TUTORIAL

DESENVOLVIMENTO DE PROJETO WEB EM JAVA COM JSF 2.2 + PRIMEFACES 5.3 + HIBERNATE 5.1 + ECLIPSE 4.5.1 + PUG PLUGIN 1.0.1

Em 13 Etapas

Criado por: Alberto Henrique Sousa

2016

Objetivo

Criar um projeto Web em Java, com login de acesso e três CRUD (Create, Read, Update e Delete) para gerenciