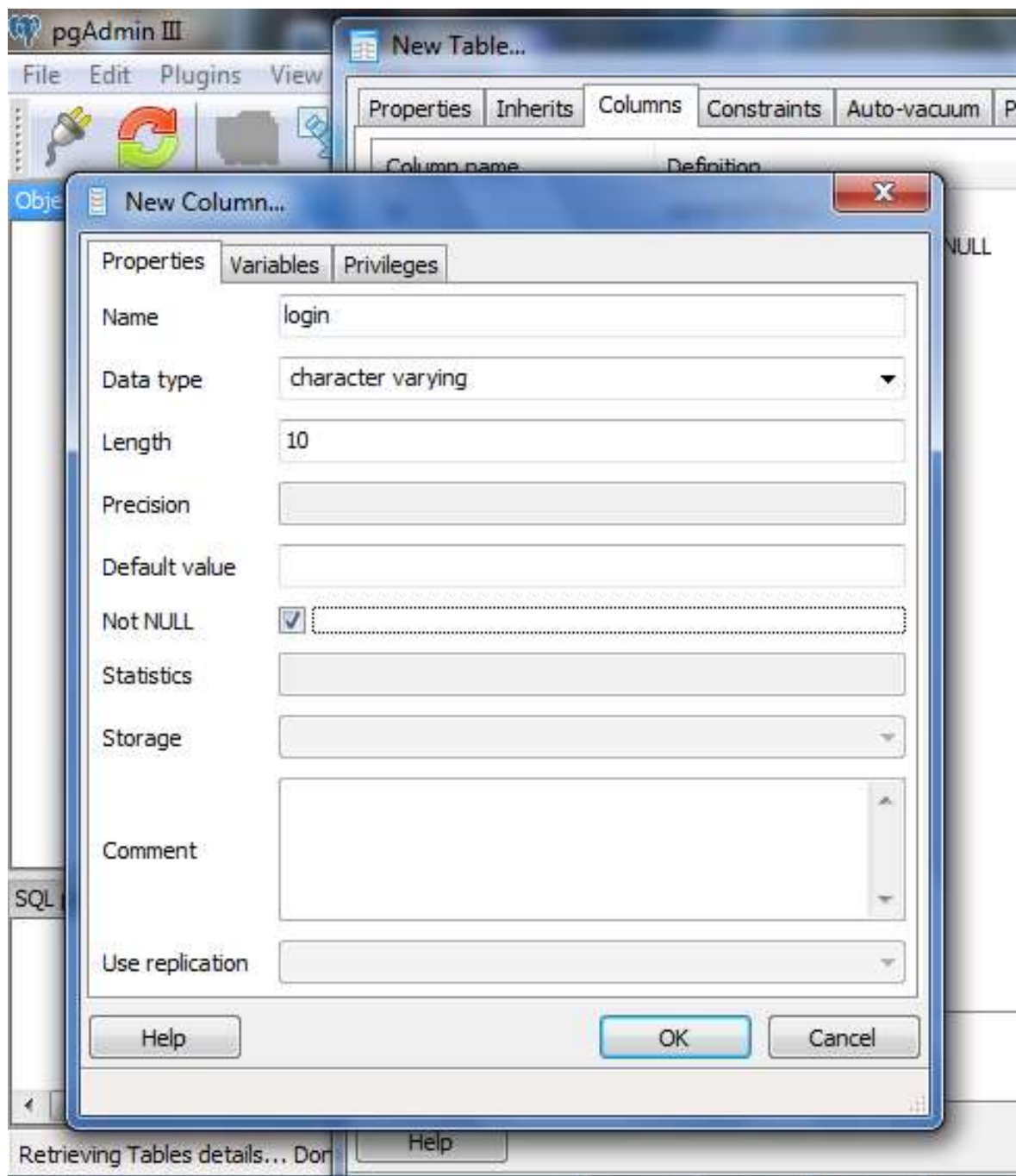


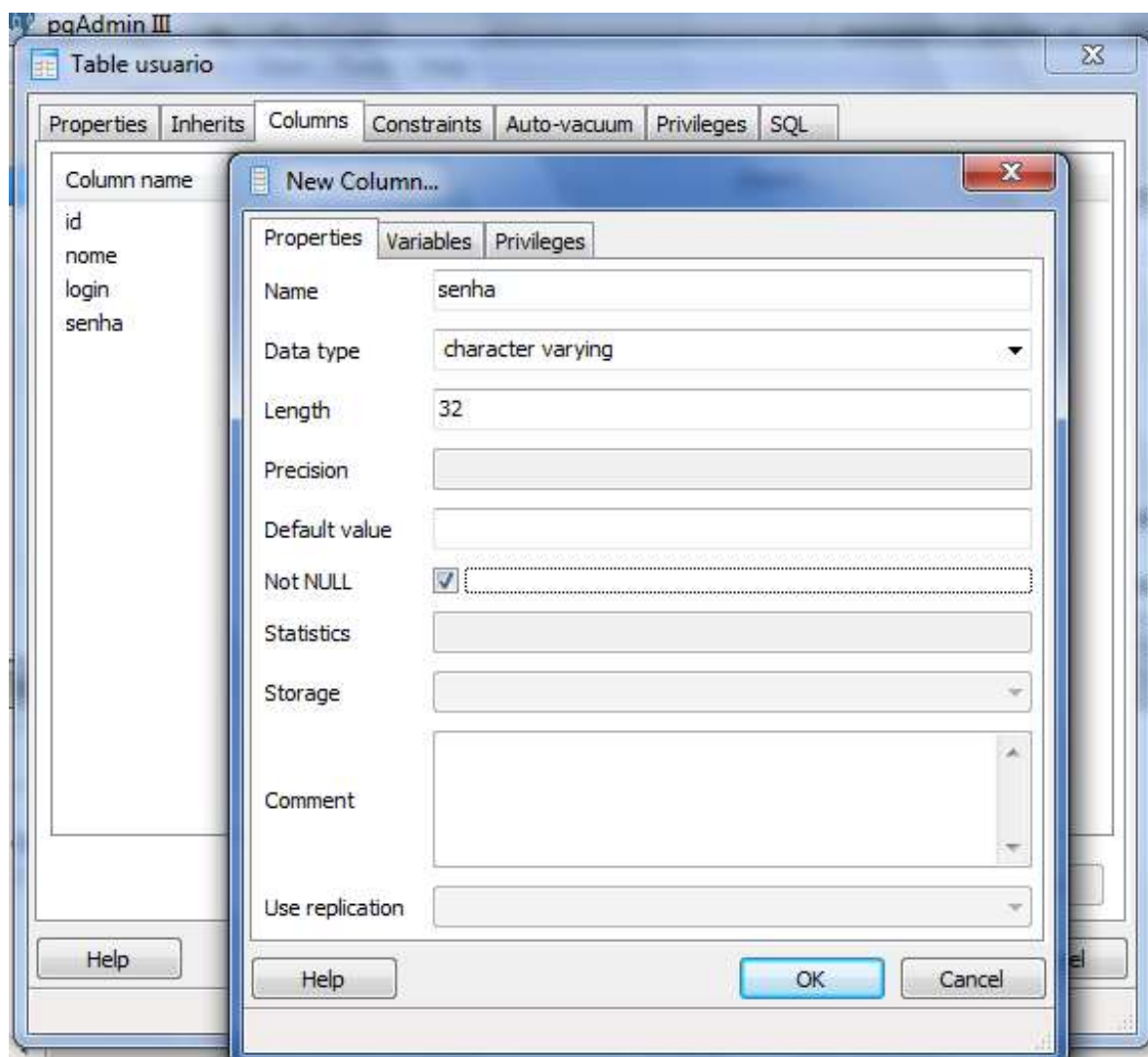


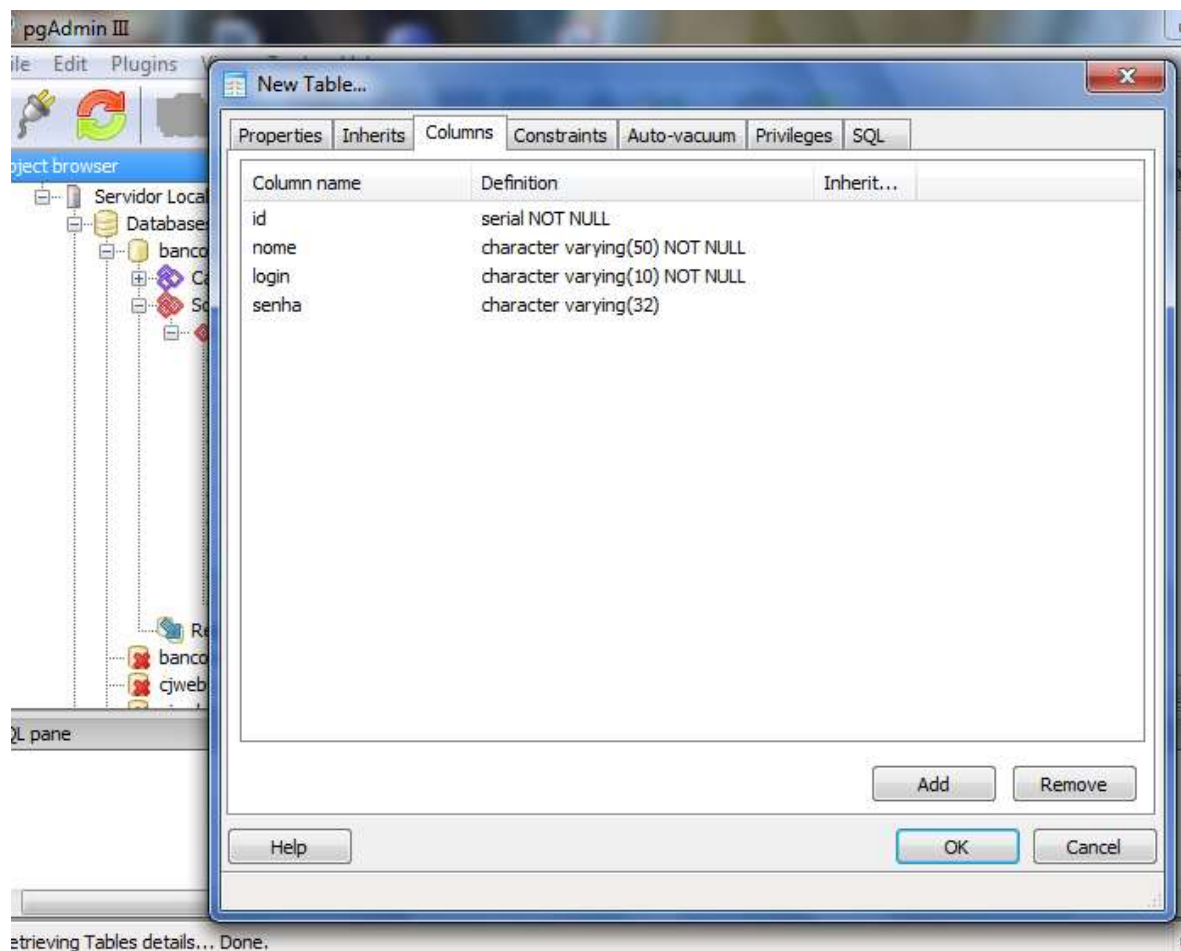
Criando colunas id Serial, nome, login e senha



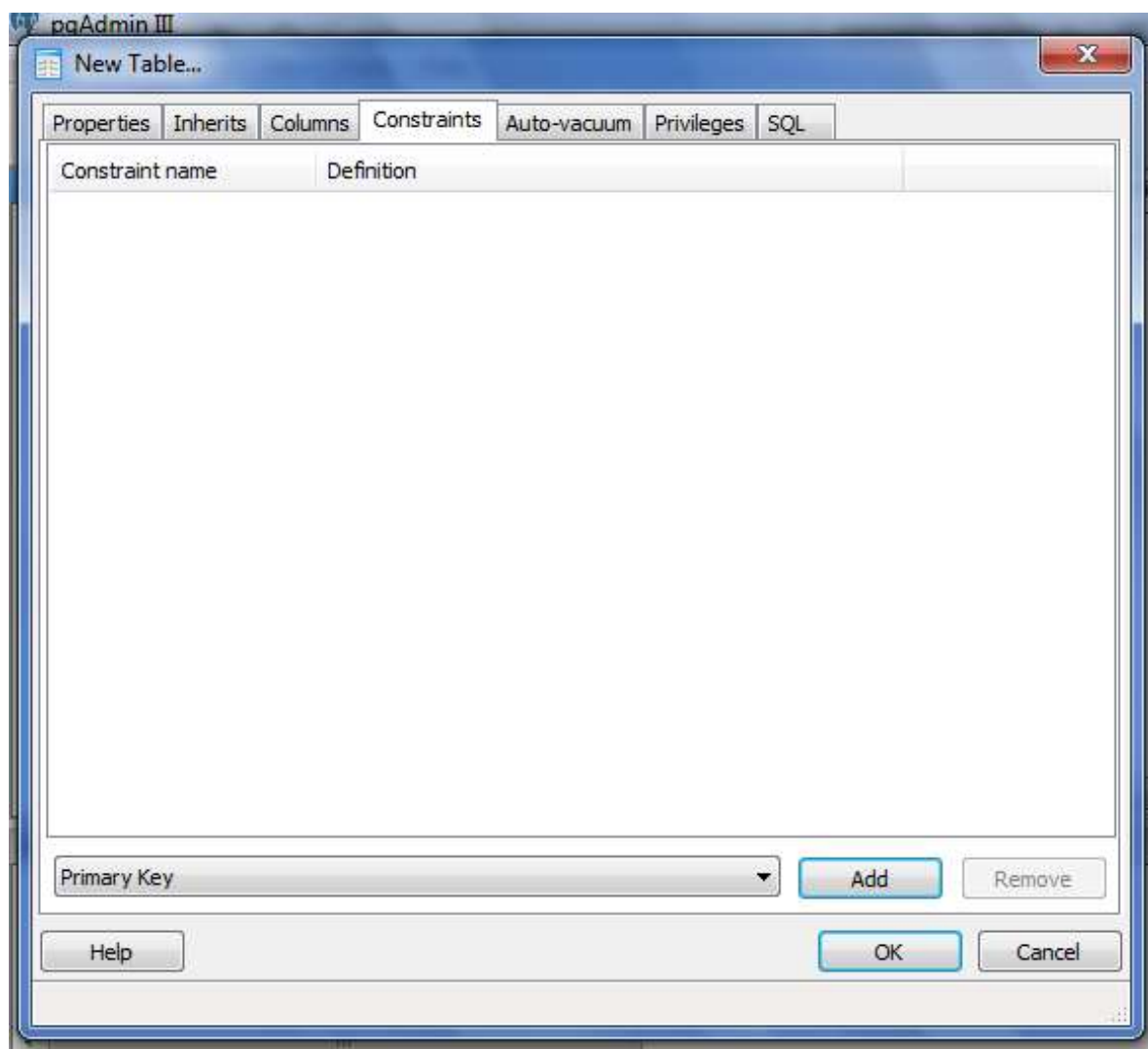


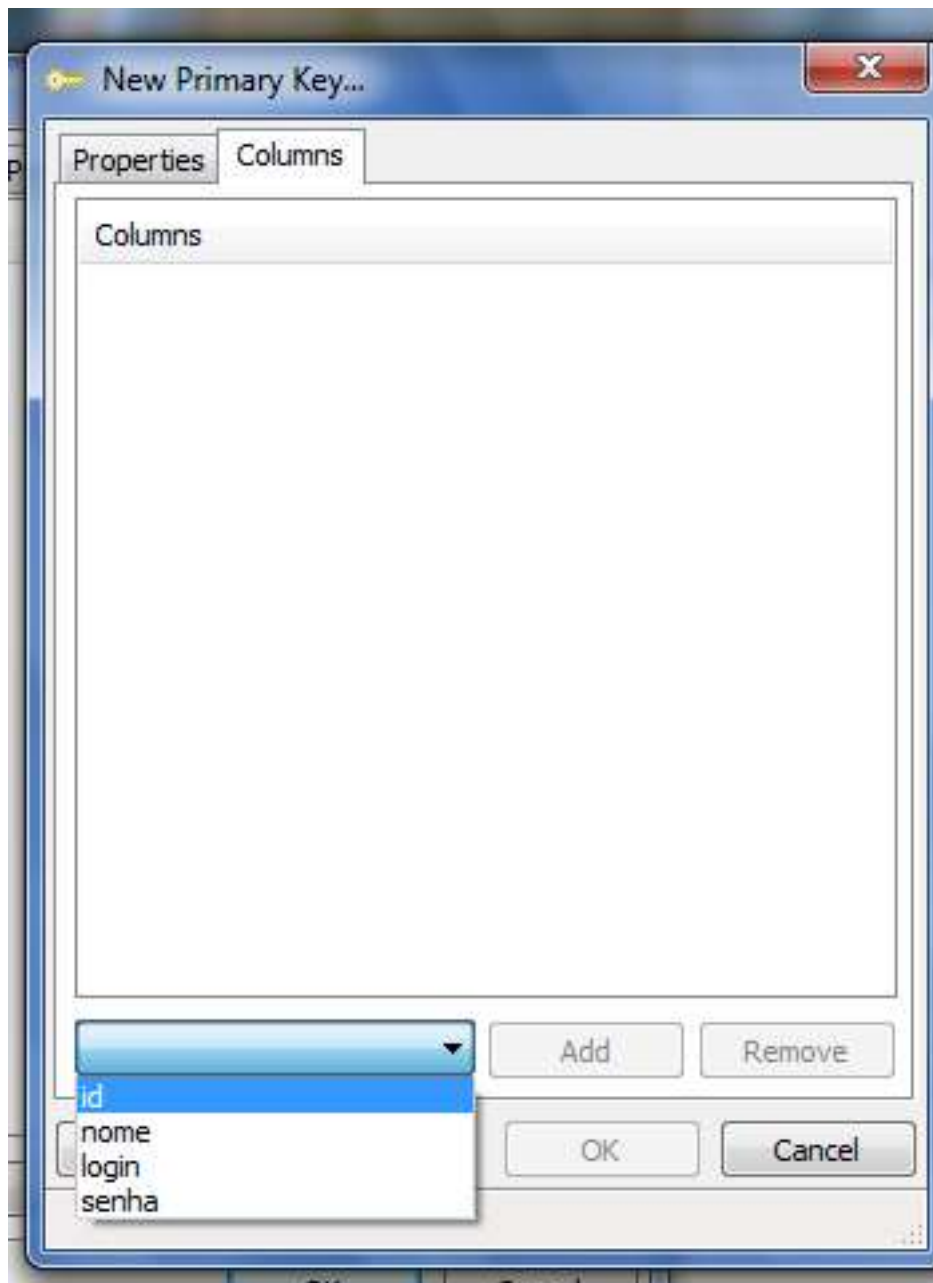






Adicionando Constraint – Chave Primária para a coluna id





Executando comandos SQL

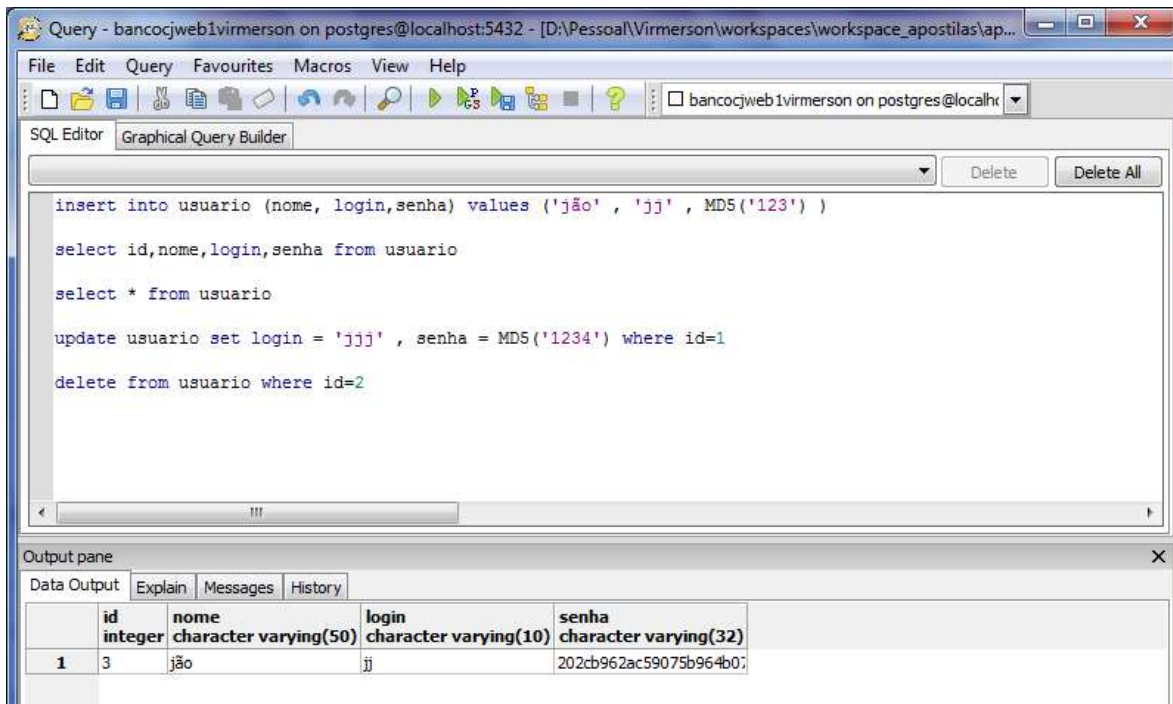
```
insert into usuario (nome, login, senha) values ('jão' , 'jj' , MD5('123') )
```

```
select id,nome,login,senha from usuario
```

```
select * from usuario
```

```
update usuario set login = 'jjj' , senha = MD5('1234') where id=1
```

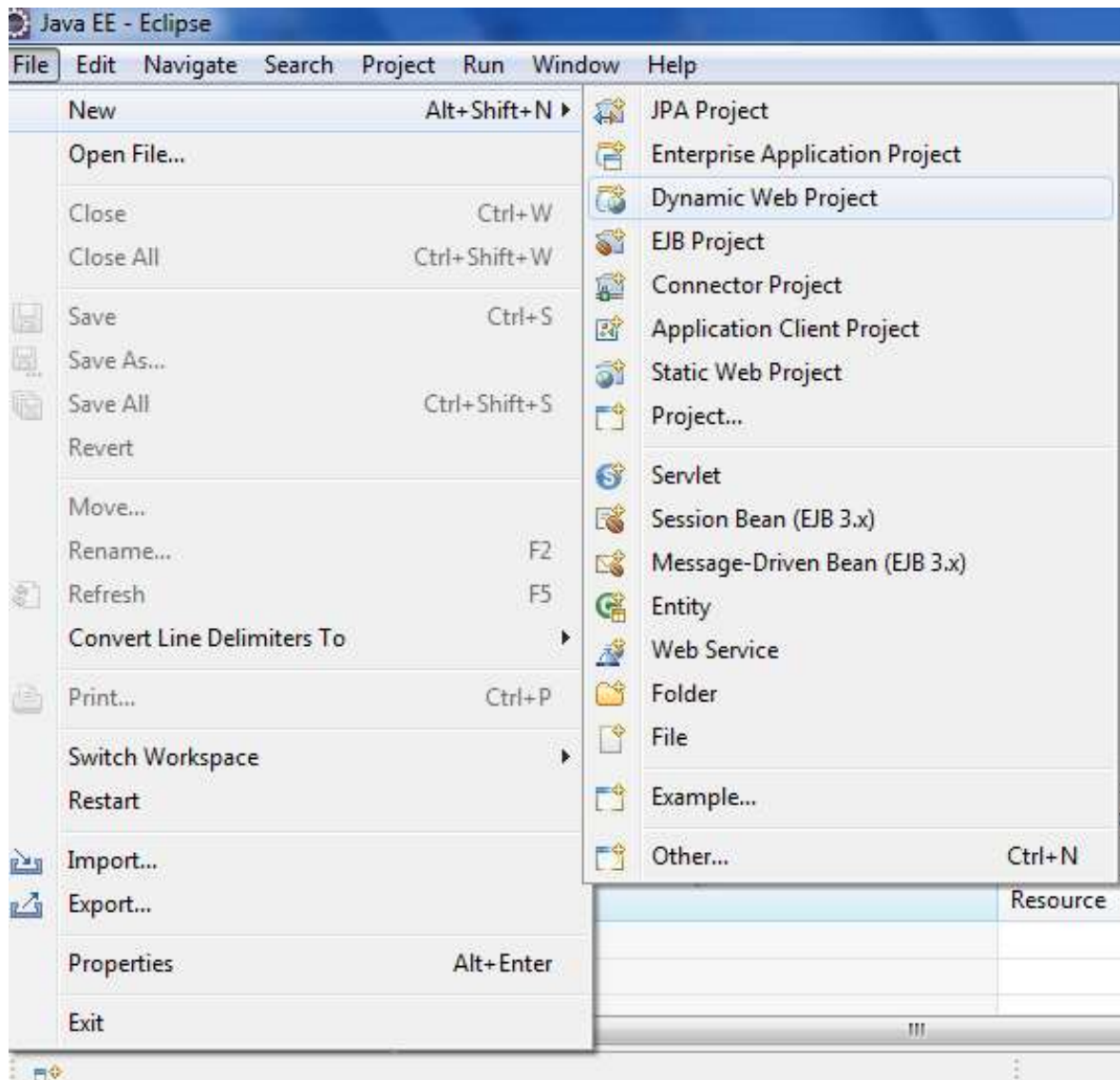
delete from usuario where id=2

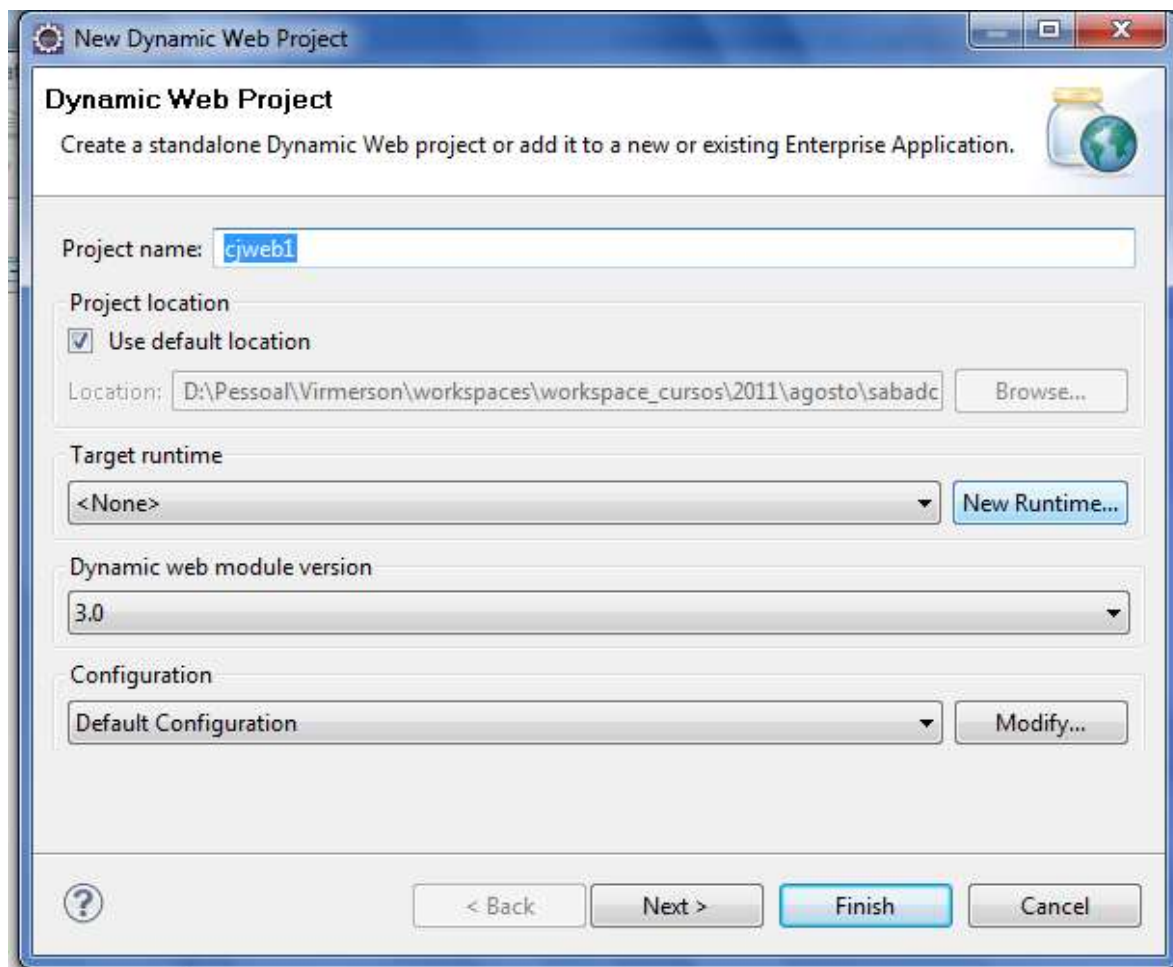


Projeto Java - Dynamic Web Project – J2EE

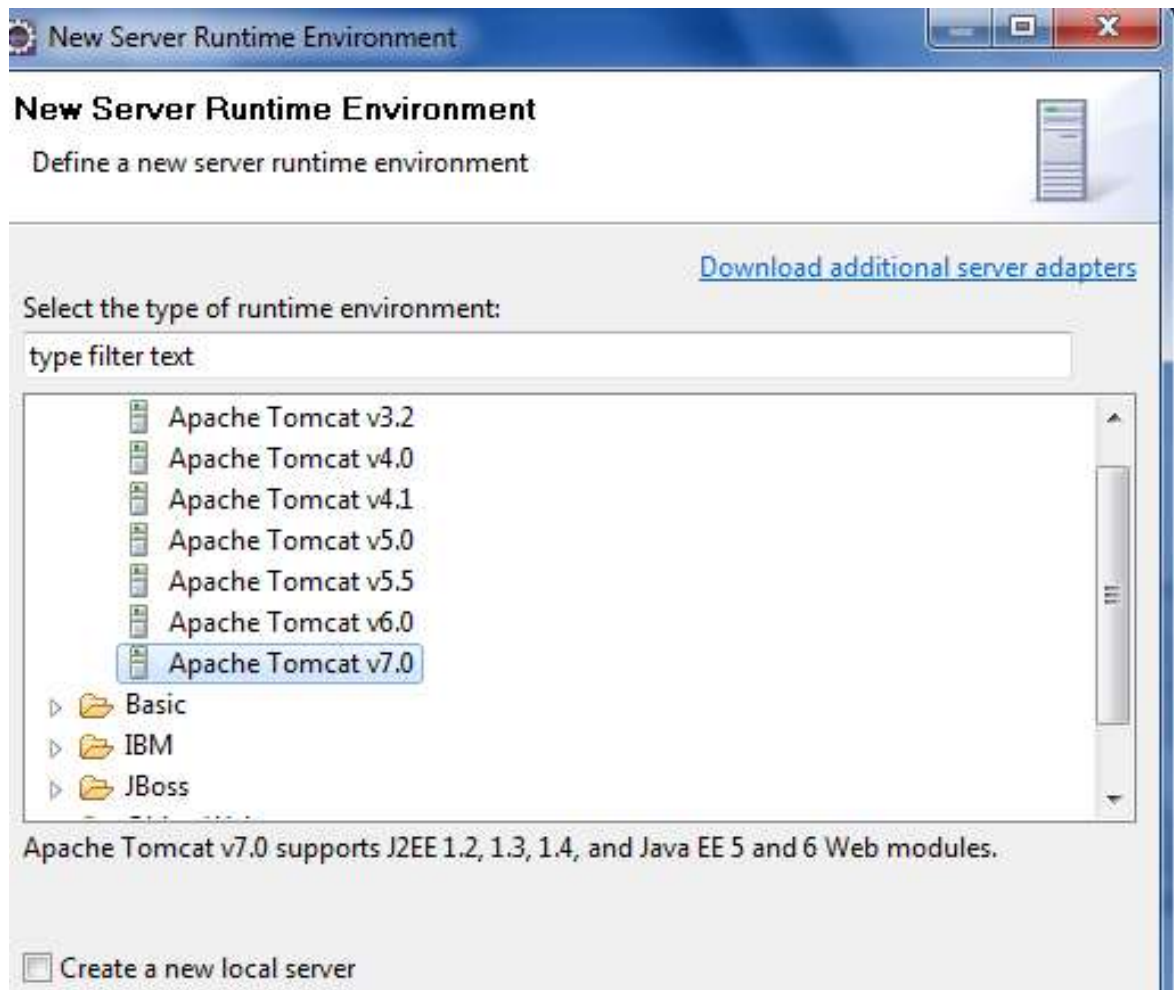
Utilizando o Eclipse na perspectiva Java EE



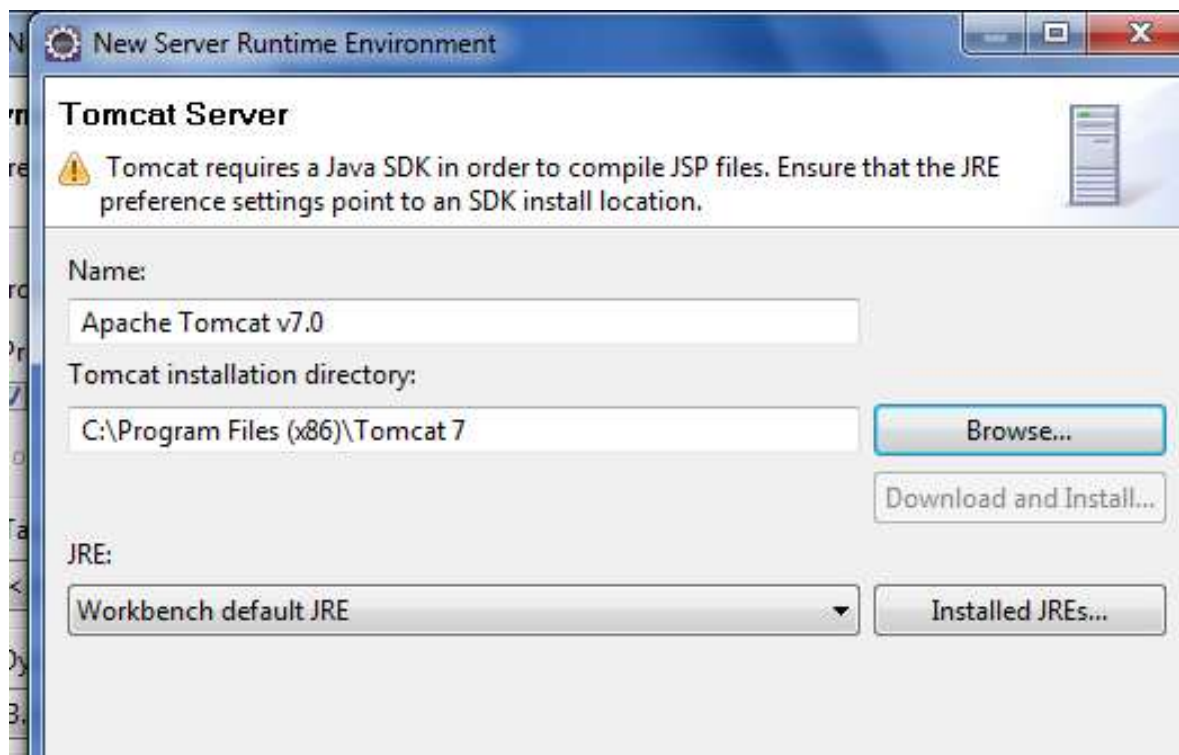




Definindo o TOMCAT como sendo nosso Servlet Container Web

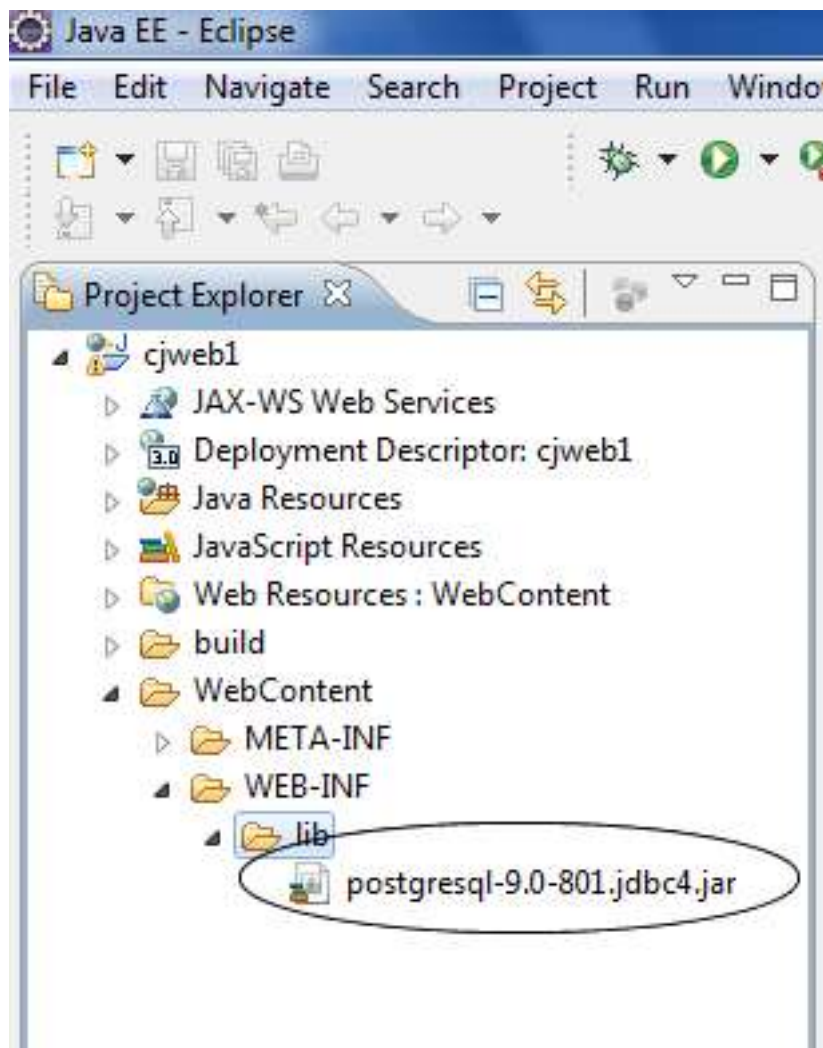


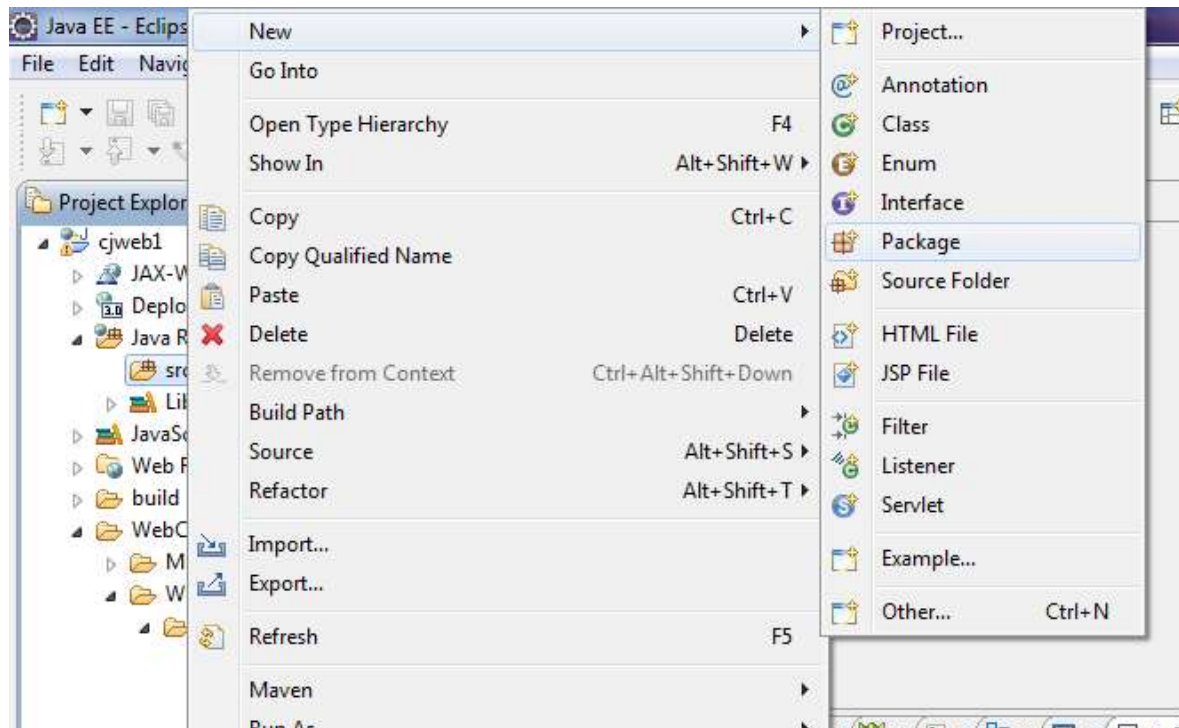
Selecionando o Endereço de instalação do Tomcat

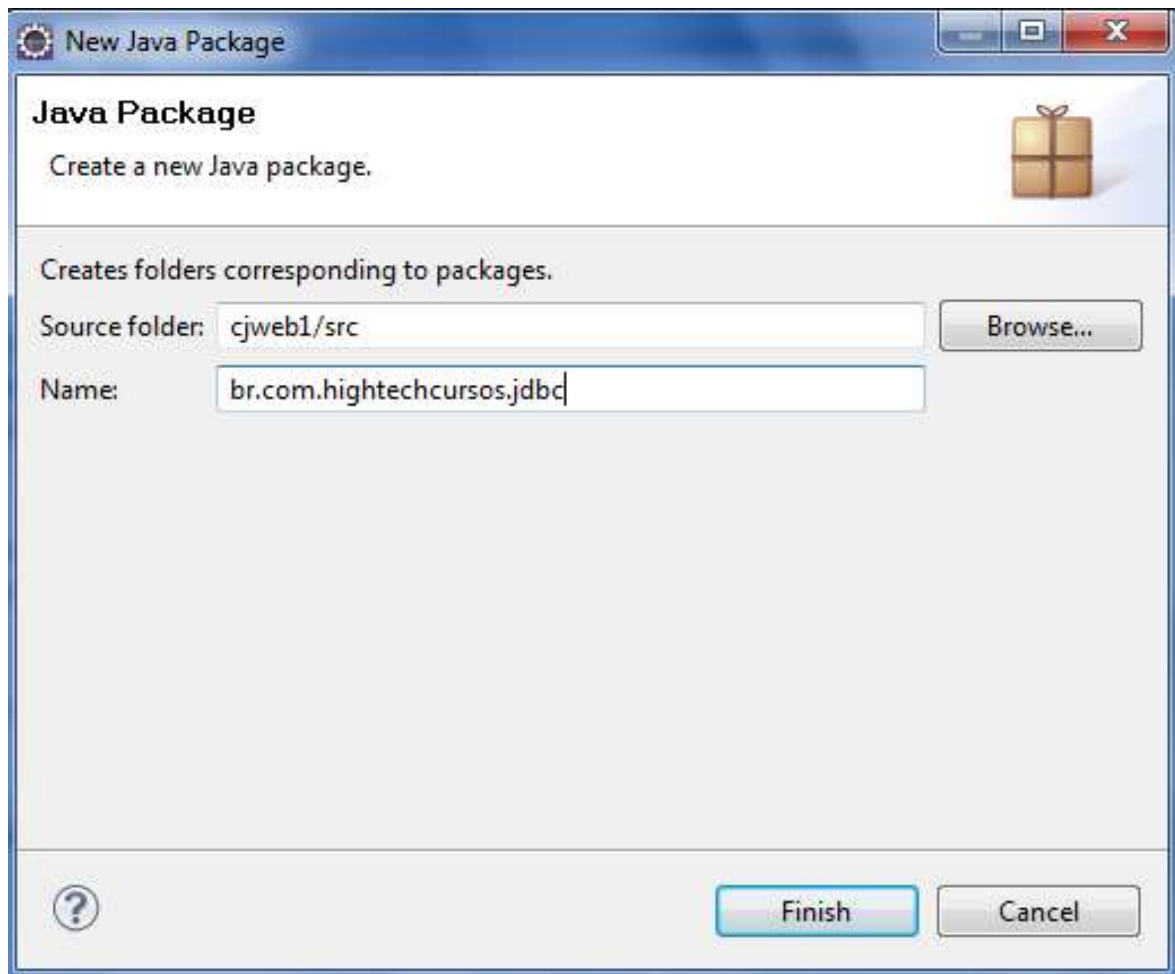


Adicionando a Lib do Postgresql contendo do DRIVER JDBC .

Será necessário adicionarmos as classes do Driver em nosso ClassPath.



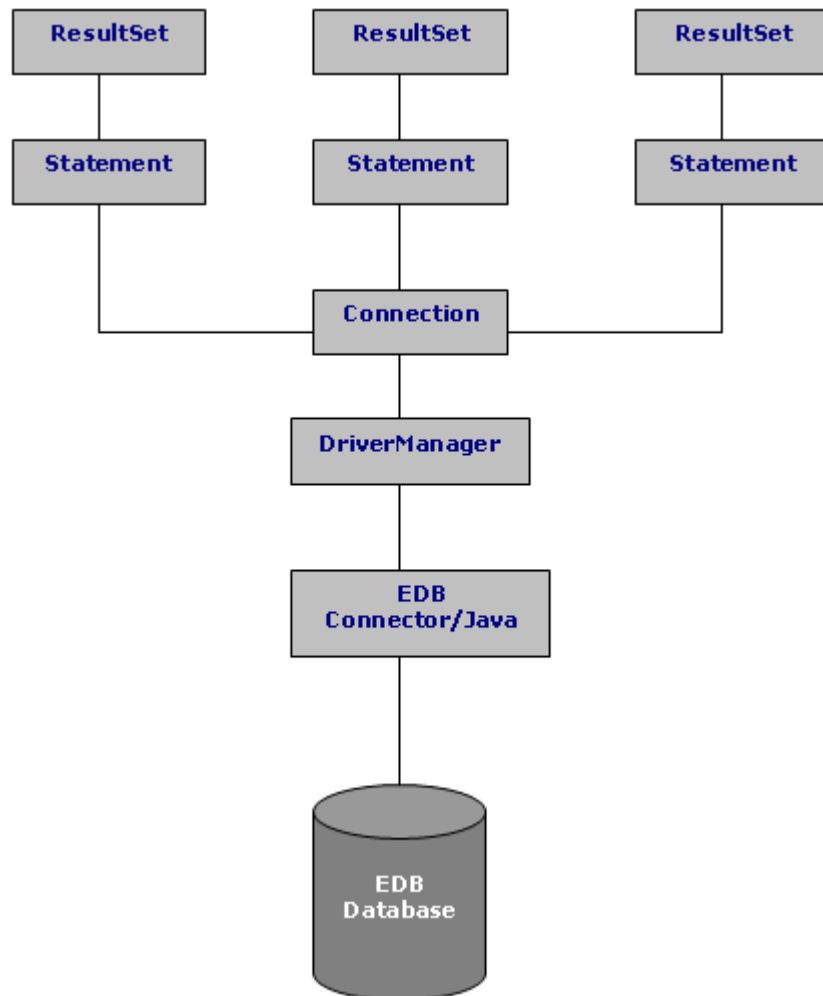




JDBC

Tipos Básicos do JDBC.

Utilizaremos essas classes e interfaces para realizar a comunicação com banco de dados.



Criando classe de Conexão com o Banco PostgreSQL

```
package br.com.hightechcursos.jdbc;  
  
import java.sql.Connection;  
import java.sql.DriverManager;
```

```

import java.sql.SQLException;

public class Conexao {

    /**
     * Método que conecta no banco postgresql
     * @return Objeto de Conexao do tipo Connection
     */
    public static Connection getConnection () {
        Connection con = null;

        try {
            //Obtendo ponteiro ao objeto de Conexao
            con =
DriverManager.getConnection("jdbc:postgresql://localhost:5432/cjweb1Jão",
"postgres", "");

            } catch (SQLException e) {
                System.out.println("Não conectou no banco:" + e.getMessage()
);
            }
            return con;
        }
    }
}

```

Criando a classe de teste no pacote de teste

```

package br.com.hightechcursos.teste;

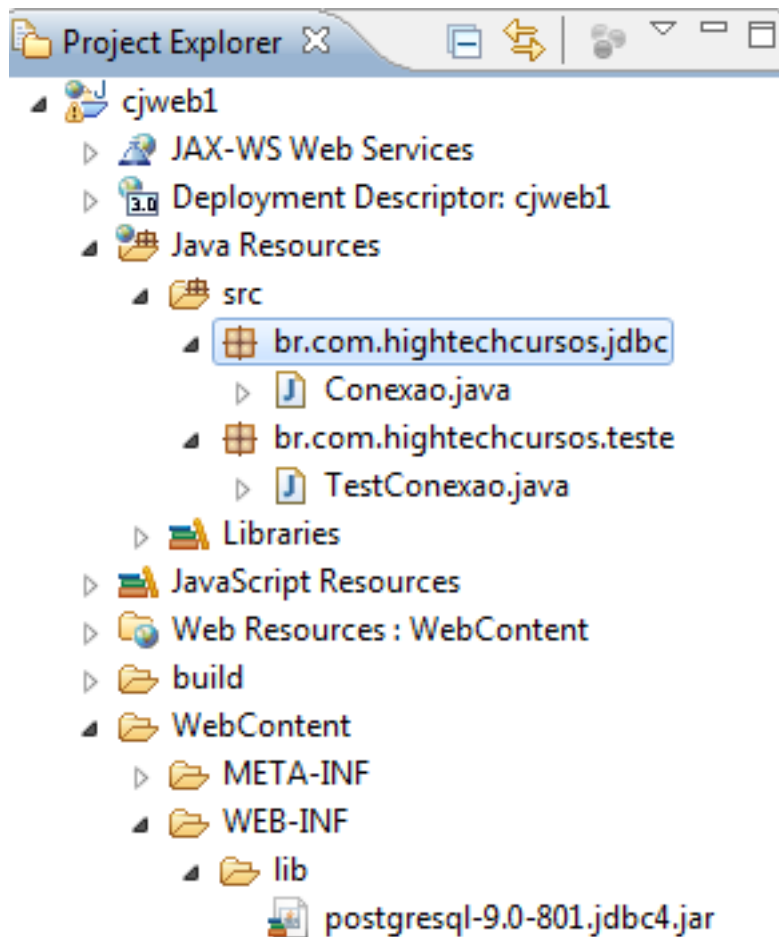
import br.com.hightechcursos.jdbc.Conexao;

public class TestConexao {

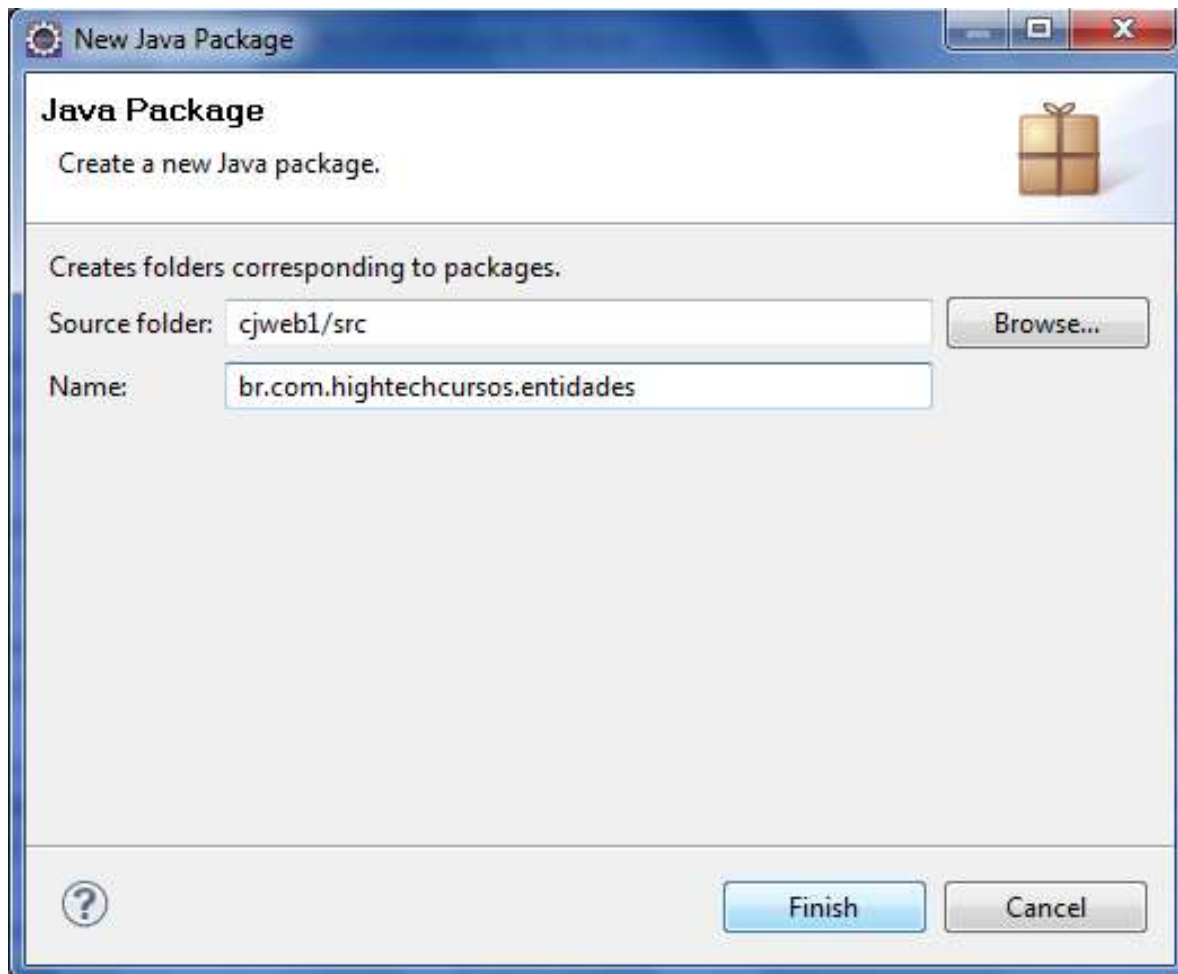
    public static void main(String[] args) {
        //Testando a classe Conexao
        Conexao.getConnection();
    }
}

```

Estrutura do projeto



Criando pacote de entidades



Criando a Classe Usuario

Esta classe será usada em todo o nosso sistema.

Mapeamento com o Banco de DADOS.

```
package br.com.hightechcursos.entidades;

public class Usuario {
    private Integer id;
    private String nome;
    private String login;
    private String senha;
    // Getters and Setters
}
```

TABELA => CLASSE
COLUNA => PROPRIEDADE
REGISTRO => OBJETO

```
public class Usuario {
    private Integer id;
    private String nome;
    private String login;
    private String senha;
    // Getters and Setters
}
```

id [PK] serial	nome character vai	login character vai	senha character varying(32)
1	JÃO	jj	202cb962ac59075b964b07152d234b70

Usuario usu1

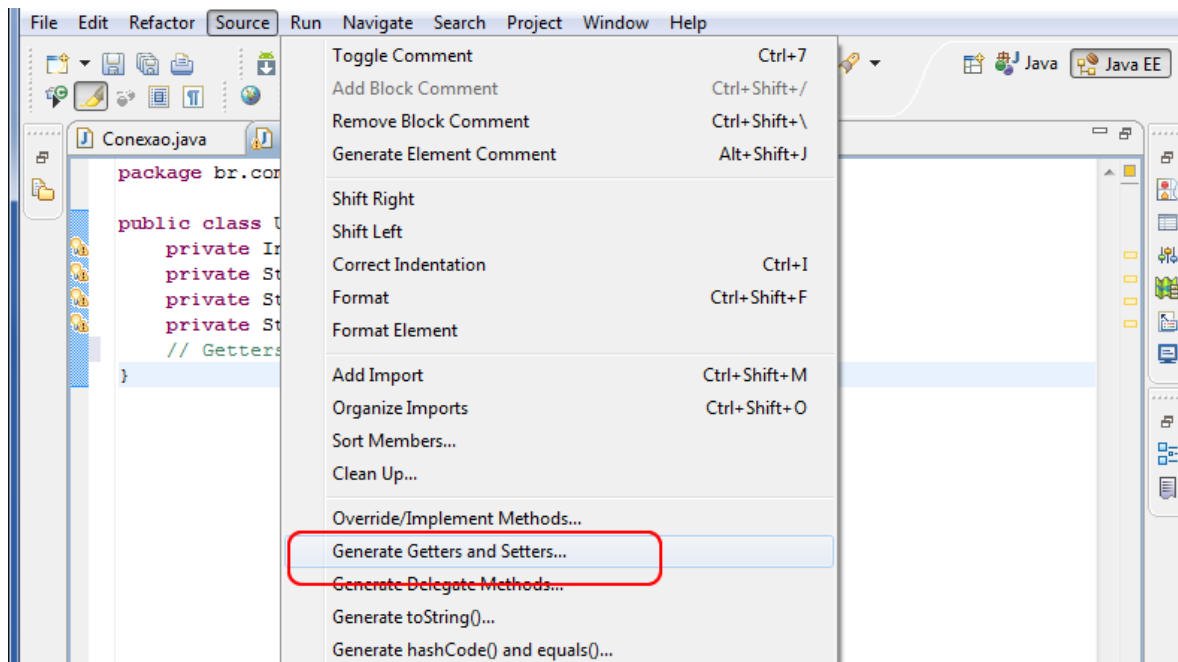
id:1
nome:JÃO
login:jjj
senha:202...

new Usuario()

É importante saber que para cada tabela no banco nós criaremos uma classe de entidade.

Gerando Getters and Setters

Vamos criar os métodos encapsuladores utilizando o gerado automático do eclipse.



Criando a Classe UsuarioDAO para Cadastrar

```
package br.com.hightechcursos.jdbc;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import br.com.hightechcursos.entidades.Usuario;

public class UsuarioDAO {
    private Connection con= Conexao.getConnection();

    public void cadastrar(Usuario usu) {
        String sql = "INSERT INTO usuario (nome,login,senha) VALUES(?, ?
,md5(?)) ";
        try {
            //Preparando SQL
            PreparedStatement preparador = con.prepareStatement(sql);
            //Substituindo os parametros do SQL pelos valores do objeto
            preparador.setString(1, usu.getNome());
            preparador.setString(2, usu.getLogin());
            preparador.setString(3, usu.getSenha());
            //Executa SQL
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

```

        preparador.execute();
        //Fecha Statment
        preparador.close();
        System.out.println("Cadastrado com sucesso!");
    } catch (SQLException e) {

        System.out.println("Erro de SQL:" + e.getMessage());
    }
}

```

Classe Teste Unitário Completa de UsuarioDAO

Esta classe é muito importante, pois nela vamos testar todos os métodos da classe UsuarioDAO aplicando o conceito de teste unitário, onde testamos todas unidades(classes) do sistema.

Para Cada Método Criado devemos criar um método de teste dentro da Classe de Teste.

```

package br.com.hightechcursos.teste;

import br.com.hightechcursos.entidades.Usuario;
import br.com.hightechcursos.jdbc.UsuarioDAO;

public class TestUsuarioDAO {
    public static void main(String[] args) {
        //Instancia do objeto a ser cadastrado
        Usuario usuCad = new Usuario();
        usuCad.setNome("CARLOS");
        usuCad.setLogin("CAR");
        usuCad.setSenha("123");

        //Intancia do DAO a ser testado
        UsuarioDAO usuDao = new UsuarioDAO();
        // Invoca o método cadastrar
        usuDao.cadastrar(usuCad);
    }
}

```

Criando o Método Alterar

Este método é muito parecido com o de cadastro. Observe que temos que definir um ID para especificar o registro que deve ser alterado

```

public void alterar(Usuario usu){
    String sql = "update usuario set nome=?, login=?, senha=md5(?)
where id=?";
    try {
        PreparedStatement preparador = con.prepareStatement(sql);
        preparador.setString(1, usu.getNome());
        preparador.setString(2, usu.getLogin());
        preparador.setString(3, usu.getSenha());
        preparador.setInt(4, usu.getId());
        preparador.execute();
        preparador.close();
        System.out.println("Alterado com sucesso!");
    } catch (SQLException e) {

        System.out.println("Erro de SQL:" + e.getMessage());
    }
}

```

Testando a método Alterar

Para nosso código ficar mais organizado os testes, vamos criar um método de teste para cada método que queremos testar do DAO. Assim podemos testar cada para de nosso DAO separadamente chamando pelo método “main” e colocando em comentário aquela que não queremos executar.

```

package br.com.hightechcursos.teste;

import java.util.List;

import br.com.hightechcursos.entidades.Usuario;
import br.com.hightechcursos.jdbc.UsuarioDAO;
/**
 * Test Unitário de UsuarioDAO
 * @author HighTechCursos
 */
public class TestUsuarioDAO {

    static UsuarioDAO usuDao = new UsuarioDAO();

    public static void main(String[] args) {
        //deveCadastrar();
        //deveAlterar();
    }
    private static void deveCadastrar() {

```

```

        Usuario usuCad = new Usuario();
        usuCad.setNome("CARLOS");
        usuCad.setLogin("CAR");
        usuCad.setSenha("123");
        usuDao.cadastrar(usuCad);
    }
    private static void deveAlterar() {
        Usuario usuAlt = new Usuario();
        usuAlt.setId(1);
        usuAlt.setNome("Jão Bento");
        usuAlt.setLogin("Jão");
        usuAlt.setSenha("Jão123");
        usuDao.alterar(usuAlt);
    }
}

```

Criando Método Excluir

Neste método apagamos o registro do usuário utilizando apenas o id como parâmetro

```

public void excluir(Usuario usu){
    String sql = "delete from usuario where id=?";
    try {
        PreparedStatement preparador = con.prepareStatement(sql);

        preparador.setInt(1, usu.getId());
        preparador.execute();
        preparador.close();
        System.out.println("Excluído com sucesso!");
    } catch (SQLException e) {

        System.out.println("Erro de SQL:" + e.getMessage());
    }
}

```

Testando excluir

Adicionado o novo teste na classe *TestUsuarioDAO*.

```

private static void deveExcluir() {
    Usuario usuExc = new Usuario();
    usuExc.setId(1);
    usuDao.excluir(usuExc);
}

```

```
}
```

Criando Método Salvar

Método que encapsula os métodos cadastrar e alterar.

O critério de decisão é o id do usuário.

Toda a vez que o usuário tiver um "id" definido como diferente de null e maior que zero então o realiza alteração senão realiza cadastro.

```
public void salvar(Usuario usu) {  
    if(usu.getId() != null && usu.getId() > 0) {  
        alterar(usu);  
    } else {  
        cadastrar(usu);  
    }  
}
```

Testando o método

Teste este método definindo um id diferente de zero que exista no banco de dados e também teste não definindo um id.

Quando o objeto tiver com id definido o método deve realizar alteração senão cadastro.

```
private static void deveSalvar() {  
    Usuario usuSalvar = new Usuario();  
    //usuSalvar.setId(3);  
    usuSalvar.setNome("CARLOS");  
    usuSalvar.setLogin("CAR");  
    usuSalvar.setSenha("123");  
    usuDao.salvar(usuSalvar);  
}
```

Criando um Método buscaPorId

Este método será necessário toda vez que desejarmos carregar um usuário pelo id para realizar uma alteração.

```
public Usuario buscaPorId(Integer id) {  
  
    //Objeto de retorno do método  
    Usuario usuRetorno = null;
```



```

String sql = "select * from usuario where id=?";
try {

    PreparedStatement preparador = con.prepareStatement(sql);

    preparador.setInt(1, id);

    //Retorno da consulta em Resultset
    ResultSet resultado = preparador.executeQuery();

    //Se tem registro
    if(resultado.next()){
        //instancia o objeto Usuario
        usuRetorno = new Usuario();
        usuRetorno.setId(resultado.getInt("id"));
        usuRetorno.setNome(resultado.getString("nome"));
        usuRetorno.setLogin(resultado.getString("login"));
        usuRetorno.setSenha(resultado.getString("senha"));
    }

    System.out.println("Encontrado com sucesso!");
} catch (SQLException e) {

    System.out.println("Erro de SQL:" + e.getMessage());
}

return usuRetorno;
}

```

Testando o método

Adicionado o novo teste na classe *TestUsuarioDAO*.

```

private static void deveBuscarPorId() {
    Usuario usuRetorno = usuDao.buscaPorId(5);
    if(usuRetorno!=null)
        System.out.println(
            usuRetorno.getId() + " " +
            usuRetorno.getNome() + " " +
            usuRetorno.getLogin() + " " +
            usuRetorno.getSenha());
}

```

Criando Método Busca Todos

```
public List<Usuario> buscaTodos() {
    //Objeto de retorno do método
    List<Usuario> listaRetorno = new ArrayList<Usuario>();
    String sql = "select * from usuario order by id";
    try {
        PreparedStatement preparador = con.prepareStatement(sql);

        //Retorno da consulta em Resultset
        ResultSet resultado = preparador.executeQuery();
        //Navegada nos registros
        while(resultado.next()){
            //instancia o objeto Usuario
            Usuario usu = new Usuario();
            //Carga de dados no usuário
            usu.setId(resultado.getInt("id"));
            usu.setNome(resultado.getString("nome"));
            usu.setLogin(resultado.getString("login"));
            usu.setSenha(resultado.getString("senha"));

            //adiciona na lista
            listaRetorno.add(usu);
        }
        System.out.println("Busca com sucesso!");
    } catch (SQLException e) {
        System.out.println("Erro de SQL:" + e.getMessage());
    }
    return listaRetorno;
}
```

Testando o método

Adicionado o novo teste na classe *TestUsuarioDAO*.

```
private static void deveBuscarTodos() {

    List<Usuario> lista = usuDao.buscaTodos();
    for (Usuario u : lista) {

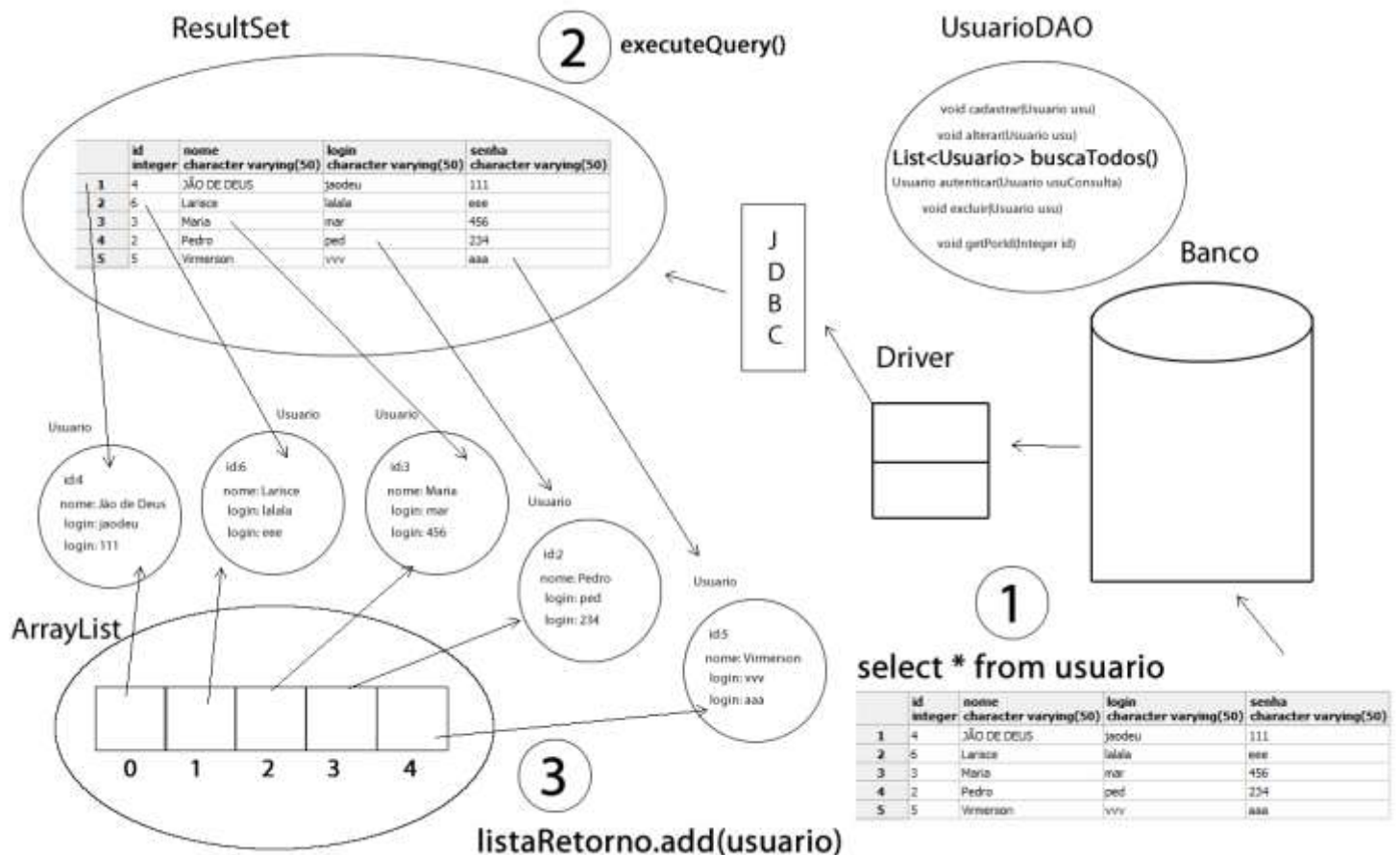
        System.out.println(
            u.getId() + " " +
            u.getNome() + " " +
            u.getLogin() + " " +

```

```

        u.getSenha());
    }
}

```



Criando Método autenticar

Este método é muito parecido com o buscaPorId, mas o critério de busca que muda, pois neste caso estamos buscando um usuário que tenha o login e a senha informado no objeto usuário de consulta.

```

public Usuario autenticar(Usuario usuConsulta){
    //Objeto de retorno do método
    Usuario usuRetorno = null;
    String sql = "select * from usuario where login=? and
senha=md5(?)";
    try {
        PreparedStatement preparador = con.prepareStatement(sql);
        preparador.setString(1, usuConsulta.getLogin());
    }
}

```

```

        preparador.setString(2, usuConsulta.getSenha());
        //Retorno da consulta em Resultset
        ResultSet resultado = preparador.executeQuery();
        //Se tem registro
        if(resultado.next()){
            //instancia o objeto Usuario
            usuRetorno = new Usuario();
            usuRetorno.setId(resultado.getInt("id"));
            usuRetorno.setNome(resultado.getString("nome"));
            usuRetorno.setLogin(resultado.getString("login"));
            usuRetorno.setSenha(resultado.getString("senha"));
            System.out.println("Usuário Autenticado");
        }
    } catch (SQLException e) {

        System.out.println("Erro de SQL:" + e.getMessage());
    }

    return usuRetorno;
}

```

Testando o método

Adicionado o novo teste na classe *TestUsuarioDAO*.

```

private static void deveAutenticar() {
    Usuario usu = new Usuario();
    usu.setLogin("jao");
    usu.setSenha("123");

    Usuario usuRetorno = usuDao.autenticar(usu);

    if(usuRetorno!=null)
        System.out.println(
            usuRetorno.getId() + " " +
            usuRetorno.getNome() + " " +
            usuRetorno.getLogin() + " " +
            usuRetorno.getSenha());
}

```

Exercício 1:

Criar Tabela Cliente no banco de dados com os seguintes campos (id, nome, e-mail, endereço)

Criar Classe Cliente com as seguintes propriedades (id, nome, e-mail, endereço)

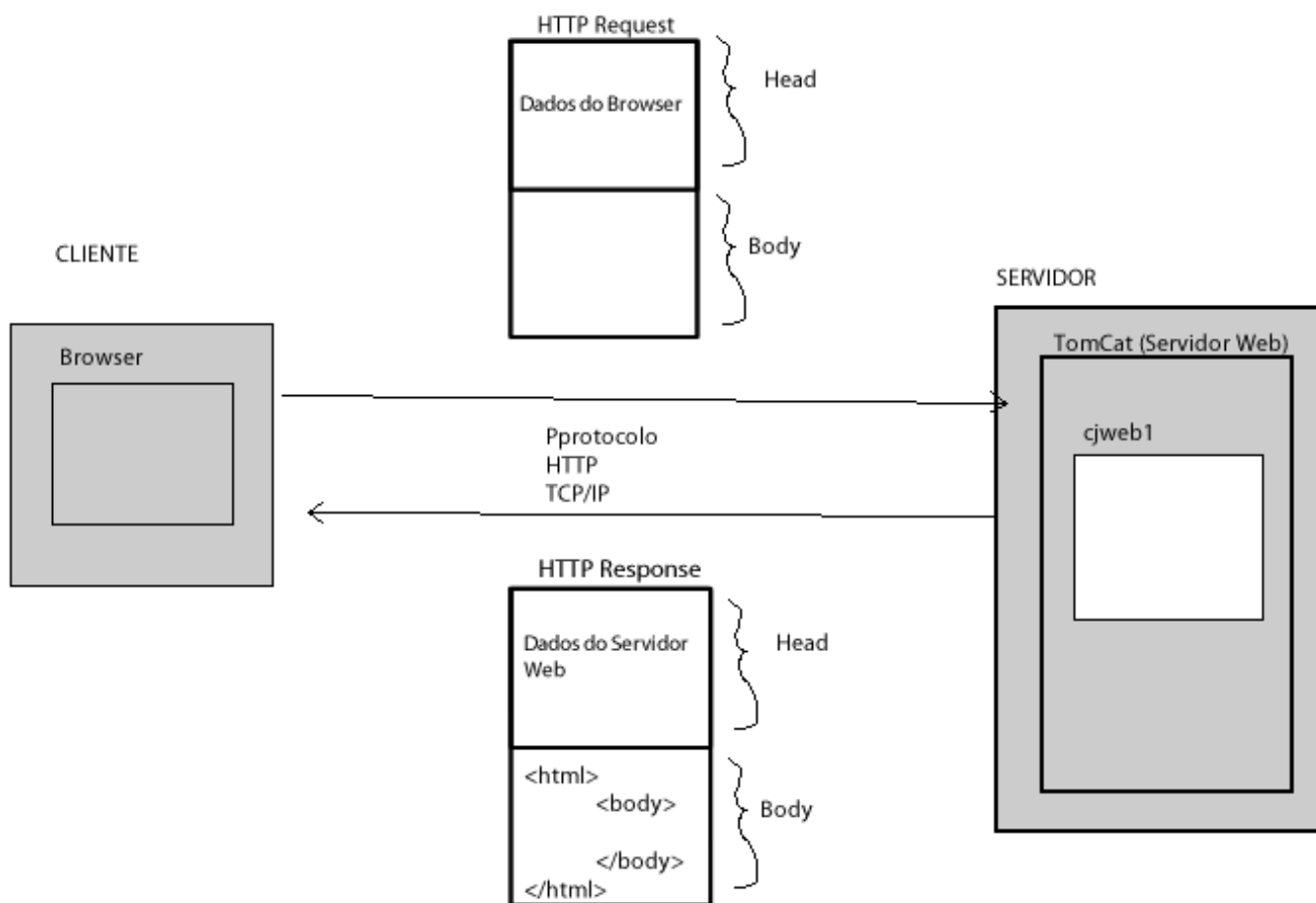
Criar Classe ClienteDAO com os Método cadastrar, excluir, alterar, buscaTodos, buscaPorId e salvar.

Criar classe de Teste TestClienteDAO com métodos para testar todos método de clienteDAO

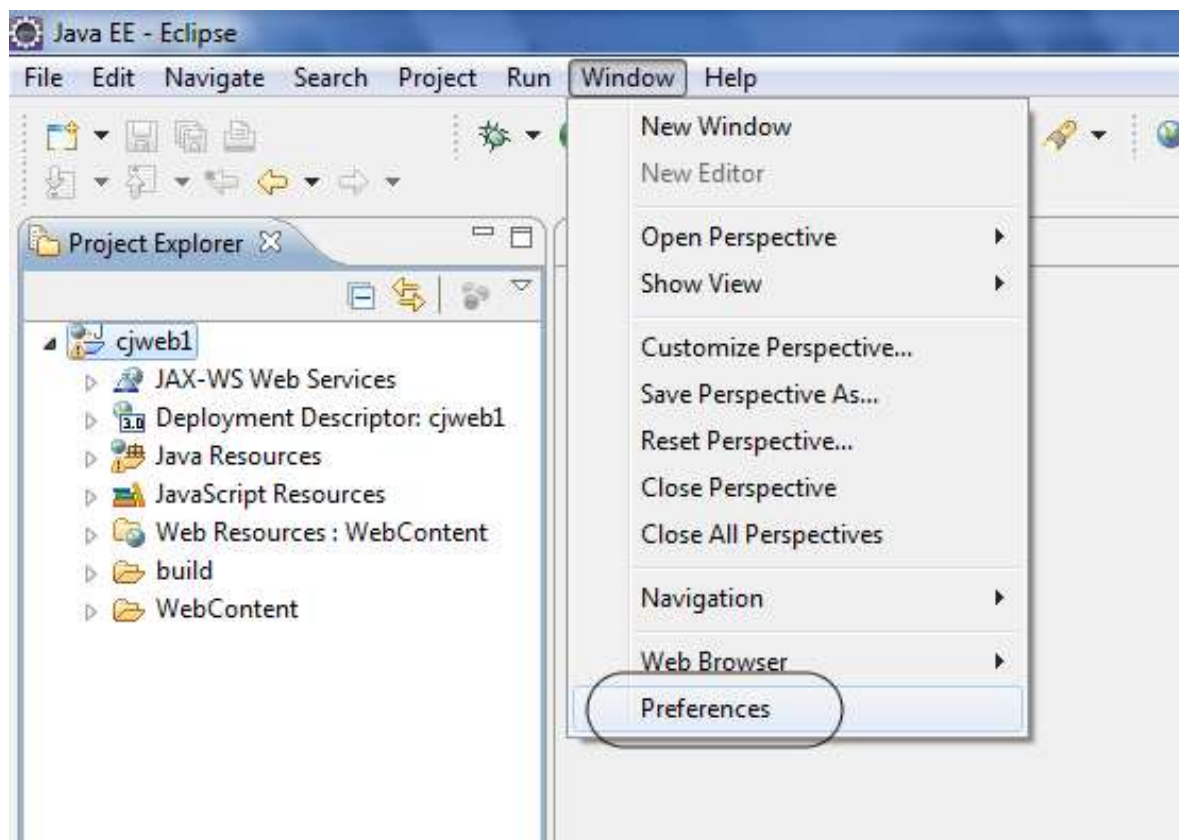
O PROTOCOLO HTTP

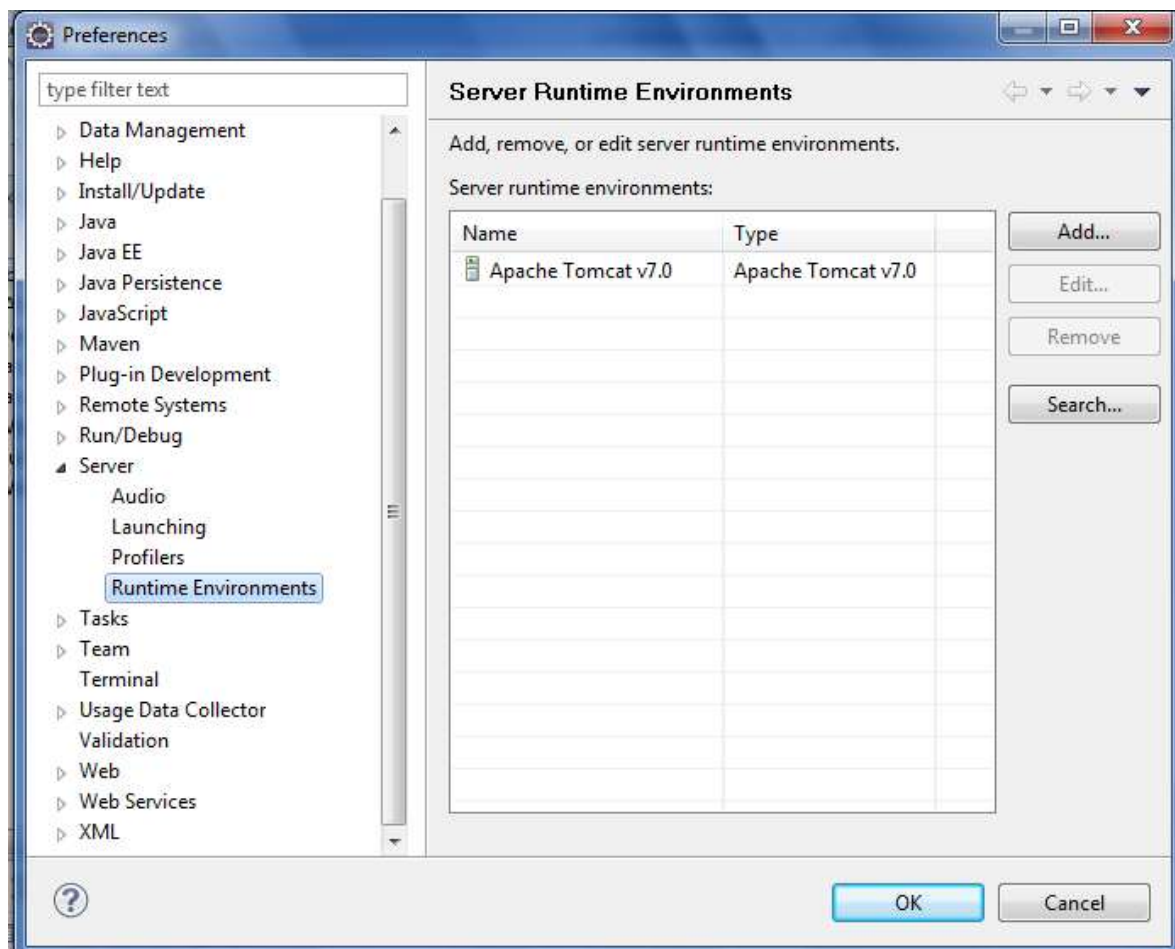
O HyperText Transfer Protocol (Protocolo de Transferência de Hipertexto - HTTP) é o protocolo de comunicação utilizado para a troca de dados entre um navegador e um servidor web. É o protocolo de comunicação que você aciona quando digita um endereço no seu navegador: <http://www...>

É para isto que existem os métodos HTTP. Dois desses métodos, associados à transferência de dados de formulários, são muito importantes: o método GET e o método POST.

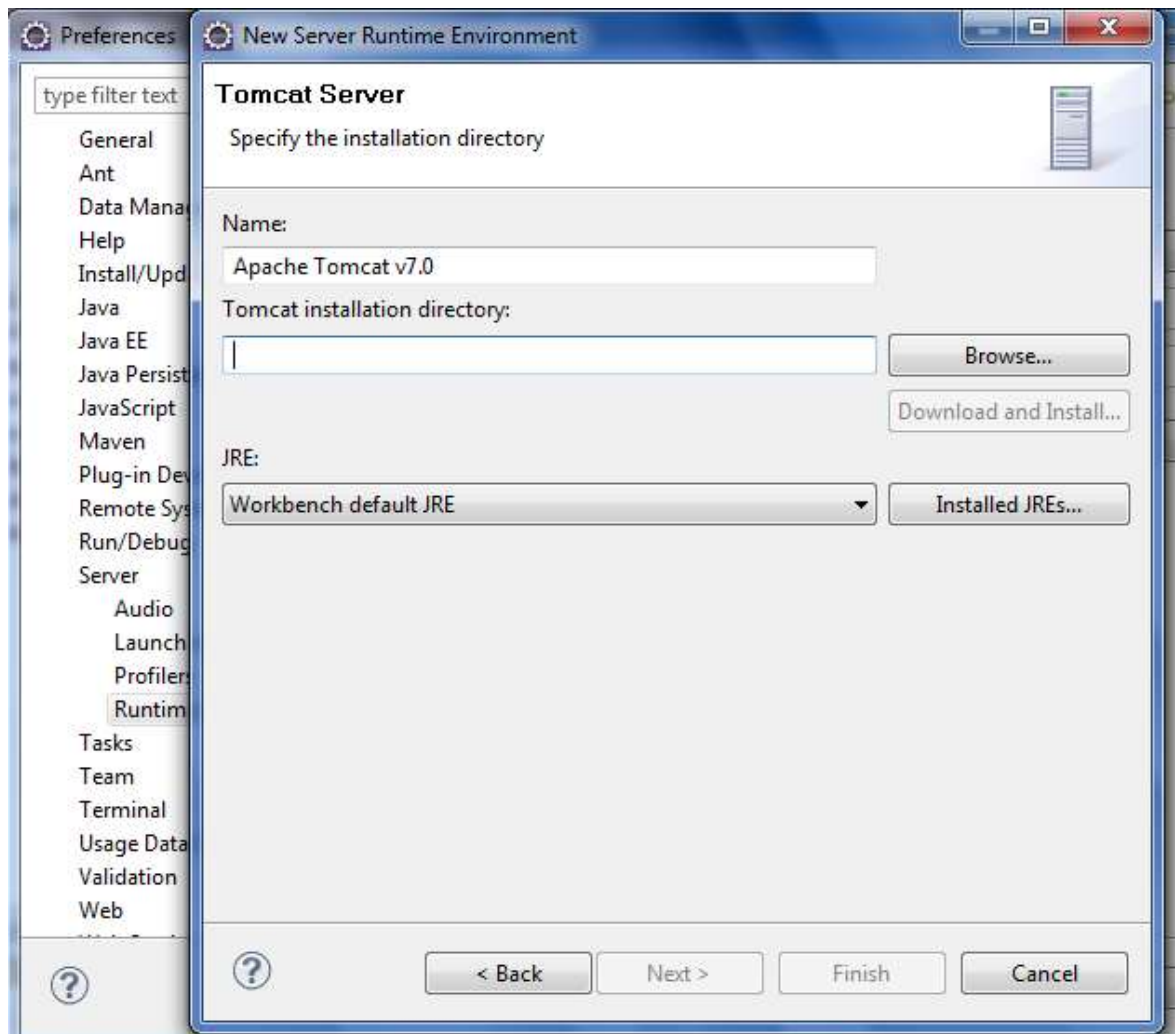


Configurando o TomCat





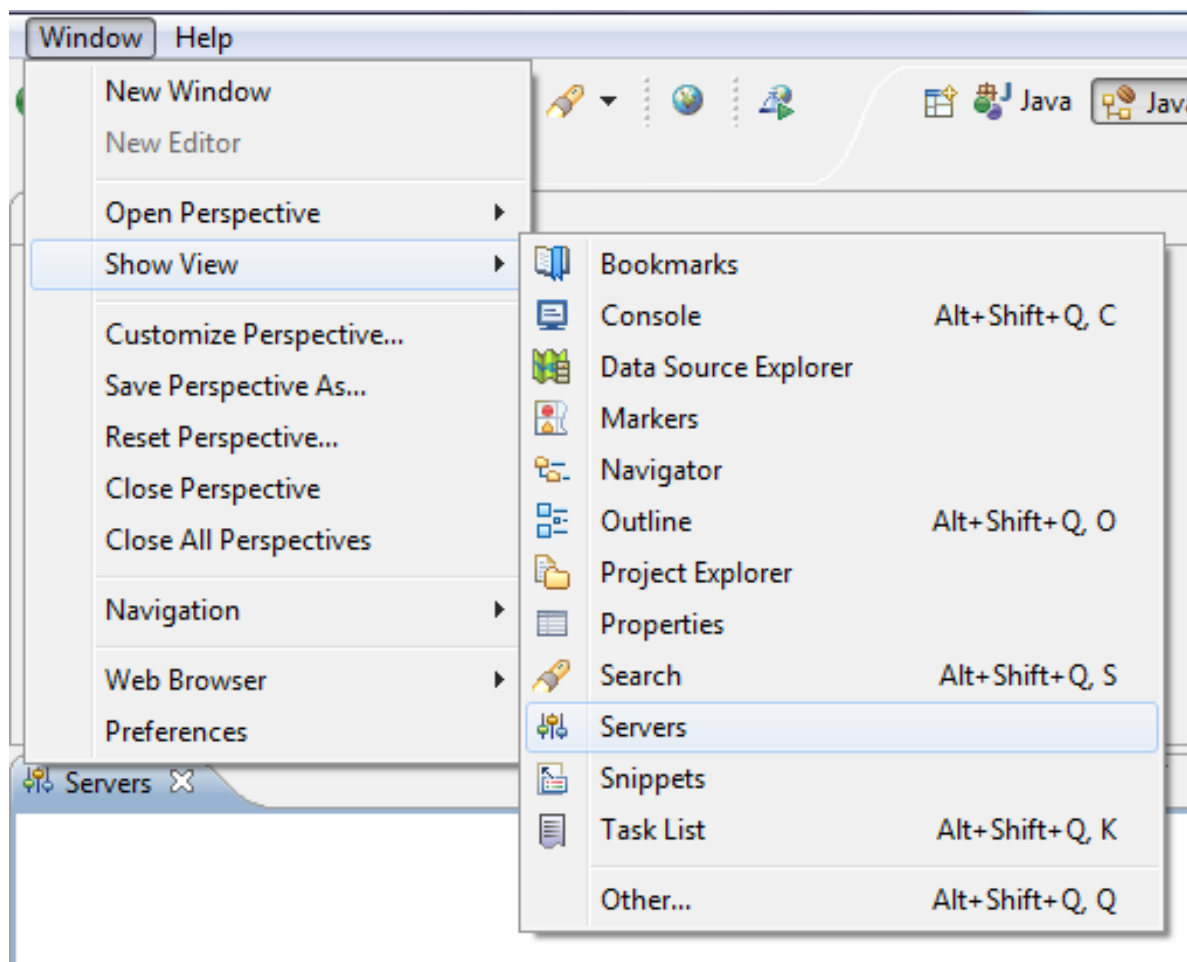
Clicando em Adicionar



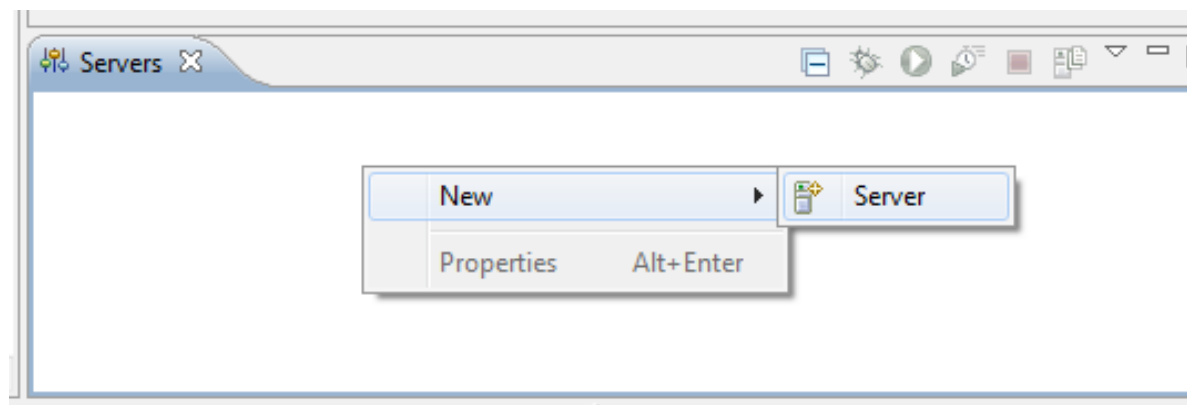
Selecione a pasta raiz do tomcat no seu computador.

Iniciando o TOMCAT

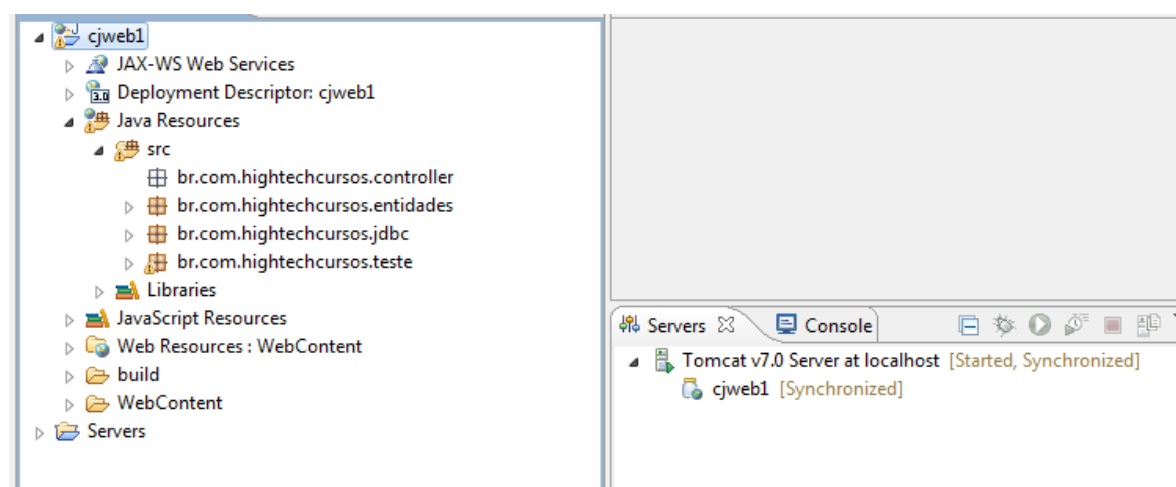
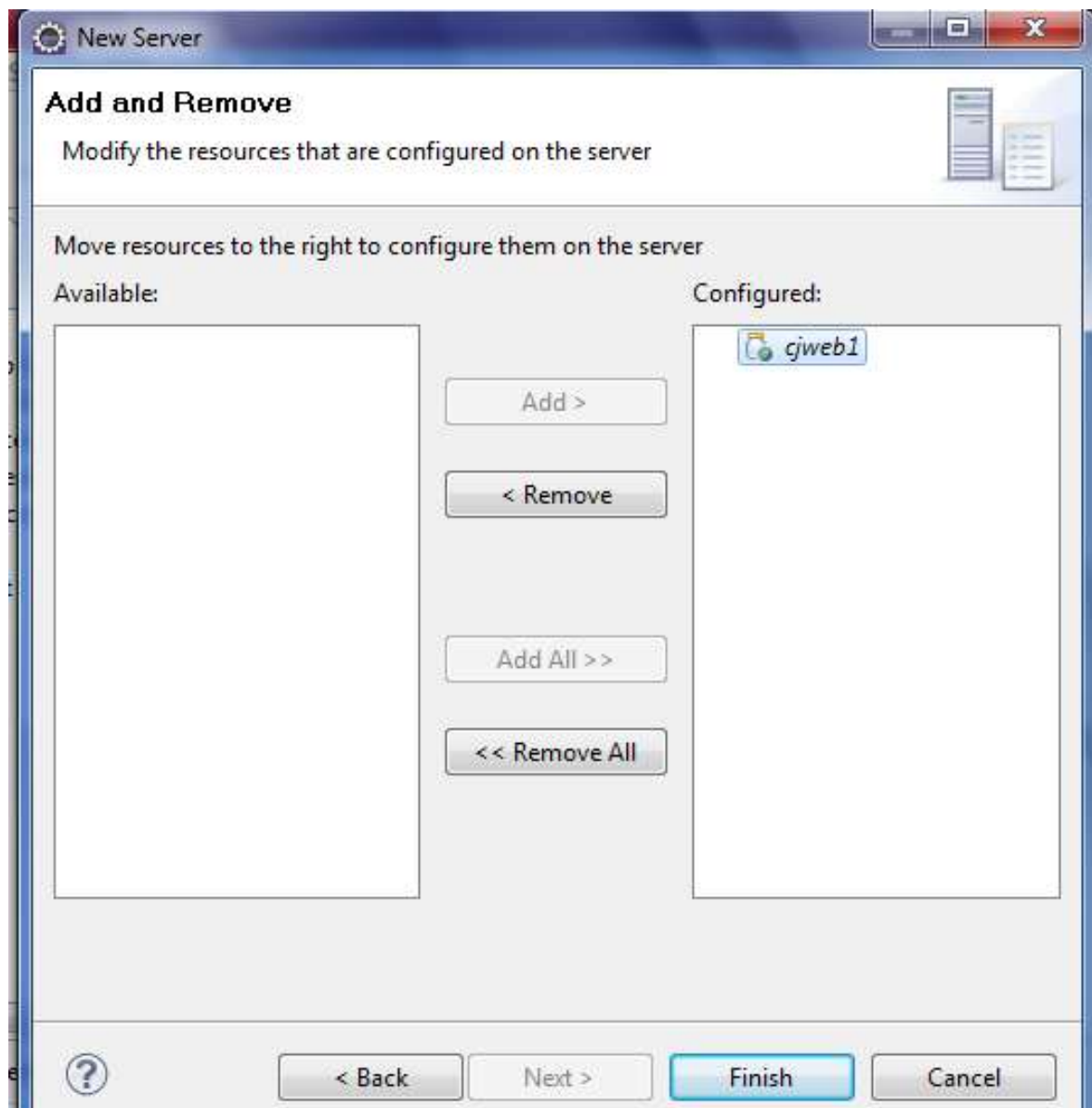
Abrindo painel de Servers para gerenciarmos e testarmos se está tudo certo com o tomcat.



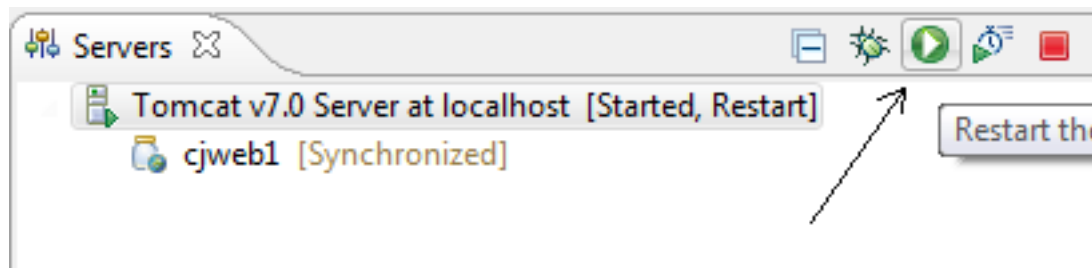
Clicando com o Direito do mouse para adicionar o controle do servidor tomcat



Adicionando o projeto no TomCat

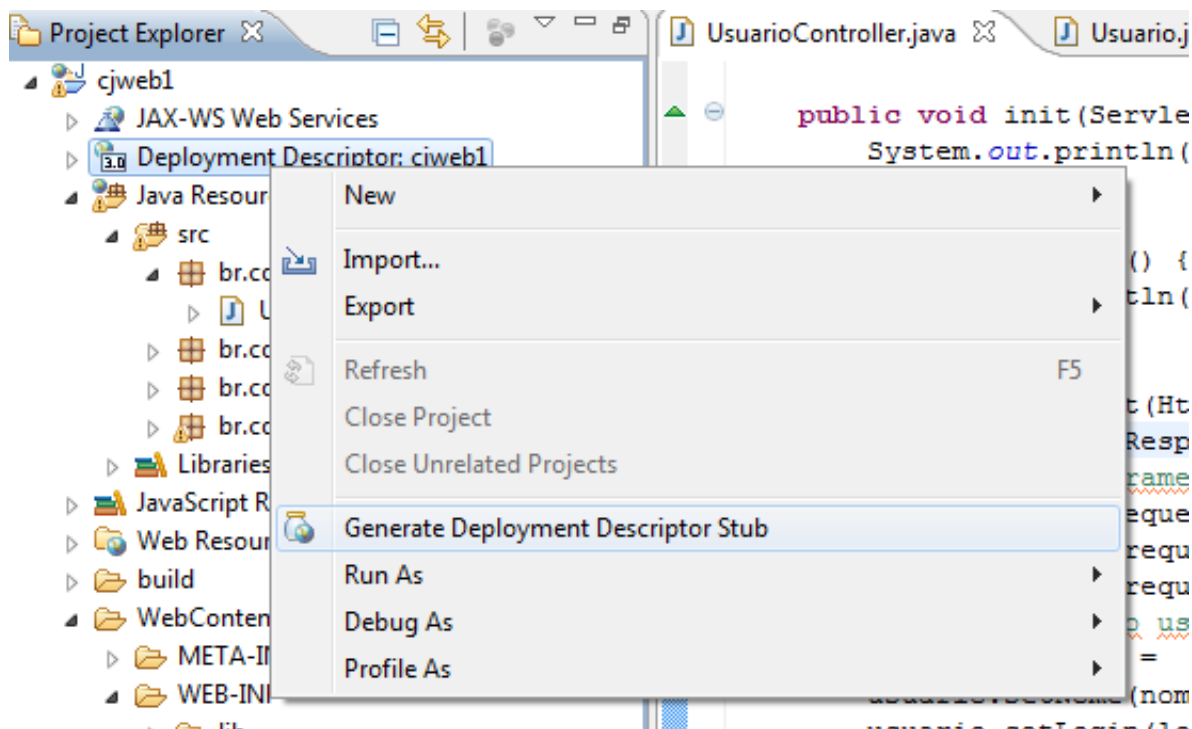


Start no TomCat

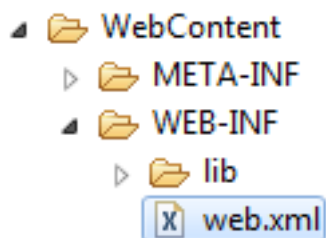


Criando web.xml

Na perspectiva Java EE, no projeto em “Deployment Descriptor” escolha “Generate ...”
Conforme figura abaixo:



Arquivo web.xml gerado

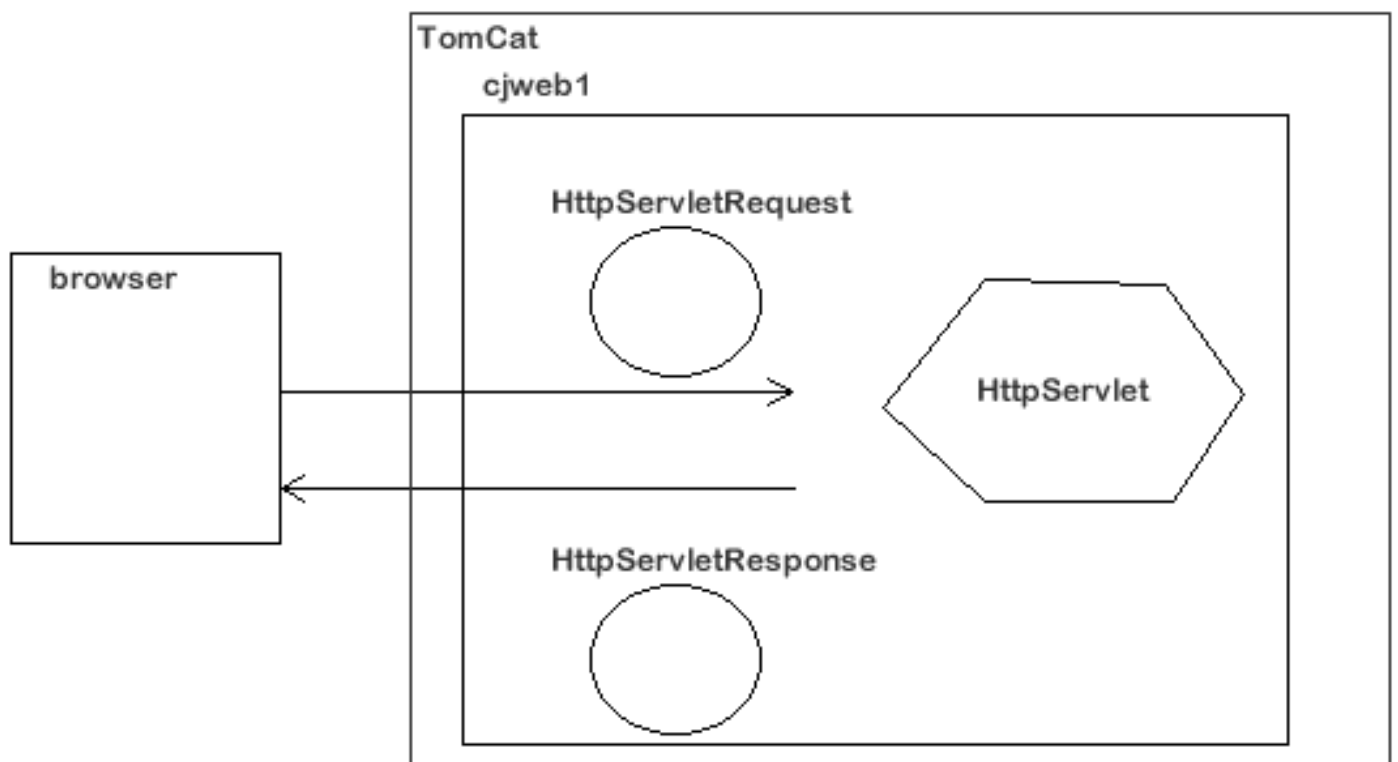


Deixe apenas o index.html como sendo o arquivo de boas vindas.

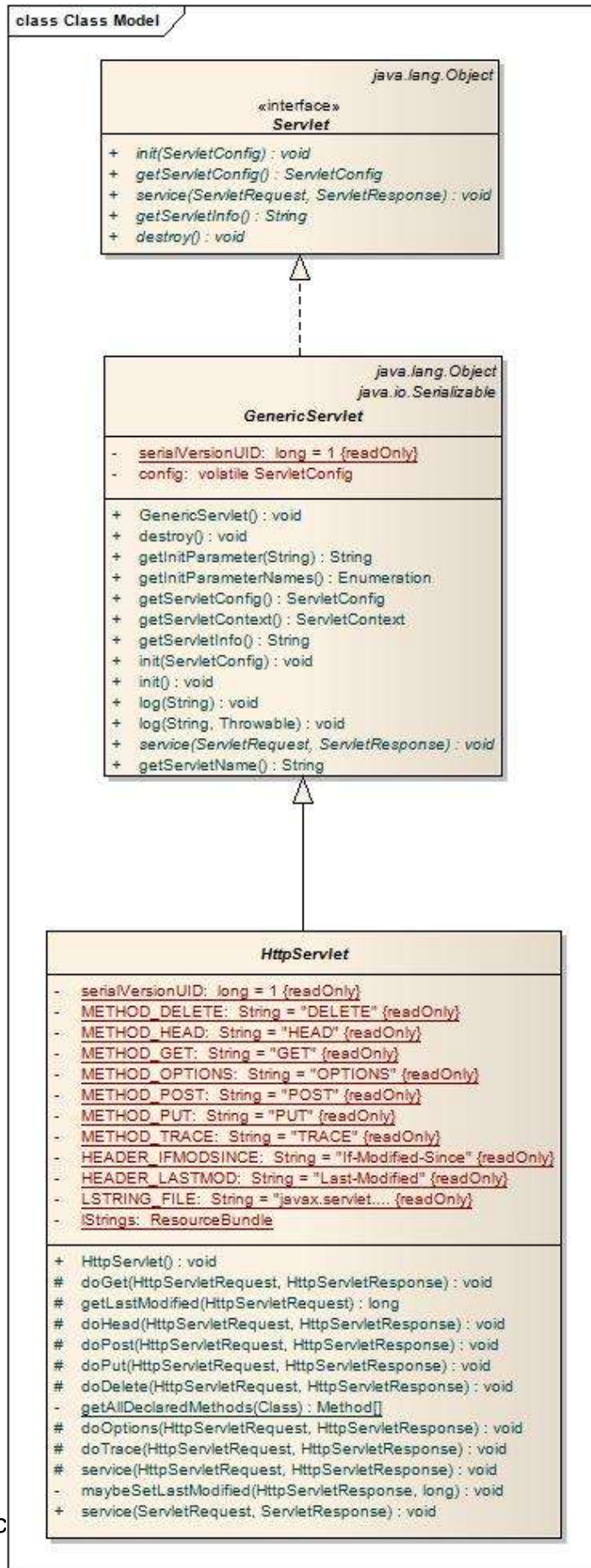
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">
  <display-name>cjweb1</display-name>
  <welcome-file-list>

    <welcome-file>index.html</welcome-file>

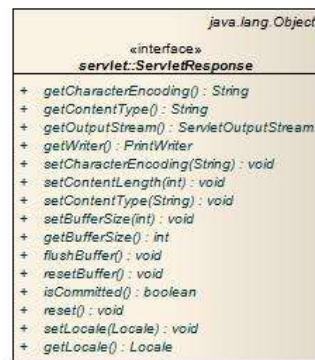
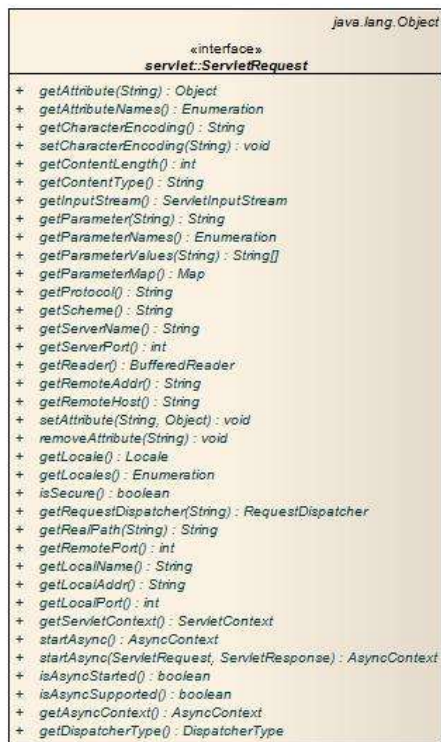
  </welcome-file-list>
</web-app>
```



Servlets

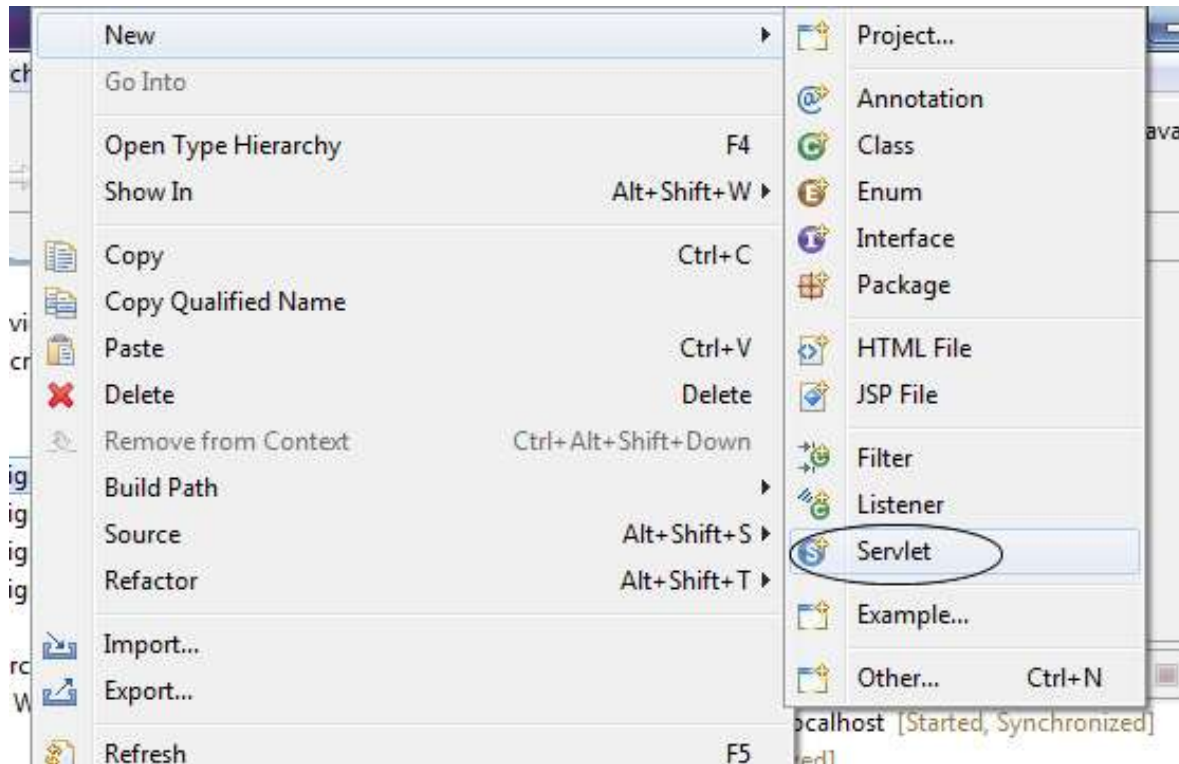


HttpServletRequest e HttpServletResponse

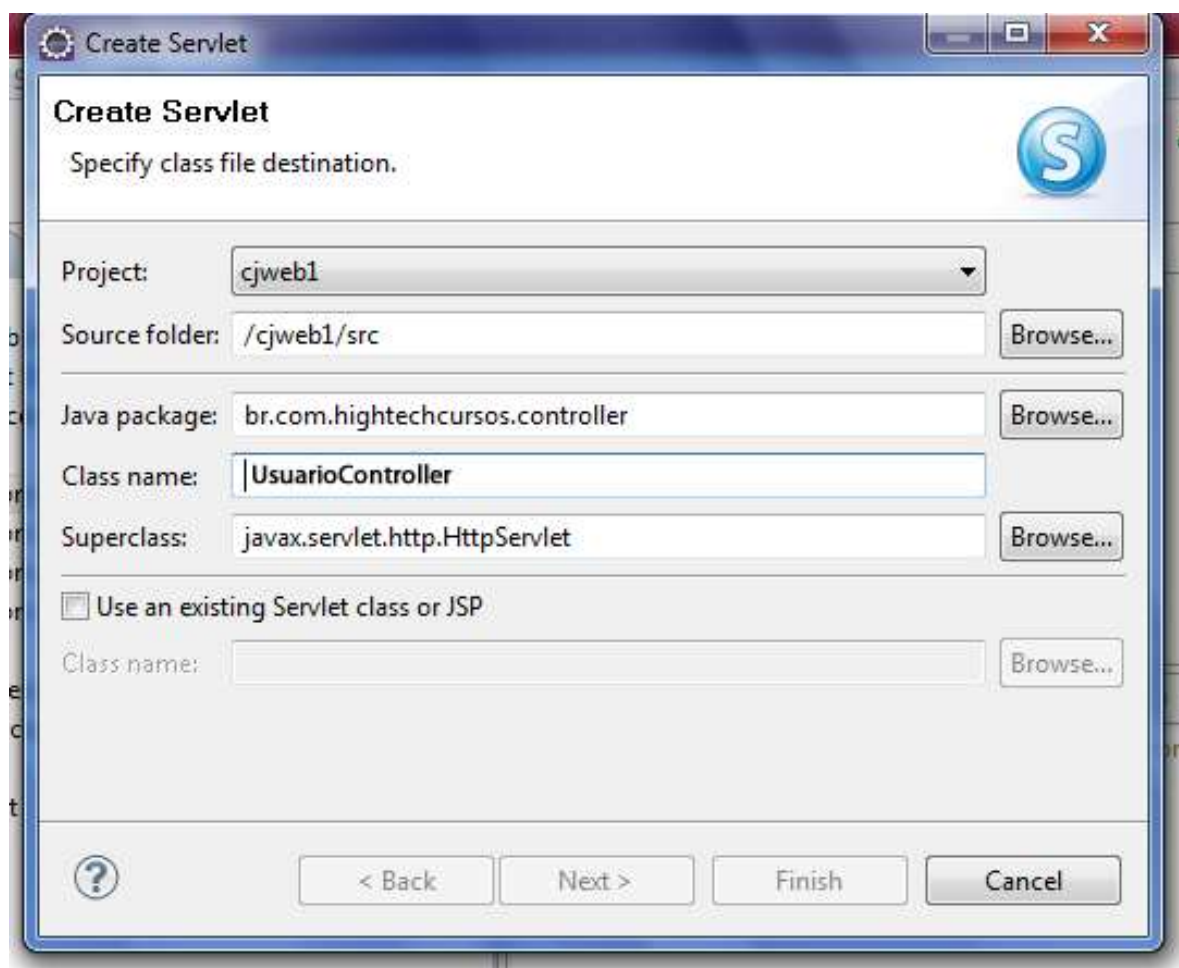


Criando um Servlet UsuarioController

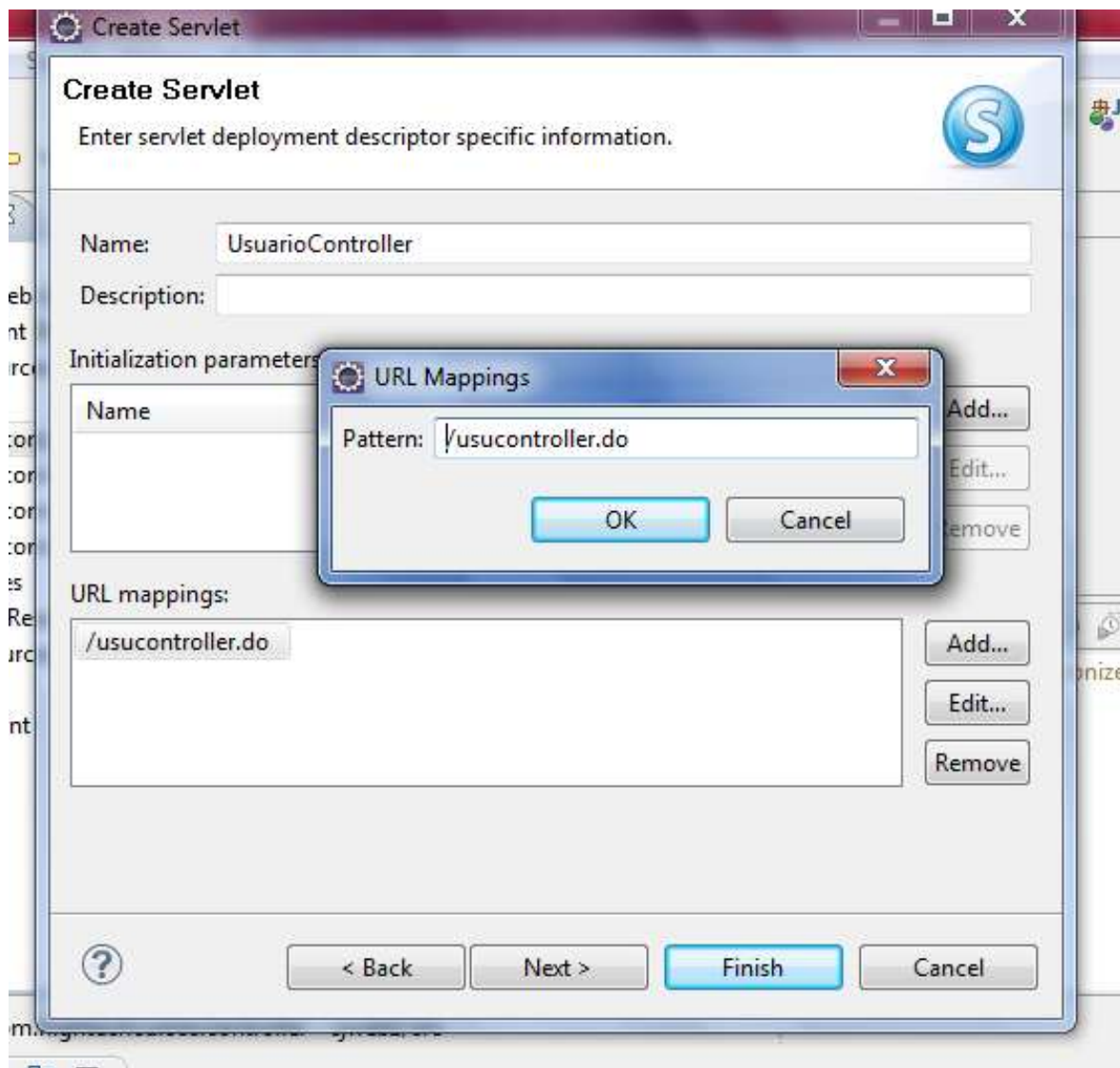
Com o botão direito do mouse no projeto



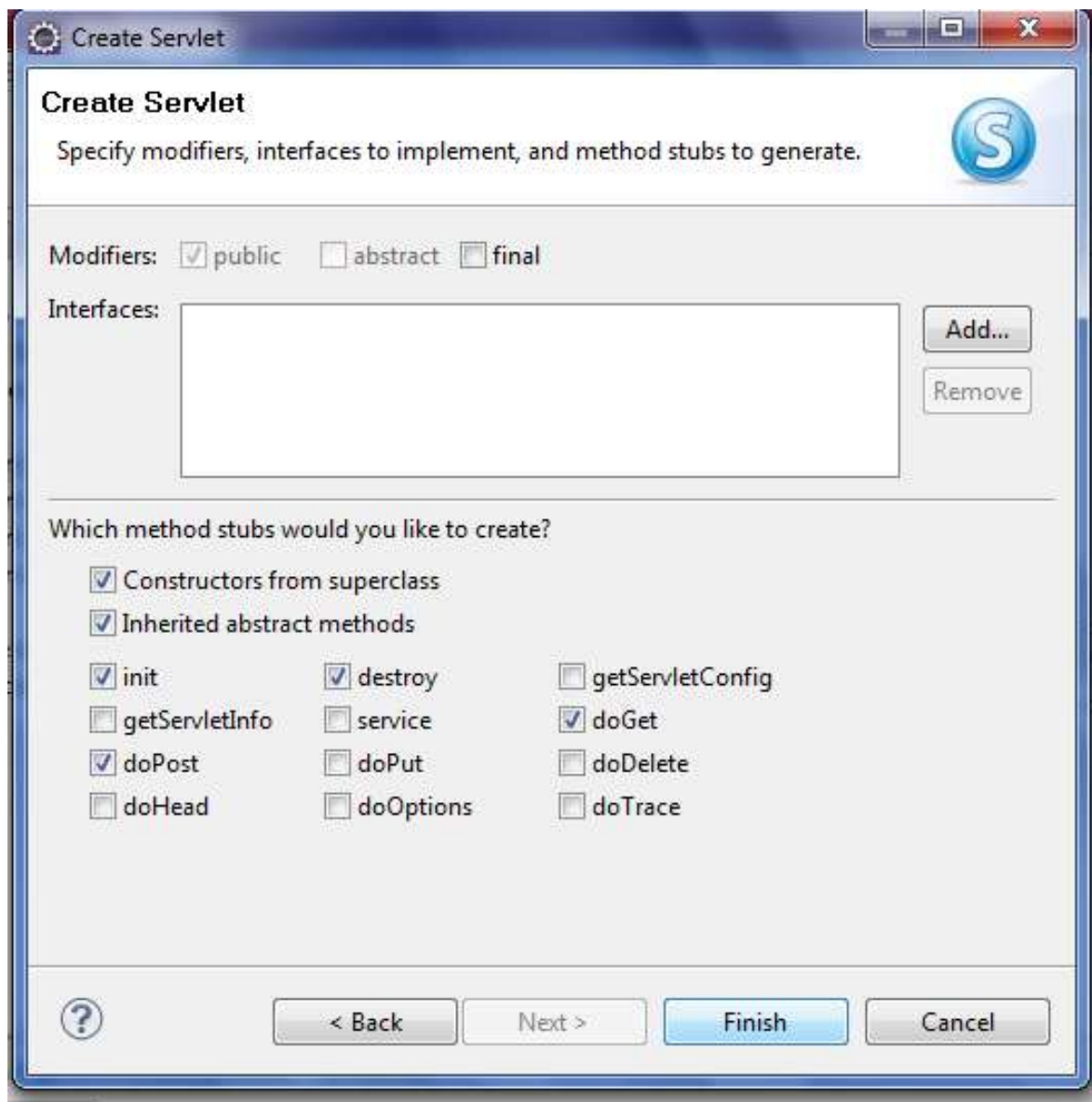
Dando um nome ao Sevlet



Dando um nome público para acesso pela URL via Browser



Selecionando os métodos a serem sobrescritos e implementados.



Estudando o ciclo de Vida do Servlet

- 1) Inicia o Objeto da classe UsuárioController chamando o construtor
- 2) Invoca o método init()
- 3) Invoca o método service(HttpServletRequest request, HttpServletResponse response)
- 4) Invoca o doGet ou doPost

Se o método da requisição for GET então invoca o método void doGet(HttpServletRequest request, HttpServletResponse response)

Se o método da requisição for POST então invoca o método void doPost(HttpServletRequest request, HttpServletResponse response)

5) Invoca o método destroy() antes de destruir o objeto

```
package br.com.hightechcursos.controller;

import java.io.IOException;
import javax.servlet.ServletConfig;
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("/usucontroller.do")
public class UsuarioController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public UsuarioController() {
        System.out.println("Chamando o construtor do Servlet");
    }

    public void init(ServletConfig config) throws ServletException {
        System.out.println("Iniciando o Servlet");
    }

    public void destroy() {
        System.out.println("Finalizando o Servlet");
    }

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
        System.out.println("Requisição pelo método GET");
    }

    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
        System.out.println("Requisição pelo Método POST");
    }
}
```

Acessando pelo browser

Chamando o Servlet no browser pelo método GET

<http://localhost:8080/cjweb1/usucontroller.do>

Saída no console

```
Chamando o construtor do Servlet  
Iniando o Servlet  
Requisição pelo método GET
```

Erro HTTP Status 404 – (Não Encontrado)

Este erro é muito comum, se ocorrer com você é porque o tomcat não encontrou o servlet ou qualquer outro recurso que está tentando chamar.

Requisição pelo método GET

O método *GET* utiliza a própria *URI* (normalmente chamada de *URL*) para enviar dados ao servidor, quando enviamos um formulário pelo método *GET*, o navegador pega as informações do formulário e coloca junto com a *URI* de onde o formulário vai ser enviado e envia, separando o endereço da *URI* dos dados do formulário por um “?” (ponto de interrogação).

Quando você busca algo no Google, ele faz uma requisição utilizando o método *GET*, você pode ver na barra de endereço do seu navegador que o endereço ficou com um ponto de interrogação no meio, e depois do ponto de interrogação você pode ler, dentre outros caracteres, o que você pesquisou no Google.

Requisição pelo Método POST

O método *POST* envia os dados colocando-os no corpo da mensagem. Ele deixa a *URI* separada dos dados que serão enviados e com isso podemos enviar qualquer tipo de dados por esse método. Quando você faz um registro em um formulário e depois de enviar a *URI* não tem o ponto de interrogação separando os dados que você digitou, provavelmente o formulário foi enviado pelo método *POST*.

Passando parâmetros pelo método GET

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException,
    IOException {

    String nome = request.getParameter("nome");
    String sexo = request.getParameter("sexo");

    System.out.println("Requisição pelo método GET: nome="+ nome + "
sexo="+sexo);
}
```

No browser

<http://localhost:8080/cjweb1/usucontroller.do?nome=maria&sexo=f>

Saída no console

```
Chamando o contrutor do Servlet
Iniando o Servlet
Requisição pelo método GET: nome=maria sexo=f
```

Salvando usuário no banco pelo método GET

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException,
    IOException {
    //Capturando parametros da tela
    String nome = request.getParameter("nome");
    String login = request.getParameter("login");
    String senha = request.getParameter("senha");
    //criando objeto usuario e atribuindo valores da tela
    Usuario usuario = new Usuario();
    usuario.setNome(nome);
    usuario.setLogin(login);
    usuario.setSenha(senha);
    //criando um usuarioDAO
    UsuarioDAO usuDao = new UsuarioDAO();
    //Salvando no banco de dados
    usuDao.salvar(usuario);
    System.out.println("Salvo com Sucesso");
}
```



```
}
```

Requisição do Browser

<http://localhost:8080/ciweb1/usucontroller.do?nome=jao&login=jj&senha=123>

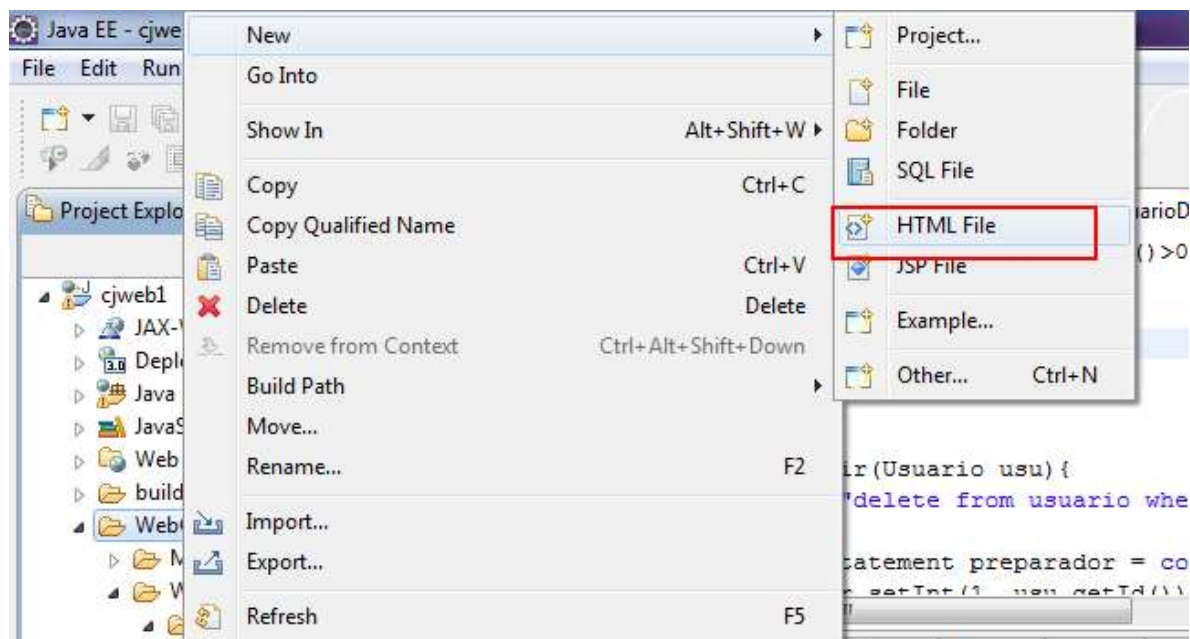
Console

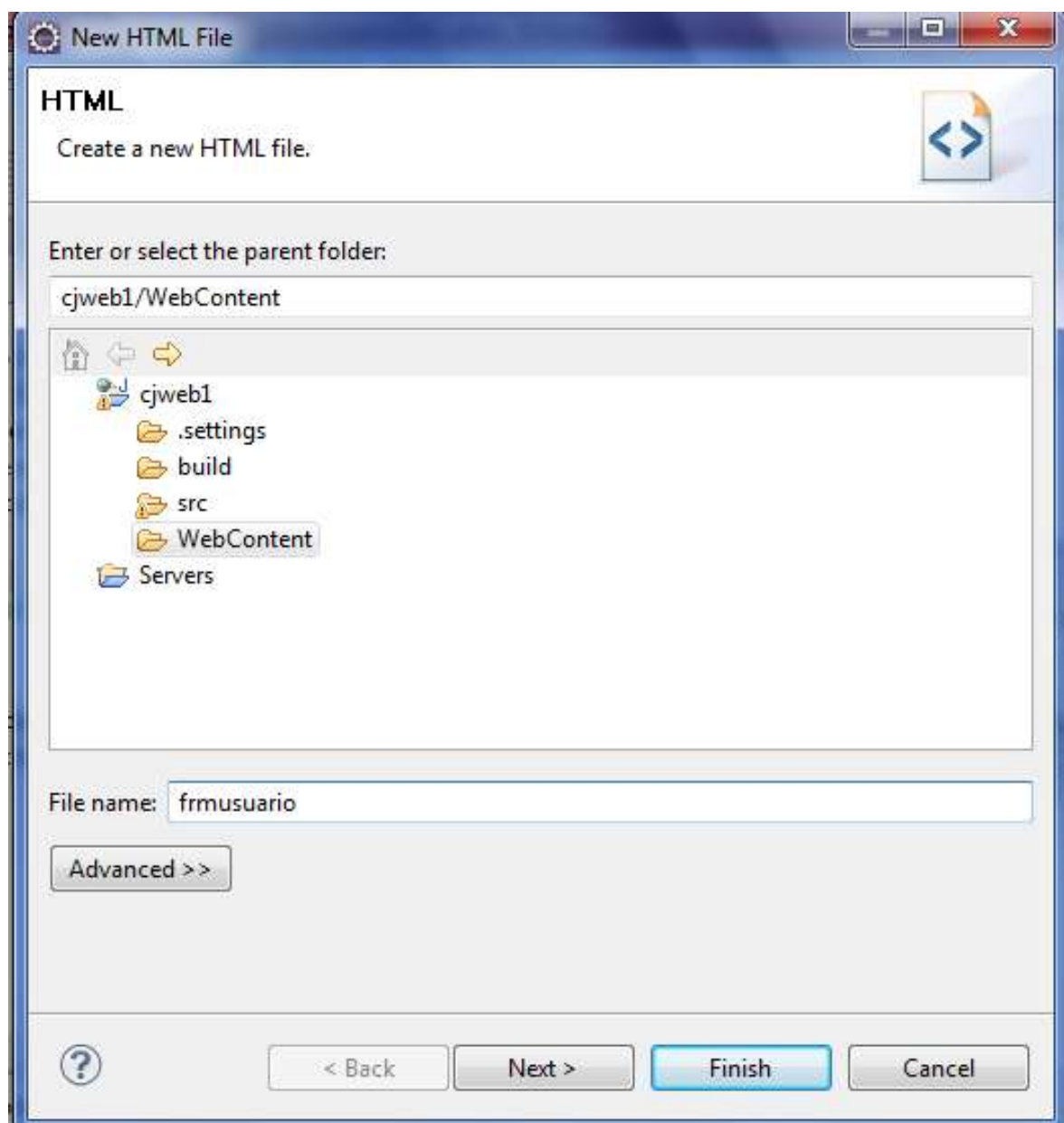
```
Chamando o contrutor do Servlet  
Iniciando o Servlet  
Conectado  
Cadastrado com sucesso!  
Salvo com Sucesso
```

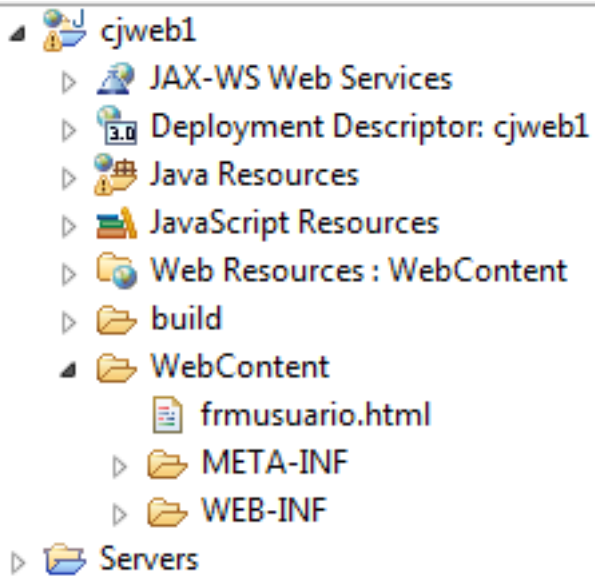
Banco

	id [PK] serial	nome character vai	login character vai	senha character varying(50)
4	7	Jão de Deus	jaodeu	123454545454
5	8	tatatat	tatatatat	tatatat
6	9	tetet	ete	rere
7	10	Maria	Mar	123333
8	11	CARLOS	CAR	123
9	12	maria	ma	202cb962ac59075b964b07152d234b70
10	13	jaojunior	jju	827ccb0eea8a706c4c34a16891f84e7b
11	14	jao	jj	202cb962ac59075b964b07152d234b70

Criando um Formulário de Cadastro de usuário – frmusuario.html







```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>

    <form method="post" action="usucontroller.do">
        Nome: <input type="text" name="nome">
        Login: <input type="text" name="login">
        Senha: <input type="password" name="senha">
        <input type="submit" value="SALVAR">
    </form>

</body>
</html>
```

Cadastrando pelo método post

```
protected void doPost(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {

    System.out.println("Requisição pelo Método POST");
    String nome = request.getParameter("nome");
    String login = request.getParameter("login");
    String senha = request.getParameter("senha");
    //criando objeto usuario e atribuindo valores da tela
    Usuario usuario = new Usuario();
    usuario.setNome(nome);
    usuario.setLogin(login);
    usuario.setSenha(senha);
    //criando um usuarioDAO
    UsuarioDAO usuDao = new UsuarioDAO();
    //Salvando no banco de dados
    usuDao.salvar(usuario);
    //Mensagem no Browser
    response.getWriter().print("Salvo com Sucesso");
}
```

Chamando a Tela de Cadastro no Browser

<http://localhost:8080/cjweb1/frmusuario.html>

Nome:	<input type="text" value="mateus"/>	Login:	<input type="text" value="mat"/>	Senha:	<input type="password"/>	<input type="button" value="SALVAR"/>
-------	-------------------------------------	--------	----------------------------------	--------	--------------------------	---------------------------------------

Acrescentando campo ID para realizar alteração

Aproveitamos o mesmo formulário para realizarmos a alteração dos dados.

Em nosso objeto UsuarioDao já está programado para cadastrar ou alterar com base no valor da propriedade id do Objeto Usuário.

```
<form method="post" action="usucontroller.do">
  ID: <input size="5" type="text" name="id">
  Nome: <input type="text" name="nome">
  Login: <input type="text" name="login">
```

```
Senha: <input type="password" name="senha">
<input type="submit" value="SALVAR">
</form>
```

ID: Nome: Login: Senha:

Alteração - Capturando o ID no Servlet

Para capturar o id no Servlet é a mesma coisa que os parâmetros.

Observe que devemos fazer uma conversão (casting) na String *id* para “Setar” na propriedade *id* do usuário através do método *setId(Integer id)*.

```
String id = request.getParameter("id");
String nome = request.getParameter("nome");
String login = request.getParameter("login");
String senha = request.getParameter("senha");
//criando objeto usuario e atribuindo valores da tela
Usuario usuario = new Usuario();

//Testando se o id é diferente de null e também não é vazio
if(id!=null && !id.isEmpty()){
    //Se o id fosse vazio ou null então daria Exception no Parse
    usuario.setId(Integer.parseInt(id));
}
```

O teste para saber se o id é diferente de nulo e de vazio serve para tratarmos a conversão *Integer.parseInt(id)* para evitar o lançamento da *Exception* que é lançada quando tentamos converter algo nulo ou vazio em inteiro.

Exclusão - método doGet

Agora vamos realizar a exclusão dos registros de usuário por meio do método *doGet*.

Criaremos um variável “acao” para nos ajudar a tomarmos decisões dentro do *doGet*, pois este método poderá ser chamado para diversas funcionalidades, e para conseguirmos tratar cada uma delas, vamos diferenciar pelo parâmetro “acao” que deverá vir na requisição do cliente.

```

protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {

    // Captura a acao
    String acao = request.getParameter("acao");
    // Testa se acao foi informada
    if (acao != null && !acao.isEmpty()) {
        // Cria objeto dao para operacao no banco de dados
        UsuarioDAO usuarioDAO = new UsuarioDAO();
        // testa se a acao é de exclusao
        if (acao.equals("exc")) {
            // Caputara o id a ser excluido
            String id = request.getParameter("id");
            // valida se o id for informado
            if (id != null && !id.isEmpty()) {
                // Criando Objeto Usuario para passa
                // ao método do excluir do UsuarioDAO
                Usuario usuExc = new Usuario();
                // Convertendo e setando o id
                usuExc.setId(Integer.parseInt(id));
                usuarioDAO.excluir(usuExc);
                response.getWriter().print("Excluido com sucesso!");
            }
        }
    }
}

```

Requisição do Browser

Na requisição abaixo o teste `!acao.isEmpty()` será **false**

<http://localhost:8080/cjweb1/usuariocontroller.do?acao=&id=19>

Na requisição abaixo o teste `acao != null` será **false**

<http://localhost:8080/cjweb1/usuariocontroller.do?id=19>

Na requisição abaixo o teste `acao.equals("exc")` será **false**

<http://localhost:8080/cjweb1/usuariocontroller.do?acao=xxx&id=19>

Na requisição abaixo o usuário será excluído com sucesso caso o id exista no banco

<http://localhost:8080/cjweb1/usuariocontroller.do?acao=exc&id=19>

Listando Usuário

A Saída Desejada

```
<html>
<body>
<table border="1">
  <tr>
    <td> Id</td> <td> Nome</td> <td> Login </td> <td> Senha </td>
  </tr>
  <tr>
    <td> 1</td> <td> Jao </td> <td> jj </td> <td> 123 </td>
  </tr>
  <tr>
    <td> 2</td> <td> Maria </td> <td> ma </td> <td> m123 </td>
  </tr>
</table>
</body>
</html>
```

Implementando HTML no Servlet

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    .....

    UsuarioDAO usuarioDAO = new UsuarioDAO();
    List<Usuario> lista = usuarioDAO.buscaTodos();

    String htmlSaida = "<html> <body> <table border='1' >" +
        "<tr> <td> Id</td> <td> Nome</td> <td> Login </td> <td> Senha
</td></tr>";

    for(Usuario usu : lista){

        htmlSaida += "<tr> <td>"+ usu.getId()+ " </td> <td> "+ usu.getNome() + " </td>
<td> "+usu.getLogin()+ " </td> <td> "+usu.getSenha()+ " </td>      </tr>";
    }

    htmlSaida+= "</table></body></html>";

    PrintWriter saida= response.getWriter();
    saida.println(htmlSaida);

    ....
}
```



```
}
```

Para listamos os usuários cadastrados vamos fazer as requisições ao Servlet pelo método doGet passando como parâmetro a “acao” igual a “lis”.

O Servlet valida a se o parâmetro é igual a lis e então pede ao UsuarioDao a List<Usuario>.

Como queremos apresentar uma página HTML com conteúdo dinâmico, então será necessário criar uma página JSP para fazer essa tarefa. Até daria para fazermos isso dentro do Servlet, mas seria um caos termos que programar um monte de HTML dentro de uma variável String.

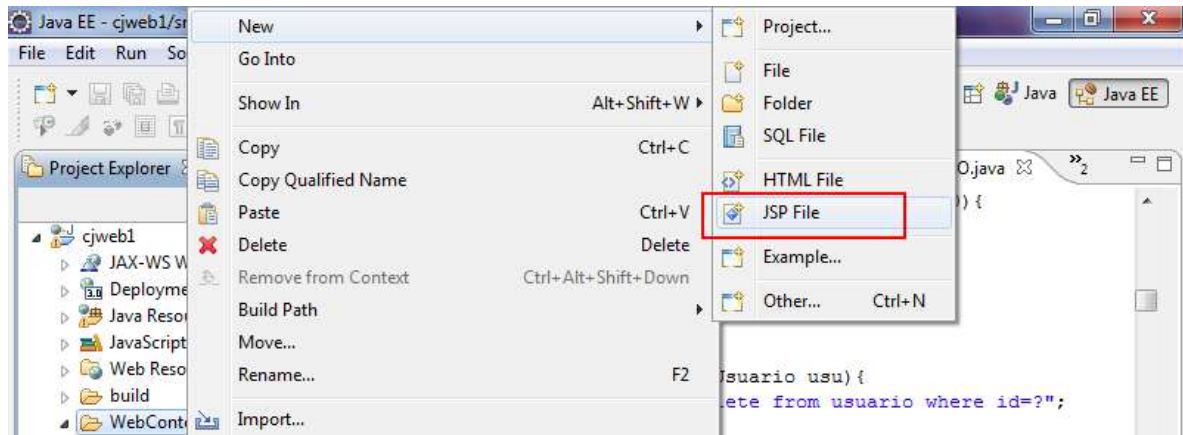
```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {

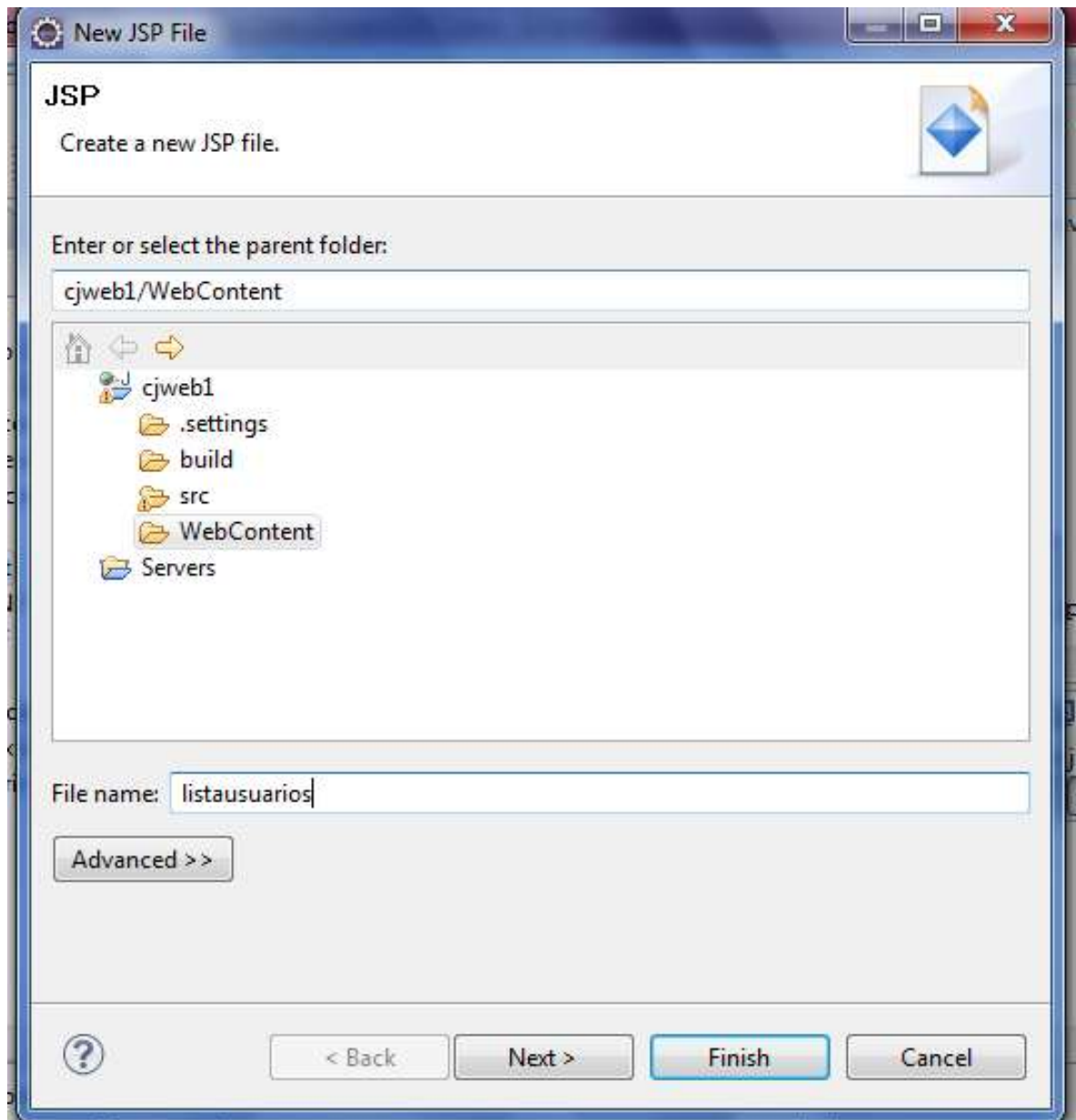
    // Captura a acao
    String acao = request.getParameter("acao");
    // Testa se acao foi informada
    if (acao != null && !acao.isEmpty()) {
        // Cria objeto dao para operacao no banco de dados
        UsuarioDAO usuarioDAO = new UsuarioDAO();
        // testa se a acao é de exclusao
        if (acao.equals("exc")) {
            // Captura o id a ser excluido
            String id = request.getParameter("id");
            // valida se o id for informado
            if (id != null && !id.isEmpty()) {
                Usuario usuExc = new Usuario();
                usuExc.setId(Integer.parseInt(id));
                usuarioDAO.excluir(usuExc);
                response.getWriter().print("Excluido com sucesso!");
            }
        } else if (acao.equals("lis")) {
            // Captura a lista de usuários do banco
            List<Usuario> lista = usuarioDAO.buscaTodos();
            // Adidicio a lista no request
            request.setAttribute("lista", lista);
            // encaminha o request e o response para o JSP
            request.getRequestDispatcher("listausuarios.jsp").forward(request,
response);
        }
    }
}
```

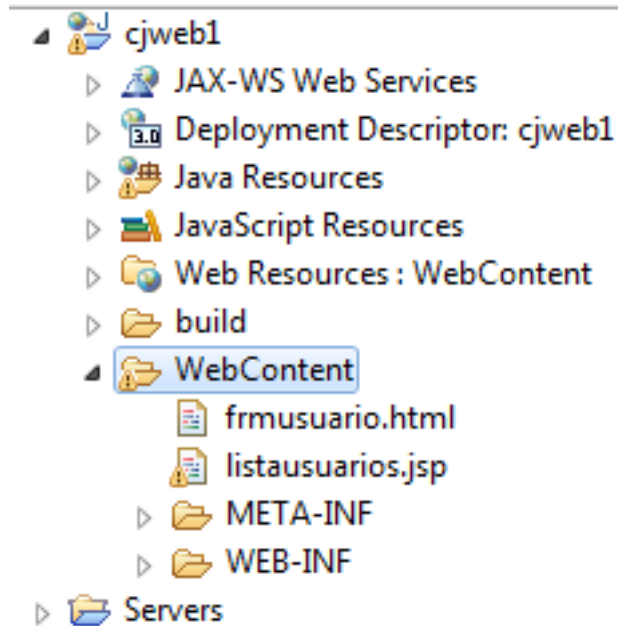
Imprimindo a lista de Usuários

Criando a página JSP de apresentação da lista de usuários.

Dentro da pasta WebContent crie um arquivo JSP chamado listausuarios.jsp







Esta página utiliza comandos *java* conhecidos como *scriptlet* para processar no lado do servidor o conteúdo dinâmico da página.

```
<%@page import="br.com.hightechcursos.entidades.Usuario"%>
<%@page import="java.util.List"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Lista de Usuários</title>
</head>
<body>
<%
    //captuando a lista do request
    List<Usuario> lista = (List<Usuario>) request.getAttribute("lista");
%>

<table border="1">
    <tr bgcolor="#eaeaea">
        <td> ID </td>
        <td> NOME </td>
        <td> LOGIN </td>
        <td> SENHA </td>
    </tr>

    <% for (Usuario u: lista) {%>
    <tr>
        <td> <%= u.getId() %> </td>
        <td> <%= u.getNome() %> </td>
```

```

        <td> <%= u.getLogin() %> </td>
        <td> <%= u.getSenha() %> </td>
    </tr>
    <% } %>

</table>

</body>
</html>

```

O sinal `<%=` é um atalho para o comando `out.print()`.

Então o comando completo seria: `<% out.print(u.getId()); %>`

O comando abreviado fica: `<%= u.getId() %>`

Note que no comando abreviado não se coloca o ponto e vírgula, pois se o fizéssemos seria equivalente ao seguinte comando: `<% out.print(u.getId()); %>`

Então estaríamos colocando ponto e vírgula como sendo parte do parâmetro do método `print`, o que causaria um erro.

Chamando no Browser a lista

<http://localhost:8080/cjweb1/usucontroller.do?acao=lis>

Excluindo múltiplos registros selecionados

Agora se faz necessário criar um parâmetro para controlar ação, pois cada o método post será chamado para salvar usuários e também excluir usuários. Para diferenciar as chamadas do formulário de cadastro do formulário de exclusão criaremos um campo oculto nos formulários para passar as ações correspondentes.

Acrescentando campo oculto no formulário de cadastro frmusuario.html

Colocando um campo “acao” igual a salvar para controlarmos no `servlet` a requisição.

```

<form method="post" action="usucontroller.do">
    <input type="hidden" name="acao" value="salvar">
    ID: <input size="5" type="text" name="id">
    Nome: <input type="text" name="nome">
    Login: <input type="text" name="login">
    Senha: <input type="password" name="senha">
    <input type="submit" value="SALVAR">
</form>

```

Acrescentando formulário na lista

Agora se faz necessário colocar a lista entre as tags `<form>` e `</form>` para submeter os ids selecionados pelos *checkbox* com os *ids* a serem registrados.

```
<body>
<%
    //captuando a lista do request
    List<Usuario> lista = (List<Usuario>) request.getAttribute("lista");
%>
<form action="usucontroller.do" method="post">
<input type="hidden" name="acao" value="exc">
<table border="1">
    <tr bgcolor="#eaeaea">
        <td> ID </td>
        <td> NOME </td>
        <td> LOGIN </td>
        <td> SENHA </td>
        <td> ACAO </td>
    </tr>
    <% for (Usuario u: lista) {%>
    <tr>
        <td> <%= u.getId() %> </td>
        <td> <%= u.getNome() %> </td>
        <td> <%= u.getLogin() %> </td>
        <td> <%= u.getSenha() %> </td>
        <td> <input type="checkbox" name="id" value="<%= u.getId() %>" </td>
    </tr>
    <% } %>
</table>
<input type="submit" value="Excluir">
</form>
</body>
```

Carregando a página

<http://localhost:8080/cjweb1/usucontroller.do?acao=lis>

ID	NOME	LOGIN	SENHA	ACAO
5	Virmerson	vvv	aaa	<input type="checkbox"/>
6	Larisce	lalala	eee	<input type="checkbox"/>
7	Jão de Deus	jaodeu	1234545454	<input type="checkbox"/>
8	tatatat	tatatatat	tatatat	<input type="checkbox"/>
9	tetet	ete	rere	<input checked="" type="checkbox"/>
10	Maria	Mar	123333	<input checked="" type="checkbox"/>
11	CARLOS	CAR	123	<input checked="" type="checkbox"/>

Excluir

Alterando o método *doPost* para realizar as operações de salvar e de exclusão de múltiplos *ids* selecionados.

```
protected void doPost(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {

    System.out.println("Requisição pelo Método POST");

    // Captura a acao
    String acao = request.getParameter("acao");

    // criando um usuarioDAO
    UsuarioDAO usuDao = new UsuarioDAO();

    if(acao.equals("salvar")){
        String id = request.getParameter("id");
        String nome = request.getParameter("nome");
        String login = request.getParameter("login");
        String senha = request.getParameter("senha");
        // criando objeto usuario e atribuindo valores da tela
        Usuario usuario = new Usuario();
        // Testando se o id é diferente de null e também não é vazio
        if (id != null && !id.isEmpty()) {
            // Se o id fosse vazio ou null então daria Exception do Parse
            usuario.setId(Integer.parseInt(id));
        }

        usuario.setNome(nome);
        usuario.setLogin(login);
        usuario.setSenha(senha);

        // Salvando no banco de dados
        usuDao.salvar(usuario);
        // Mensagem no Browser
    }
}
```

```

        response.getWriter().print("Salvo com Sucesso");

    }else if(acao.equals("exc")){
        //Captura todos os id checados
        String ids[] = request.getParameterValues("id");
        //navega no vetor que contem os id checados
        for(String id:ids){
            //Cria um usuario para cada id selecionado
            Usuario usuExc = new Usuario();
            usuExc.setId(Integer.parseInt(id));
            //exclui usuario selecionado
            usuDao.excluir(usuExc);
        }
        //carregando a lista após exclusão
        response.sendRedirect("usucontroller.do?acao=lis");
    }
}

```

Alterando Registro

Para realizarmos a alteração de um registro no banco vamos criar em um link para cada usuário listado na página *listausuarios.jsp*.

Este *link* fará uma requisição pelo método *get* ao nosso *servlet* que o identificará pelo parâmetro *acao* que definiremos como *alt* (de alteração) e nos redirecionará para uma página contendo um formulário para apresentação dos dados a serem alterados.

Neste caso vamos aproveitar o mesmo formulário *frmusuario.html* que fizemos para cadastra ou alterar para apresentar os dados do usuário.

Adicionando um link na lista para alterar

```

<% for (Usuario u: lista) {%>
<tr>
    <td> <%= u.getId() %> </td>
    <td> <%= u.getNome() %> </td>
    <td> <%= u.getLogin() %> </td>
    <td> <%= u.getSenha() %> </td>
    <td> <input type="checkbox" name="id" value="<%= u.getId() %>"

    <a href="usucontroller.do?acao=alt&id=<%= u.getId() %>">alterar</a>

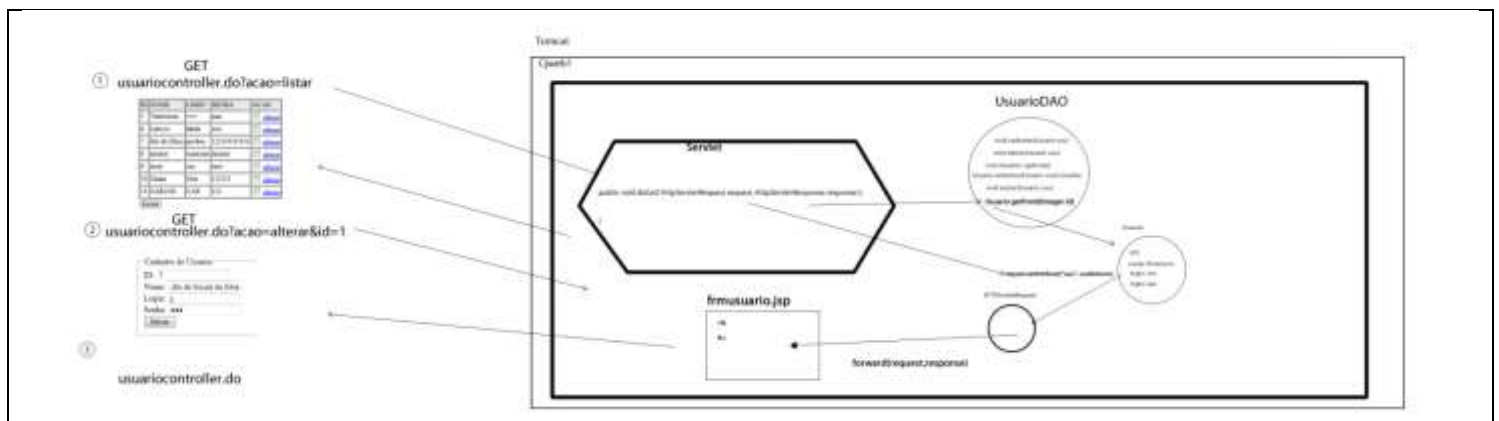
    </td>
</tr>
<% } %>

```


ID	NOME	LOGIN	SENHA	ACAO
5	Virmerson	vvv	aaa	<input type="checkbox"/> alterar
6	Larisce	lalala	eee	<input type="checkbox"/> alterar
7	Jão de Deus	jaodeu	1234545454	<input type="checkbox"/> alterar
8	tatatat	tatatatat	tatatat	<input type="checkbox"/> alterar
9	tetet	ete	rere	<input type="checkbox"/> alterar
10	Maria	Mar	123333	<input type="checkbox"/> alterar
11	CARLOS	CAR	123	<input type="checkbox"/> alterar

Carregando a lista

<http://localhost:8080/cjweb1/usucontroller.do?acao=lis>



- 1) Ao Clicar no link *alterar* a cliente faz uma requisição para o *Servlet* passando os parâmetros *acao* igual a "alt" e "id" do usuário da lista.
- 2) O *Servlet* pede para o *UsuarioDAO* um objeto do tipo *Usuario* com o id correspondente
- 3) O *Servlet* encaminha o objeto usuário para o formulário de alteração
- 4) O Formulário de alteração imprime os dados do usuário dentro dos campos de *input* para serem alterados.

Alterando o método doGet

Implementando o método *doGet* para tratar o parâmetro quando a "acao" for igual a "alt".

www.hightechcursos.com.br - contato@hightechcursos.com.br (67) 3387-2941

```

protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {

    // Captura a acao
    String acao = request.getParameter("acao");
    // Testa se acao foi informada
    if (acao != null && !acao.isEmpty()) {
        // Cria objeto dao para operacao no banco de dados
        UsuarioDAO usuarioDAO = new UsuarioDAO();
        // testa se a acao é de exclusao
        if (acao.equals("exc")) {
            // Caputara o id a ser excluido
            String id = request.getParameter("id");
            // valida se o id for informado
            if (id != null && !id.isEmpty()) {
                Usuario usuExc = new Usuario();
                usuExc.setId(Integer.parseInt(id));
                usuarioDAO.excluir(usuExc);
                response.getWriter().print("Excluido com sucesso!");
            }
        }
        else if (acao.equals("lis")) {
            //Captura a lista de usuários do banco
            List<Usuario> lista = usuarioDAO.buscaTodos();
            //Adidicio a lista no request
            request.setAttribute("lista", lista);
            //encaminha o request e o response para o JSP
            request.getRequestDispatcher("listausuarios.jsp").forward(request,
response);

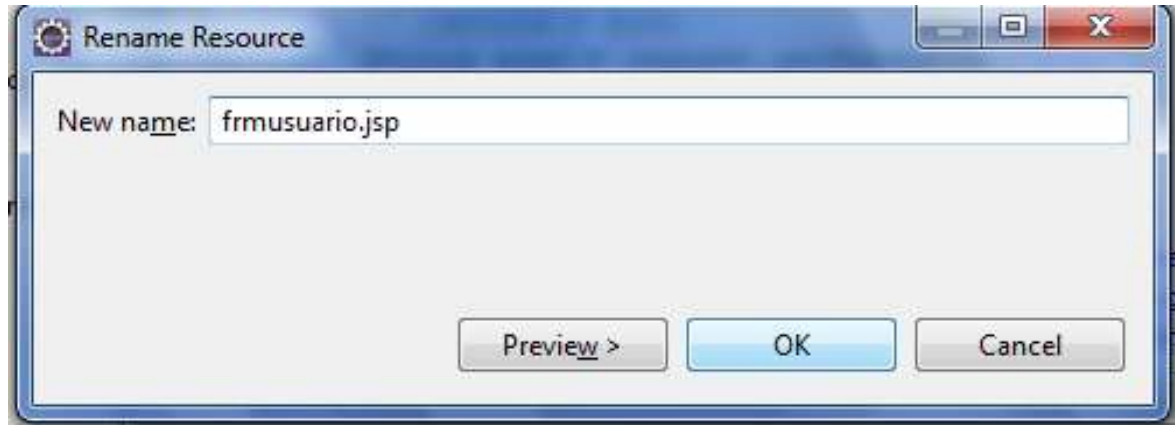
        }
        else if (acao.equals("alt")) {
            //Captura o parametro da tela
            String id = request.getParameter("id");
            //Buscando do banco o usuario pelo Id vindo do parametro
            Usuario usuAlt = usuarioDAO.buscaPorId(Integer.parseInt(id));
            //Adicionando objeto usuario no request pra encaminhar pra tela
            request.setAttribute("usu", usuAlt);
            //encaminhando o usuario para a tela de alteracao
            request.getRequestDispatcher("frmusuario.jsp").forward(request, response);
        }
    }
}
}

```

Tela de alteração

Alterando o *frmusuario.html* para *frmusuario.jsp*.

Selecione o arquivo *frmusuario.html* e pressione F2 para renomear.



```
<%
    Usuario usuAlt = (Usuario)request.getAttribute("usu");
%>
<form method="post" action="usucontroller.do">
    <input type="hidden" name="acao" value="salvar">
    ID: <input size="5" type="text" name="id" value="<%=usuAlt.getId() %>">
    Nome: <input type="text" name="nome" value="<%=usuAlt.getNome() %>">
    Login: <input type="text" name="login" value="<%=usuAlt.getLogin() %>">
    Senha: <input type="password" name="senha"
value="<%=usuAlt.getSenha() %>">
        <input type="submit" value="SALVAR">
</form>
```

- 1) Captura o atributo "usu" que foi definido no Servlet UsuarioController e atribuído ao request.
- 2) Preenche o formulário com os dados do objeto Usuario.
- 3) Abra a lista e clique no link alterar para fazer uma requisição com esta:
<http://localhost:8080/cjweb1/usucontroller.do?acao=alt&id=11>
- 4) A tela de alteração é carregada

ID: Nome: Login: Senha:

- 5) Clique no salvar

Criando ação Cad no doGet

```
...  
else if (acao.equals("cad")) {  
  
    Usuario usuCad =new Usuario();  
    usuCad.setId(0);  
    usuCad.setNome("");  
    usuCad.setLogin("");  
    usuCad.setSenha("");  
    //Adicionando objeto usuario no request pra encaminha pra tela  
    request.setAttribute("usu", usuCad);  
    //encaminhando o usuario para a tela de alteracao  
    request.getRequestDispatcher("frmusuario.jsp").forward(request,  
response);  
}
```

- 1) Criamos um objeto usuário novo com as propriedades todas com valores iniciais vazios para não ficar null preenchido nos campos.
- 2) Definimos um id igual à zero para representar que é para cadastrar um novo usuário e quando o id for diferente de zero significará que o usuário deverá ser alterado.

Chamando: <http://localhost:8080/cjweb1/usucontroller.do?acao=cad>

Alterando teste para na ação “salvar” no método doPost

De

```
// Testando se o id é diferente de null e também não é vazio  
if (id != null && !id.isEmpty()) {  
    // Se o id fosse vazio ou null então daria Exception do Parse  
    usuario.setId(Integer.parseInt(id));  
}
```

Para

```
// Testando se o id é diferente de null e também não é vazio  
if (id != "0" && !id.isEmpty()) {  
    // Se o id fosse vazio ou null então daria Exception do Parse  
    usuario.setId(Integer.parseInt(id));  
}
```

Corrigindo o teste, pois agora estamos definindo um valor inicial para o campo *id* igual à zero. Como favor deixar o campo *id* oculto, então sempre inicial começará com zero.

Mensagem Java Script no Browser

Para nosso sistema dar mensagens mais elegantes ao usuário, vamos enviar ao browser um comando JavaScript para dar mensagens de alerta após realizar a operação *salvar*.

Substituir do método doPost na ação “salvar”

De:

```
...  
// Salvando no banco de dados  
    usuDao.salvar(usuario);  
// Mensagem no Browser  
    response.getWriter().print("Salvo com Sucesso");  
...
```

Comando JavaScript

Este comando exibe uma mensagem de alerta e após o clique no botão “OK” da janela o seguinte de requisição é disparado, onde será solicitada a página da lista.

```
<script>  
    alert('Salvo com sucesso!');  
    location.href='usucontroller.do?acao=lis';  
</script>
```

Para:

Colocando o comando JavaScript dentro de uma String no Servlet para enviar para o browser imprimir.

```
...  
usuDao.salvar(usuario);  
  
String mensagem = "<script>" +  
    "alert('Salvo com sucesso!'); " +  
    "location.href='usucontroller.do?acao=lis';" +  
    "</script>";  
  
    response.getWriter().print(mensagem);  
...
```

Com JavaScript podemos exibir uma mensagem em forma de alerta e após fazer um requisição ao Servlet para mostrar a lista novamente.

Validando formulário com JavaScript

Adicione este javascript entre as tags <head> e </head> no formulário frmusuario.jsp

Adicione um nome ao formulário para que ele possa ser acesso via Javascript.

```
<script>

//Função para validação
//Retorna True quando está tudo certo e false quando o campo não foi preenchido
function validar(){
    //Captura o campo o objeto input com a propriedade "name" igual a "nome"
    camponome = document.frmusu.nome;
    //Valida se o propriedade value do campo está vazia
    if(camponome.value==""){
        //Exibe mensagem de alerta
        alert("o Campo nome é obrigatório");
        //posiciona o foco no campo
        camponome.focus();
        //retorna falso
        return false;
    }
    //retorna true quando não passa no if
    return true;
}

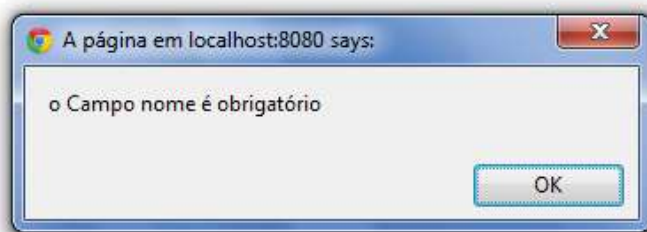
</script>
```

Adicione ao evento "onsubmit" do formulário a chamada a função "javascript" de "validar()"

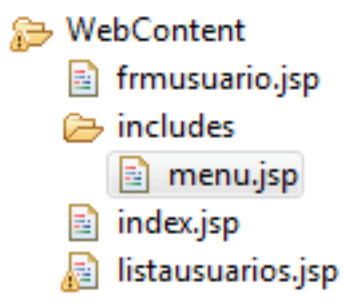
```
<form name="frmusu" method="post" action="usucontroller.do"
onsubmit="return validar()">
```

Este javascript verifica se o campo nome está preenchido e se não estiver então imprime uma mensagem de alerta ao usuário, após coloca o foco no campo obrigatório e retorna false para que os dados do formulário não sejam submetidos ao servidor.

[Página inicial](#) [Usuários](#) [Sair](#)
ID: 5 Nome: Login: vvv Senha: SALVAR



Criando um Menu de navegação.



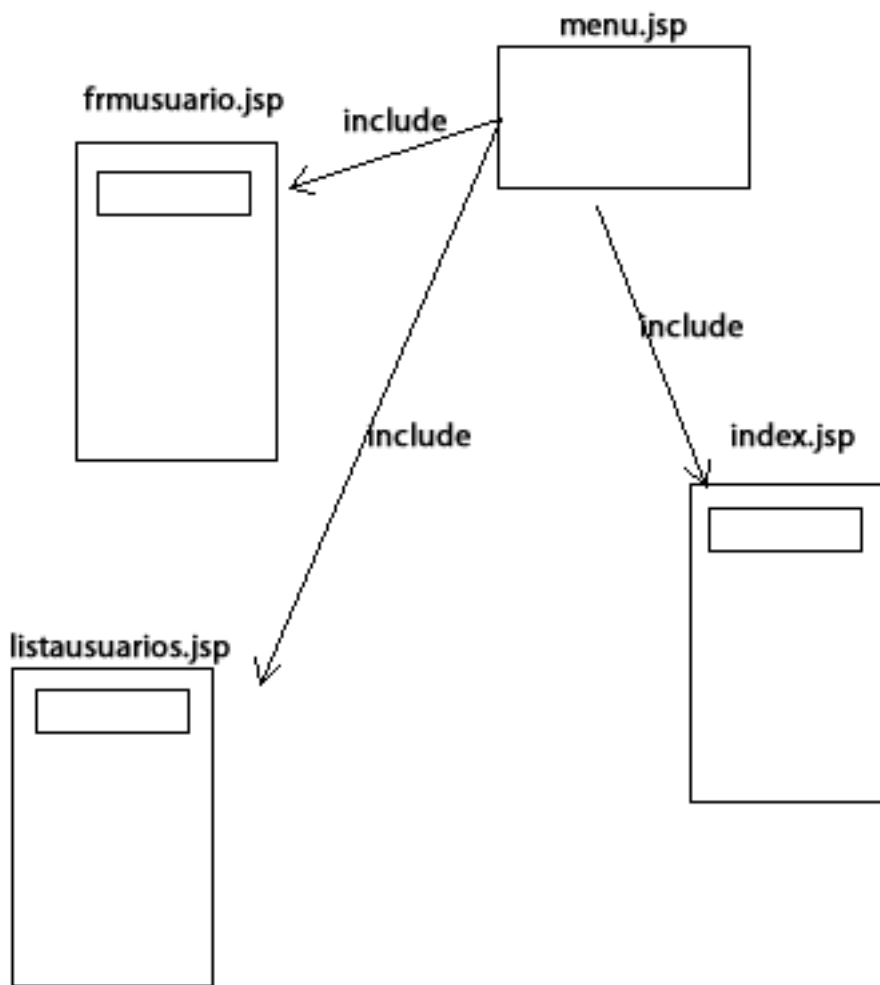
- 1) Criamos uma pasta includes para separar os arquivos de inclusão.
- 2) Criamos uma arquivo *menu.jsp*.

```
<a href="usucontroller.do?acao=inicio"> Página inicial</a>  
<a href="usucontroller.do?acao=lis"> Usuários </a>  
<a href="usucontroller.do?acao=sair"> Sair </a>
```

As ações “inicio” e “sair” ainda não foram programadas, mas já estão lá para as futuras etapas.

Colocando o comando de inclusão na *listausuarios.jsp* e *frmusuario.jsp*.

```
<body>  
  
<%@include file="includes/menu.jsp" %>  
  
...
```



Adicionando o include na listausuarios.jsp

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Lista de Usuários</title>
</head>
<body>

<%@include file="includes/menu.jsp" %>

<%
    //captuando a lista do request
```

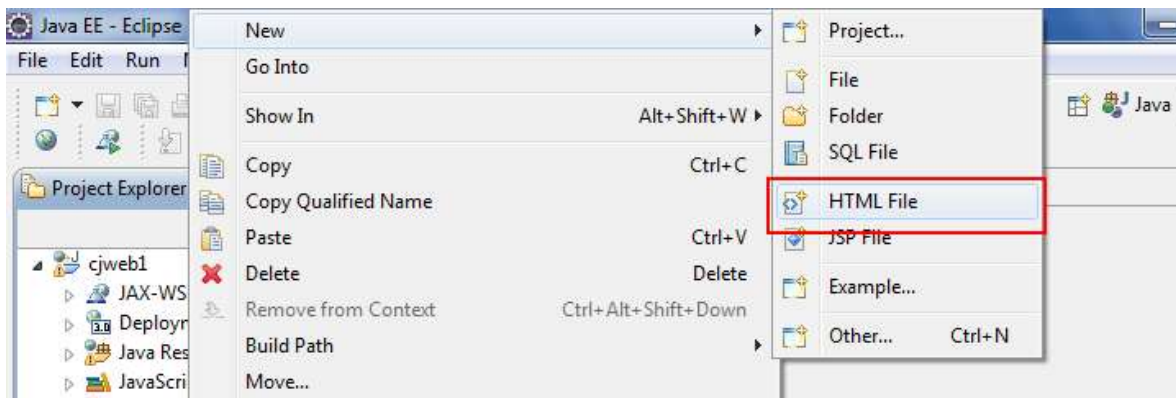


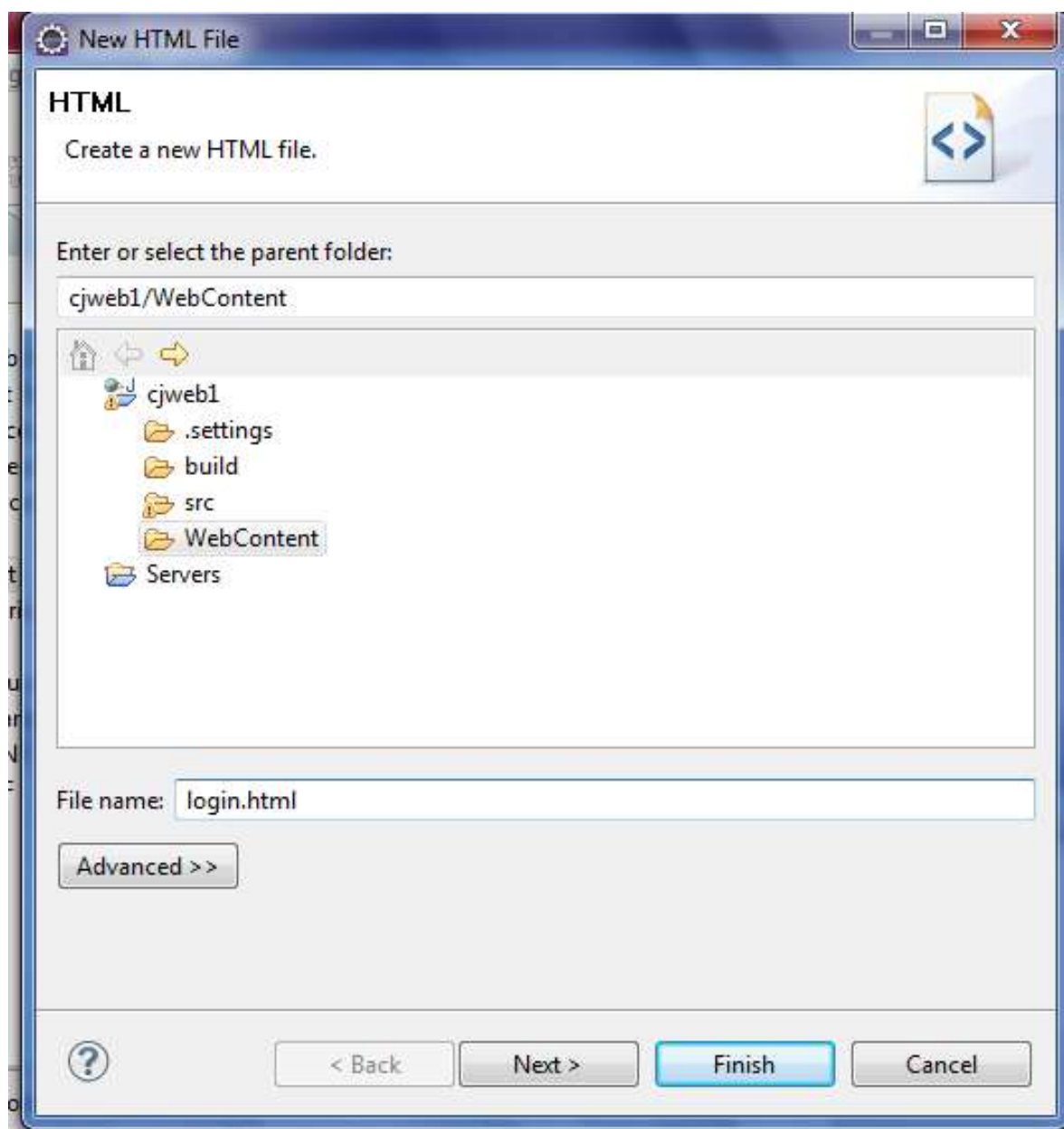
```
List<Usuario> lista = (List<Usuario>) request.getAttribute("lista");  
%>  
...
```

Include no frmusuario.jsp

```
...  
<body>  
<%@include file="includes/menu.jsp" %>  
<%  
    Usuario usuAlt = (Usuario) request.getAttribute("usu");  
%>  
  
    <form name="frmusu"  
        method="post"  
        action="usucontroller.do"  
        onsubmit="return validar()">  
...  
.....
```

Criando um Formulário para Autenticação – login.html





```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Autenticação</title>
</head>
<body>

<form method="post" action="autenticador.do">
<fieldset>
    <legend>Autenticação</legend>
```

```
        Login: <input type="text" name="login">  
        Senha: <input type="password" name="senha">  
        <input type="submit" value="Autenticar">  
    </fieldset>  
</form>  
  
</body>  
</html>
```

HTTPSession

Para identificarmos o usuário autenticado por meio das requisições será necessário fazermos o controle por meio de sessão.

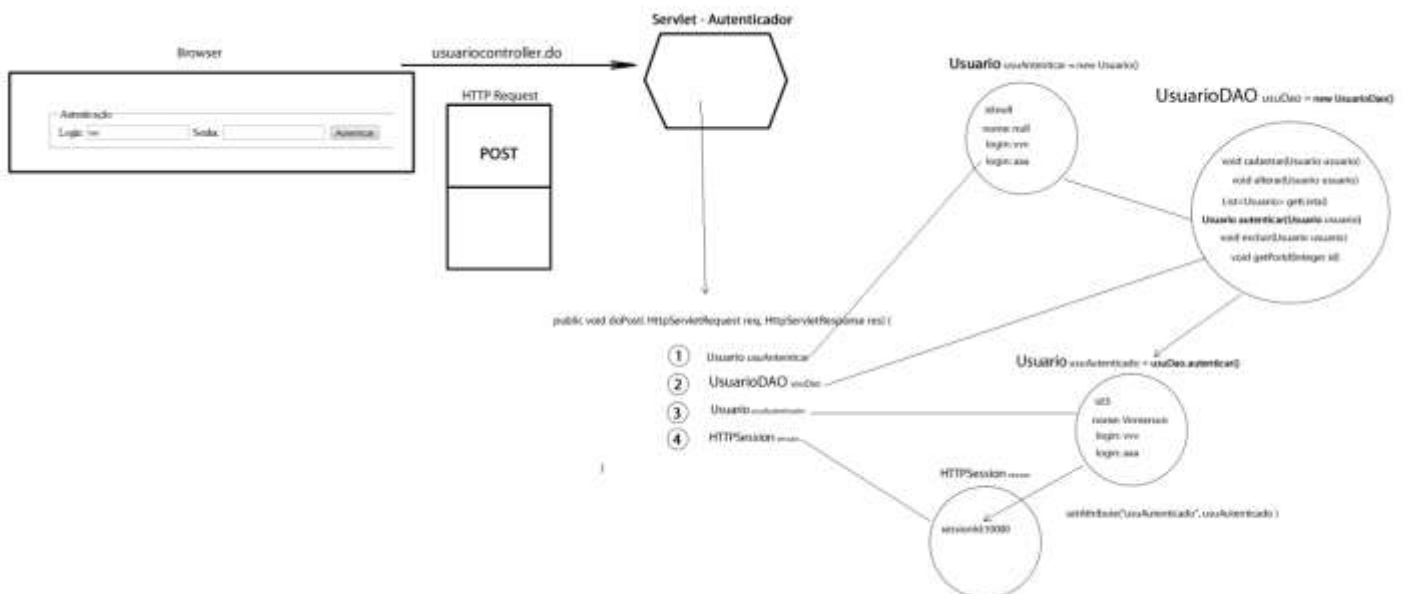
Alguns métodos importantes da Interface `HttpService`.

setMaxInactiveInterval (interval int) - Especifica o tempo, em segundos, entre solicitações do cliente antes do servlet container irá invalidar esta sessão.

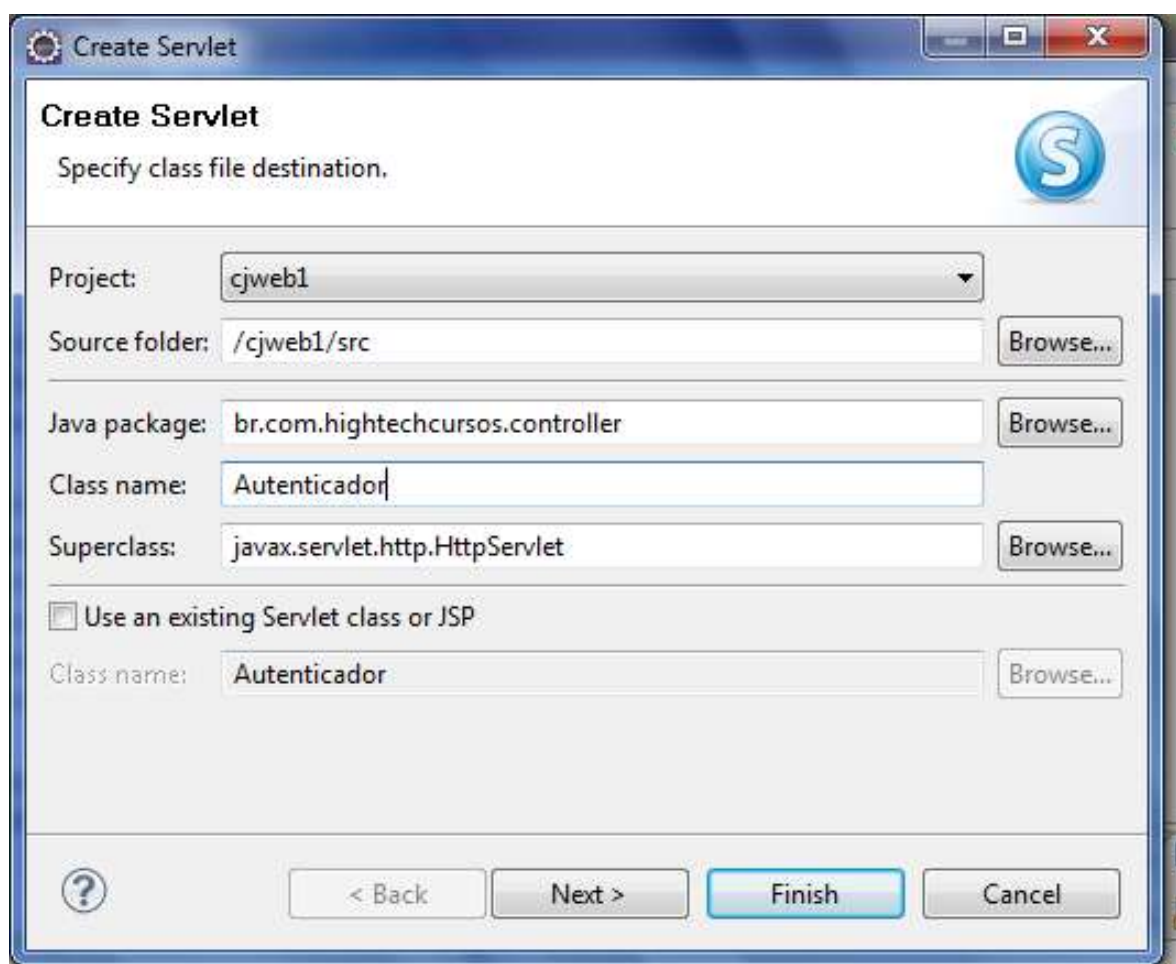
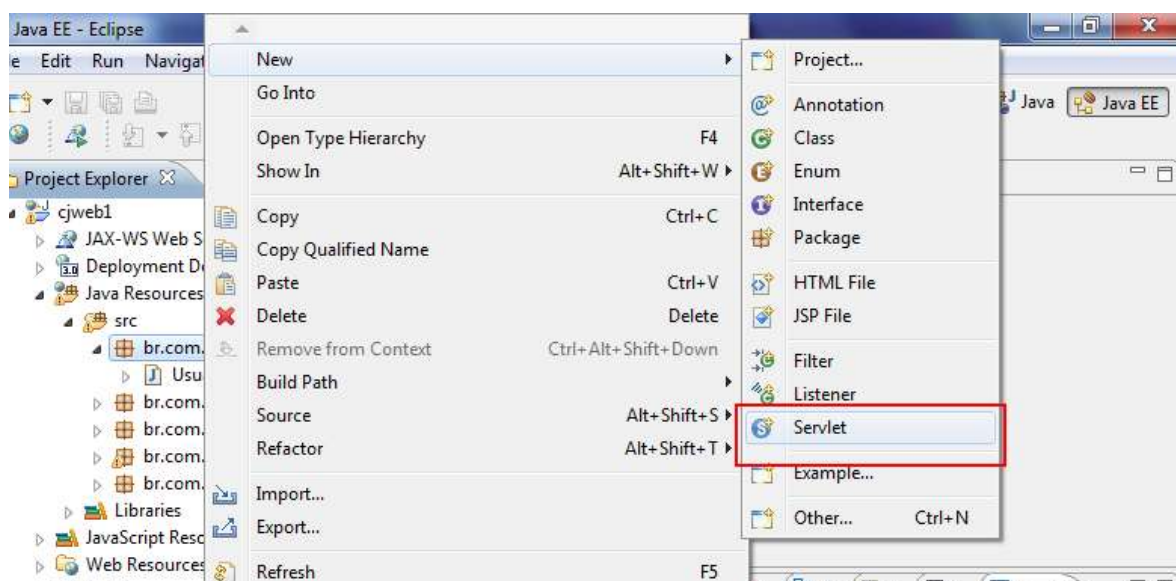
setAttribute (java.lang.String name, java.lang.Object valor) - Vincula um objeto para esta sessão, usando o nome especificado.


getAttribute (java.lang.String nome) - Retorna o objeto vinculado com o nome especificado nesta sessão, ou nulo se nenhum objeto está ligada com o nome.

invalidate () - Invalida a sessão, em seguida, libera todos os objetos ligados a ela.



Criando o Servlet Autenticador



 **Create Servlet**

Enter servlet deployment descriptor specific information.

Name:

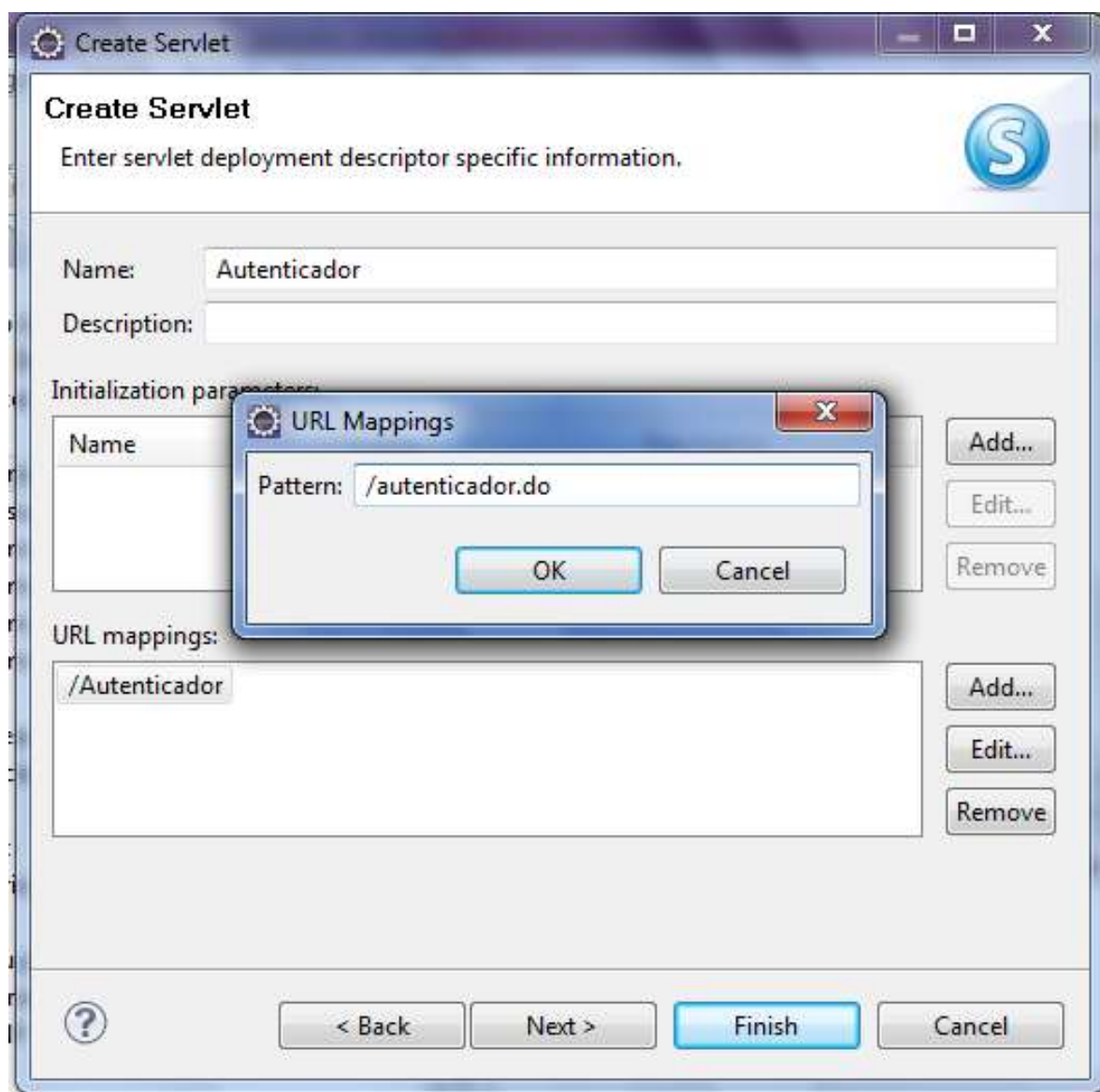
Description:

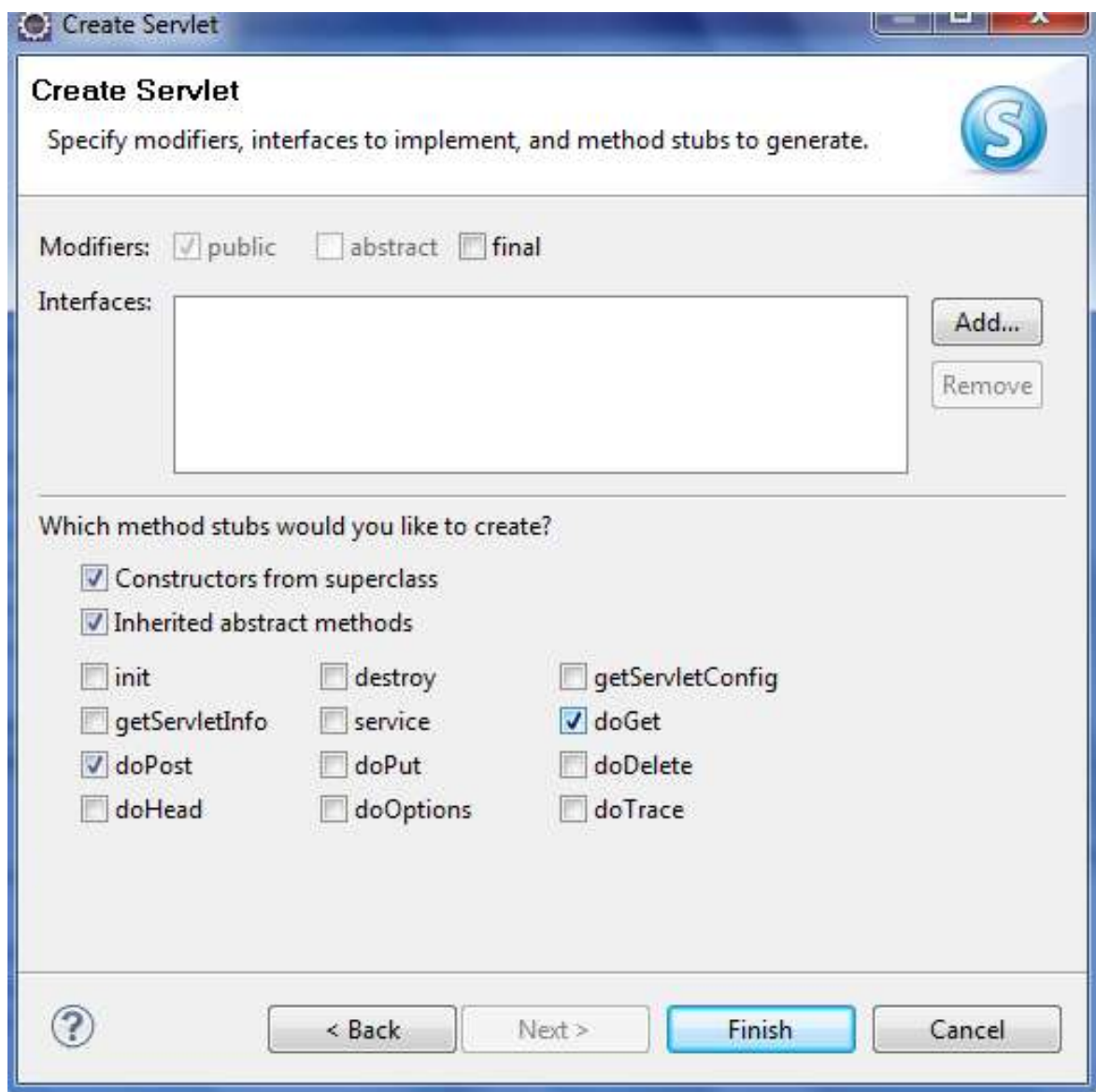
Initialization parameters:

Name	Value	Description

URL mappings:

<input type="text" value="/Autenticador"/>
--





Este Servlet recebe os dados do formulário de autenticação e faz o processo identificação do usuário e criação de Sessão.

- 1) Capturar os dados
- 2) Criar objeto **Usuario** e adicionar os dados
- 3) Criar objeto **UsuarioDAO** e Autenticar
- 4) Verificar se usuário foi encontrado
- 5) Criar Sessão
- 6) Adicionar objeto como atributo da sessão
- 7) Encaminhar para a tela de bem vindo

```
package br.com.hightechcursos.controller;
```



```

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

import br.com.hightechcursos.entidades.Usuario;
import br.com.hightechcursos.jdbc.UsuarioDAO;

/**
 * Servlet implementation class Autenticador
 */
@WebServlet("/autenticador.do")
public class Autenticador extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        //1) Capturar os dados
        String login = request.getParameter("login");
        String senha = request.getParameter("senha");
        //2) Criar objeto usuário e adicionar os dados
        Usuario usu = new Usuario();
        usu.setLogin(login);
        usu.setSenha(senha);
        //3) Criar objeto UsuarioDAO e Autenticar
        UsuarioDAO usuarioDAO = new UsuarioDAO();
        Usuario usuAutenticado = usuarioDAO.autenticar(usu);
        //4) Verificar se usuário foi encontrado
        if(usuAutenticado!=null){
            //5) Criar Sessão
            HttpSession sessao= request.getSession();
            //6) Adicionar objeto como atributo da sessão
            sessao.setAttribute("usuAutenticado", usuAutenticado);
            //Definindo um tempo para a Sessão expirar
            sessao.setMaxInactiveInterval(3000);
            //7) Encaminhar para a tela de bem vindo
            request.getRequestDispatcher("index.jsp").forward(request,
response);
        }
    }
}

```

Alterando a página de bem vindo no arquivo web.xml

Vamos definir a tela de login como senha a página de bem vindo do nosso sistema.

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"

```

```
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">
  <display-name>cjweb1</display-name>
  <welcome-file-list>
    <welcome-file>login.html</welcome-file>
  </welcome-file-list>
</web-app>
```

Acessando no navegador:

<http://localhost:8080/cjweb1/>

Lembre-se de dar o Start no tomcat.

Autenticação		
Login:	<input type="text"/>	Senha: <input type="password"/>
		<input type="button" value="Autenticar"/>

Página de Bem vindo do sistema (index.jsp)

Neste tela capturamos a sessão do usuário que passou pelo Servlet Auteticador e exibimos os dados do usuário que está como atributo da sessão.

```
<%@page import="br.com.hightechcursos.entidades.Usuario"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%@include file="includes/menu.jsp" %>
<%
//Captura usuário da sessão
Usuario usuAutenticado = (Usuario) session.getAttribute("usuAutenticado");
%>
<p>
Seja bem vindo <b><%= usuAutenticado.getNome() + " id:" +
usuAutenticado.getId() %></b>
</p>
</body>
</html>
```

Autenticando o usuário

www.hightechcursos.com.br - contato@hightechcursos.com.br (67) 3387-2941

[Página inicial](#) [Usuários](#) [Sair](#)

Seja bem vindo **Virmerson id:5**

Exibindo mensagem quando o usuário não foi encontrado

Adicionado um comando `else` para imprimir uma mensagem quando o usuário não for encontrado.

Se o usuário foi encontrado então `usuAutenticado!=null` e será redirecionado para a tela de bem vindo ao sistema (`index.jsp`) e quando o usuário não for encontrado então será entregue para o navegador um comando `javascript` contendo um mensagem `JavaScript` e um redirecionamento para a tela de `login.html`.

```
if (usuAutenticado!=null) {
    //5) Criar Sessão
    HttpSession sessao= request.getSession();
    //6) Adicionar objeto como atributo da sessão
    sessao.setAttribute("usuAutenticado", usuAutenticado);
    //Definindo um tempo para a Sessão expirar
    sessao.setMaxInactiveInterval(3000);
    //7) Encaminhar para a tela de bem vindo
    request.getRequestDispatcher("admin/index.jsp").forward(request,
response);
} else {
    response.getWriter().print("<script> alert('Usuário não
encontrado!'); "+
"location.href='login.html' </script>");
}
```

Criando o método para sair (Logout)

Vamos adotar no servlet o método `doPost` para fazer o Login e o método `doGet` para fazer o Logout.

Toda vez que o usuário quiser autenticar então será chamado um formulário de `login.html` que fará uma requisição para o Servlet Autenticador (`autenticador.do`) pelo método `post`.

E toda vez que o usuário quiser sair então será chamado por meio do link no menu o mesmo servlet pelo método get, onde será processado o doGet que tem o comando para invalidar a sessão caso ela ainda não tenha sido expirada.

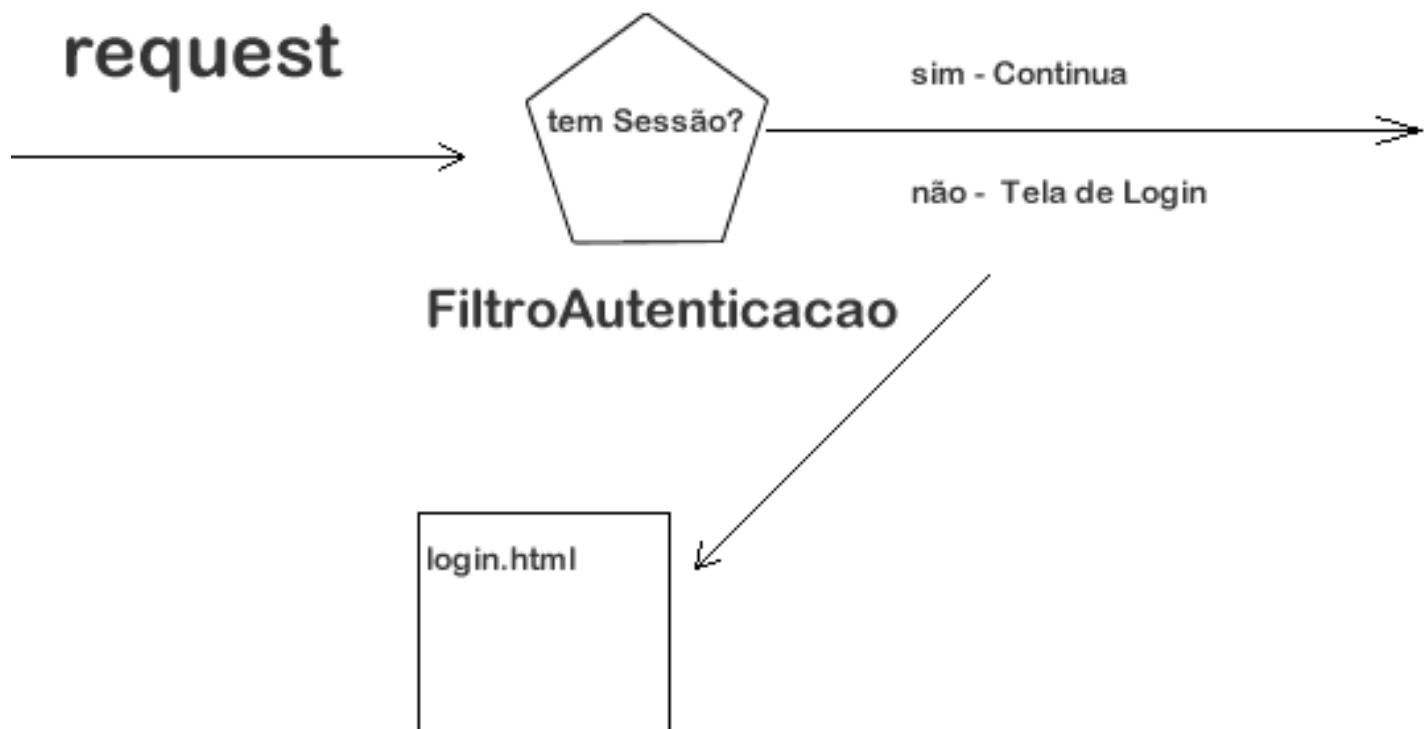
```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    //Acessando a sessao
    HttpSession sessao = request.getSession(false);
    //Se a sessão ainda não foi expirada
    if(sessao!=null){
        //invalida a sessão
        sessao.invalidate();
    }
    //Redirecionando para tela de login
    response.sendRedirect("login.html");
}
```

Criando o link sair no includes/menu.jsp

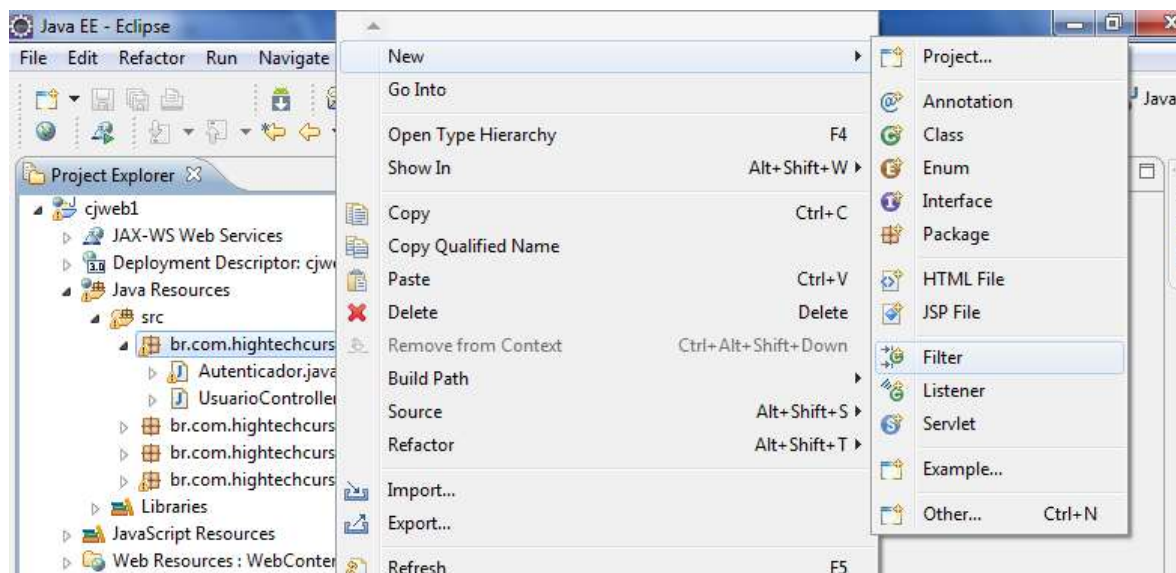
```
<a href="usucontroller.do?acao=inicio"> Página inicial</a>
<a href="usucontroller.do?acao=lis"> Usuários </a>
<a href="autenticador.do"> Sair </a>
```

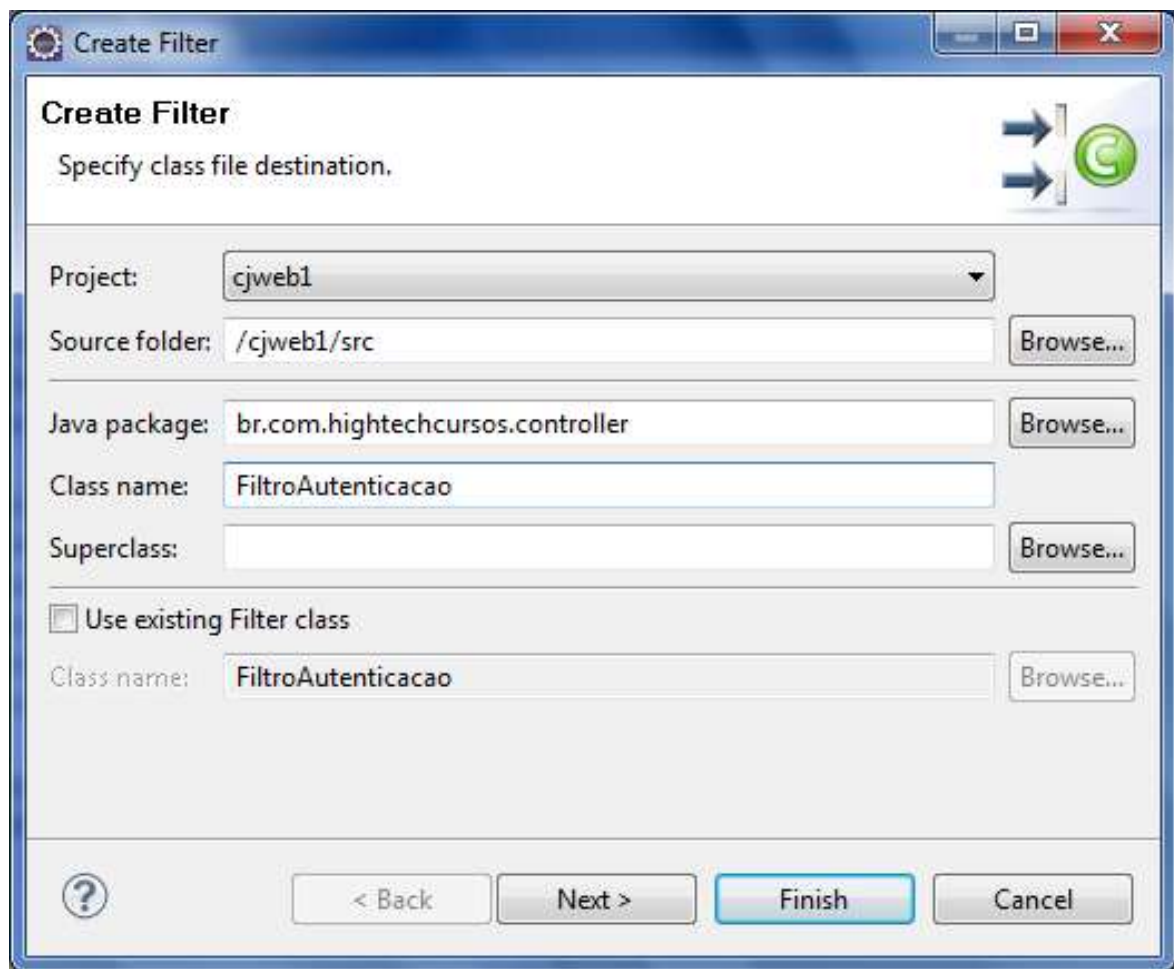
Criando um Filtro

O Filtro funcionará com um intercessor de todas as requisições e encaminhamentos para os servlets ou jsp. Para toda requisição o filtro entrará em ação. A única página que ele deixe acessar sem estar autenticado é a página de autenticação *login.html*.

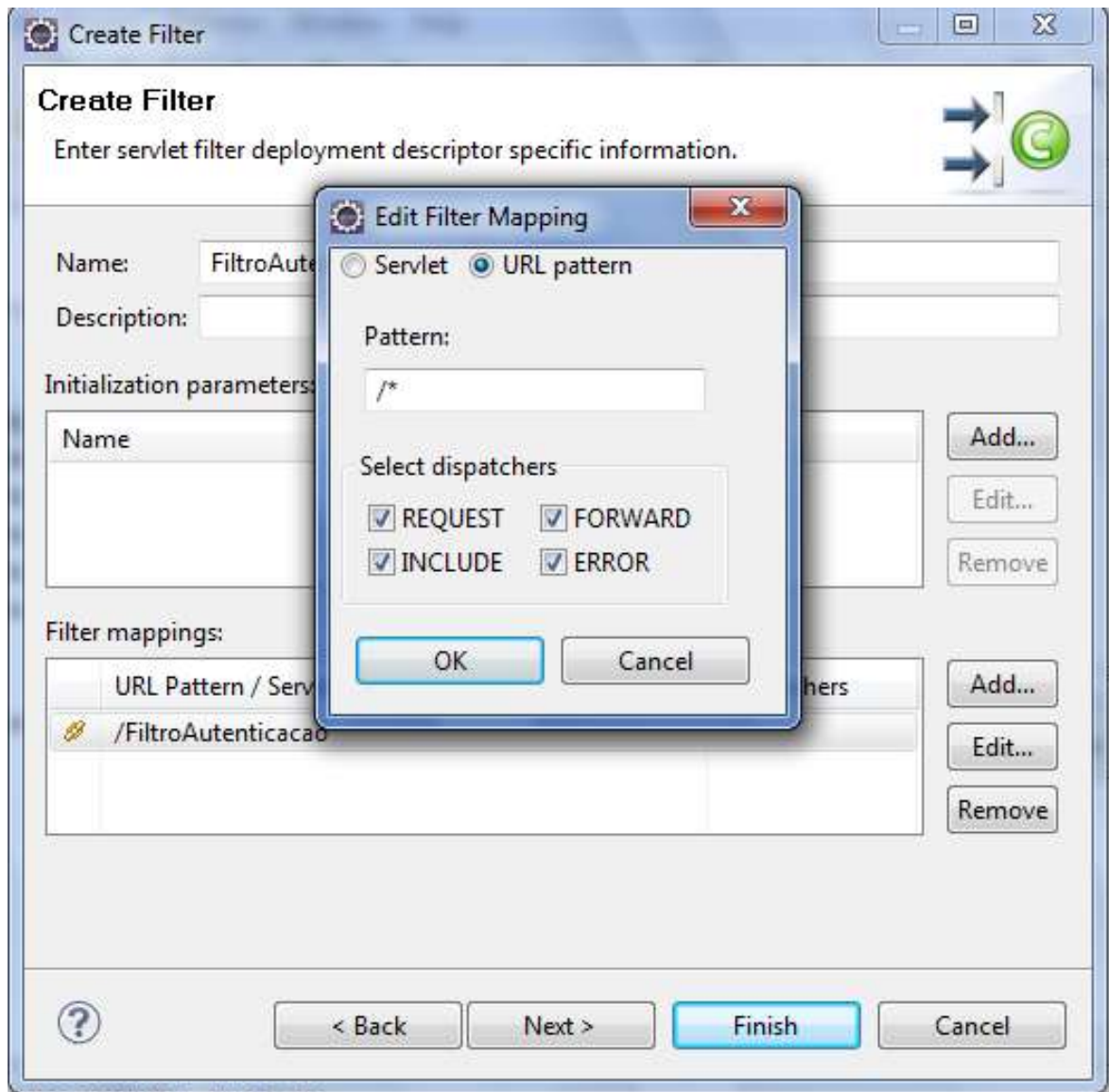


Seu papel será checar se o usuário está autenticado para acessar os arquivos restritos;





Definindo o filtro para todas as requisições do sistema



Filtro com nome FiltroAutenticacao

```
package br.com.hightechcursos.controller;

import java.io.IOException;
import javax.servlet.DispatcherType;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

www.hightechcursos.com.br - contato@hightechcursos.com.br (67) 3387-2941

```

import javax.servlet.http.HttpSession;

@WebFilter(dispatcherTypes = {
    DispatcherType.REQUEST,
    DispatcherType.FORWARD
}, urlPatterns = { "/" })
public class FiltroAutenticacao implements Filter {

    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
    throws IOException, ServletException {
        //Casting do HttpServlet Request
        HttpServletRequest httpRequest = (HttpServletRequest) request;

        String url = httpRequest.getRequestURI();
        //Capturando Sessao
        HttpSession sessao = httpRequest.getSession();

        //Está logado?
        if (sessao.getAttribute("usuLogado")!=null || url.lastIndexOf("login.html")>-1 ||
        url.lastIndexOf("autenticador.do") >-1 ){
            chain.doFilter(request, response); //Permite o fluxo da requisicao
        }else{
            //redireciona para login
            ((HttpServletResponse) response).sendRedirect("login.html");
        }

    }

    @Override
    public void destroy() {

    }

    @Override
    public void init(FilterConfig arg0) throws ServletException {

    }

}

```

O filtro será invocado para todas as requisições.

O Filtro funciona como um guarda, que se será acionado toda vez que o usuário tentar acessar o sistema pelas portas do fundo, ou seja, quanto tentar acessar uma pagina sem ter passado pela porta de entrada que é a tela de autenticação *login.html*.

Ex.: tente acessar uma página qualquer sem ter autenticado como:

<http://localhost:8080/cjweb1/usucontroller.do?acao=lis>

Para cada requisição ou redirecionamento para a qualquer recurso de nossa aplicação como *Servlet*, *JSP* ou *HTML* o filtro é acionado e faz a verificação se a sessão foi criada e se não for então ele redireciona o usuário para a porta de entrada, ou seja, a tela de login.html.

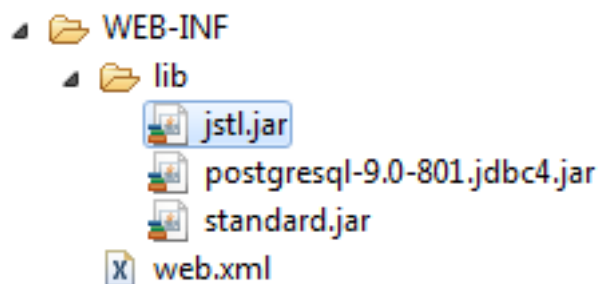
Lista de Usuários em JSTL e EL

O JavaServer Pages Standard Tag Library (JSTL) encapsula como *tags* simples a funcionalidade do núcleo comum para muitas aplicações web. JSTL tem suporte para tarefas comuns e estruturais, tais como iteração e condicionais, *tags* para manipulação de documentos XML, tags de internacionalização e tags SQL. Ele também fornece uma estrutura para integrar existentes tags personalizadas com as tags JSTL.

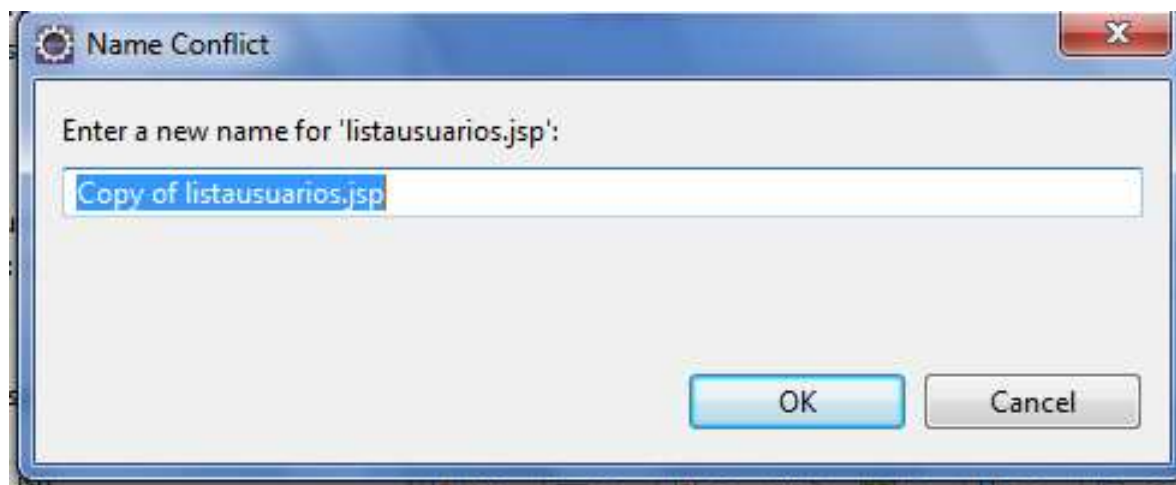
Baixe as bibliotecas do JSTL no site da High Tech:

www.hightechcursos.com.br/downloads/cjweb1/lib.rar

Adicione as bibliotecas do JSTL (jstl.jar e standard.jar) na pasta lib.



Faça uma cópia de backup da *listausuarios.jsp* (Copy listausuarios.jsp) e vamos trabalhar no arquivo *listausuarios.jsp* mesmo para não precisar modificar nada no *Servlet UsuarioController*.



A primeira coisa a fazer agora é definir a diretiva `<%@taglib %>` para o JSTL.

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

Em seguida vamos colocar os comando `<c:import>` e `<c:forEach>`

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Lista de Usuários - JSTL</title>
</head>
<body>
<c:import url="includes/menu.jsp"></c:import>
```

```

<form action="usucontroller.do" method="post">
<input type="hidden" name="acao" value="exc">
<table border="1">
  <tr bgcolor="#eaeaea">
    <td> ID </td>
    <td> NOME </td>
    <td> LOGIN </td>
    <td> SENHA </td>
    <td> ACAA </td>
  </tr>

  <c:forEach items="${requestScope.lista}" var="u" >
    <tr>
      <td> ${u.id} </td>
      <td> ${u.nome} </td>
      <td> ${u.login} </td>
      <td> ${u.senha} </td>
      <td> <input type="checkbox" name="id" value="${u.id}" >
        <a href="usucontroller.do?acao=alt&id=${u.id}">alterar</a>
      </td>
    </tr>
  </c:forEach>

</table>
<input type="submit" value="Excluir">
</form>
</body>
</html>

```

Formulário de Cadastro em JSTL e EL

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Formulário de Cadastro JSTL</title>
<script>
  function validar(){
    camponome = document.frmusu.nome;

    if(camponome.value=="") {
      alert("o Campo nome é obrigatório");
      camponome.focus();
      return false;
    }
    return true;
  }
</script>
</head>
<body>
<c:import url="includes/menu.jsp"></c:import>
  <form name="frmusu"

```

```
method="post"
action="usucontroller.do"
onsubmit="return validar()">

    <input type="hidden" name="acao" value="salvar">
    ID: <input size="5" type="text" name="id" value="\${requestScope.usu.id}">
    Nome: <input type="text" name="nome" value="\${requestScope.usu.nome}">
    Login: <input type="text" name="login" value="\${requestScope.usu.login}">
    Senha: <input type="password" name="senha" value="\${requestScope.usu.senha}">

    <input type="submit" value="SALVAR">
</form>
</body>
</html>
```

Comandos Básicos JSTL

Core Tags:

Estas tags fornece a funcionalidade do núcleo que uma linguagem de programação é suposto oferecer, como iteração, condição e etc.

1). <c:out>

Esta tag é usada para avaliar uma expressão e a saída do resultado para o objeto JspWriter atual.

Sintaxe:

```
<C: out = valor de "valor" [EscapeXML = "{true | false}"] [Default = "defaultValue"] />
```

2). <c:forEach>

Esta tag fornece funcionalidade para iterar sobre uma matriz ou coleção sem usar o código java.

3) <c:if>

Isto é usado para ramificação condicional dentro da página jsp com base em alguns testes.

4) <c:set>

Este tag é usado para definir atributos dentro escopos especificados em páginas jsp

5) *<c:remove>*

Esta tag é usada para remover uma variável de escopo.

6) *<c:url>*

Esta tag é usada para o gerenciamento de sessão e reescrita de URL.

7) *<c:import>*

Isto é usado para incluir recursos externos dentro da página jsp.

8) *<c:choose> <c:when> e <c:otherwise>*

Estas tags são usadas para ramificação condicional, que é fornecido pela chaveexposição do caso na maioria das linguagens de programação.

<c:choose> é usado para realizar execução condicional exclusiva.

Sintaxe:

```
<c:choose>
  .. Corpo ..... (<when> subtags <otherwise>)
</ C: choose>
```

<c:when> representa uma alternativa dentro de uma ação *<c:choose>*.

Sintaxe:

```
<c:when test="testCondition">
  .... corpo
</ C: quando>
```

<c:otherwise> representa a última alternativa dentro da condição *<c:choose>*.

9) *<c:catch>*

Esta tag fornecer um try ... catch como mecanismo dentro de páginas jsp sem usar scriptlets.

10) *<c:redirect>*

Isto proporciona mesma funcionalidade que o método `sendRedirect ()` da interface `HttpServletRequest`. Ele é usado para redirecionar o controle para outras páginas dentro do aplicativo web.

11) *<c:param>*

Esta tag é usada para adicionar parâmetros de solicitação para uma URL. Ele também é usado como uma marca aninhada, dentro das tags `<c:url>` `<c:import>` e `<c:redirect>`.

12) *<c:forEachToken>*

Essa marca quando fornecido com delimitadores pode ser usado para iterar sobre tokens. Sua funcionalidade é semelhante ao de `StringTokenizer`.

Tags de formatação

As tags de formatação são usados para internacionalização e localização das páginas jsp. Internacionalização fornece suporte para várias línguas e formatos de dados. Por aplicações web de localização torna-se capaz de suportar determinadas regiões ou localidade.

1) *<fmt:setLocale>*

Esta tag é usada para definir a localidade padrão dentro de um escopo JSP específicos.

```
<fmt:setLocale valor = "en_US" />
```

2) *<fmt:timeZone>*

Este tag é usado para especificar o fuso horário em que a informação do tempo é para ser formatado ou analisado no seu conteúdo corpo.

3) *<fmt:formatNumber>*

Este tag é usado para formatar um valor numérico de uma forma sensíveis à localidade ou personalizado como um número, moeda e etc.

4) *<fmt:formatDate>*

Esta tag é usada para formato de data e hora de acordo com a localidade.

Palavras-chave SQL:

As tags são usados para realizar tarefas comuns de banco de dados relacionados, tais como a inserção, seleção, exclusão, update a partir das páginas jsp, sem o uso de qualquer código java.

1) *<sql:query>*

Este tag é usado para consultar bancos de dados.

2) *<sql:update>*

Esta tag é usada para executar SQL INSERT, UPDATE ou DELETE. Também é usado para executar declarações de dados Definition Language.

3) *<sql:setDataSource>*

Esta tag é usada para exportar uma fonte de dados ou como uma variável de escopo ou como a variável de configuração de fonte de dados.

4) *<sql:param>*

Esta tag é usada para definir os valores de parâmetro.

5) *<sql:dateParam>*

Define os valores dos marcadores de parâmetro em uma instrução SQL para valores de java.util.Date

tipo. Ele também funciona como um subtag de tags e `<sql:query>` `<sql:update>`.

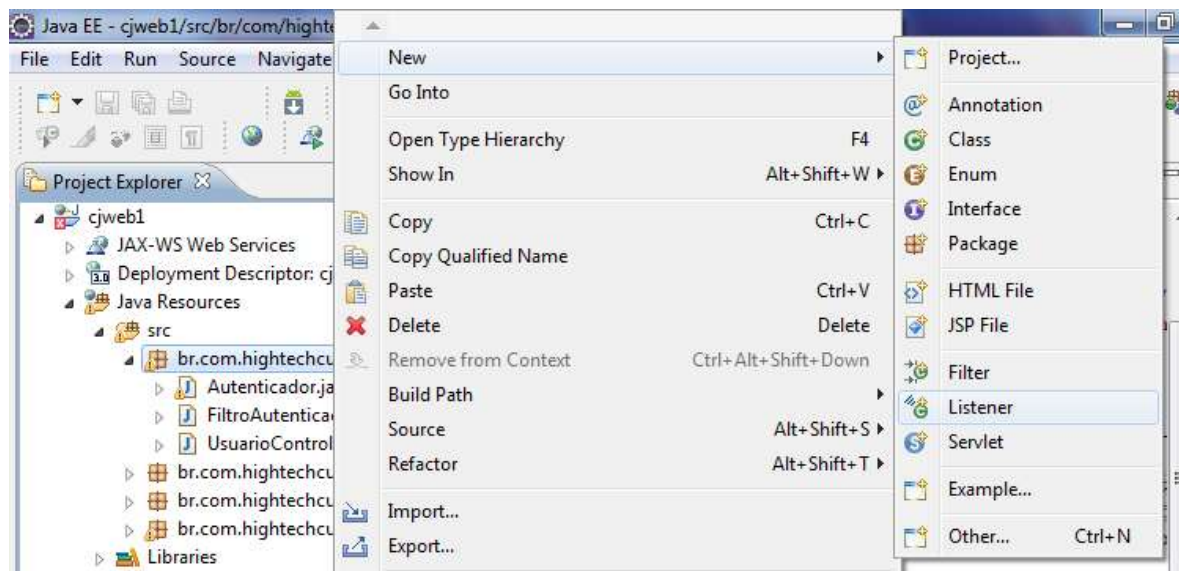
6) `<sql:transaction>`


Esta tag é usada para definir um contexto de transação para `<sql:query>` e submarcas `<sql:update>`.

Criando uma Listener.

O Listener é um objeto ouvidor que está preparado para ouvir um Objeto de evento quando o mesmo acontece.

Por exemplo: Se quisermos ouvir toda vez que uma sessão é criada, então nosso ouvidor teria que implementar os métodos da classe `HttpSessionListener`, `sessionCreated` e `sessionDestroyed`.



 **Create Listener**

Specify class file destination.

Project:

Source folder:


Java package:

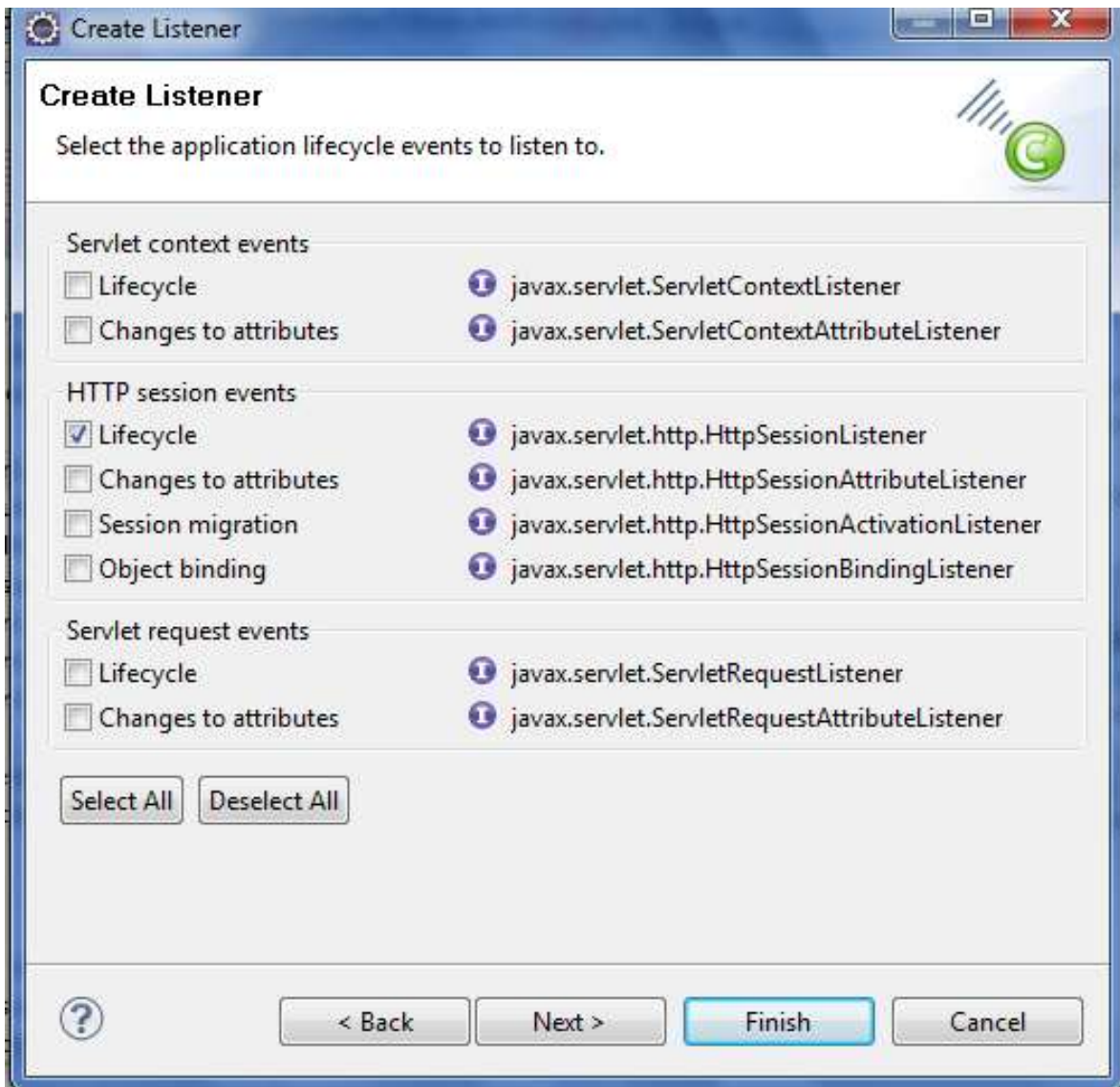
Class name:

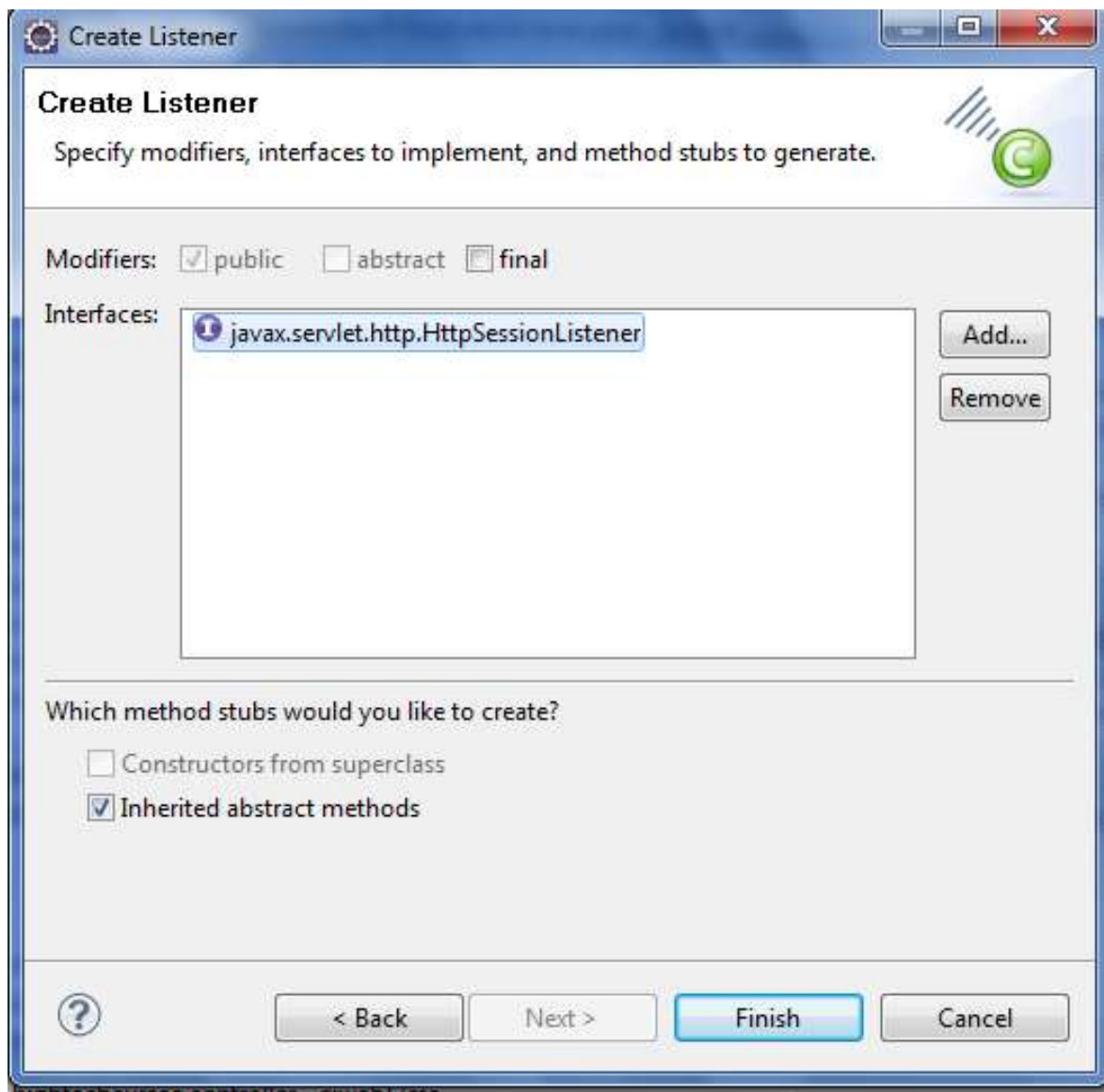
Superclass:

☐ Use existing Listener class

Class name:







Ouvindo a Sessão quando é Criada e Destruída

Neste exemplo vamos mostrar uma forma simples de exibir no browser a quantidade de pessoas que estão autenticadas no sistema.

Para cada sessão criada, nosso ouvidor incrementa mais uma na variável estática quantidade.

```
package br.com.hightechcursos.controller;  
  
import javax.servlet.annotation.WebListener;  
import javax.servlet.http.HttpSessionEvent;  
import javax.servlet.http.HttpSessionListener;  
  
@WebListener
```

```

public class OuvidorSessao implements HttpSessionListener {

    private Integer quantidade = 0 ;

    public OuvidorSessao() {
    }

    /**
     * Chamado toda vez que uma sessão é criada
     */
    public void sessionCreated(HttpSessionEvent event) {
        System.out.println("Sessão Criada");
        quantidade = quantidade +1 ;
        //Adicionando a propriedade quantidade no escopo da aplicacao
        //de modo que fique visível para todos usuários
        adicionarQuantidadeEscopoAplicacao(event);
    }
    /**
     * Chamado toda vez que uma sessão é destruída
     */
    public void sessionDestroyed(HttpSessionEvent event) {
        System.out.println("Sessão Destruída");
        quantidade = quantidade -1 ;
        adicionarQuantidadeEscopoAplicacao(event);
    }
    //Método para colocar a quantidade no escopo da aplicação
    private void adicionarQuantidadeEscopoAplicacao(HttpSessionEvent event) {
        event.getSession().getServletContext().setAttribute("quantidade", quantidade);
    }
}

```

Mostrando a quantidade de autenticados na tela de bem vindo (index.jsp)

```

<%@page import="br.com.hightechcursos.controller.LogUsuario"%>
<%@page import="br.com.hightechcursos.controller.OuvidorSessao"%>
<%@page import="br.com.hightechcursos.entidades.Usuario"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%@include file="includes/menu.jsp" %>
<%
//Captura usuário da sessão
Usuario usuAutenticado = (Usuario) session.getAttribute("usuAutenticado");
%>
<p>
Seja bem vindo <b><%= usuAutenticado.getNome() + " id:" + usuAutenticado.getId() %></b>

```

```
Quantidade de logados: <%= application.getAttribute("quantidade") %>
```

```
</p>  
</body>  
</html>
```

Este material é para a comunidade estudante da tecnologia Java e também é usado como apoio em nossos cursos.

Por favor, nos ajude a melhorar nosso material enviando sugestões, críticas e elogios para apostilas@hightechcursos.com.br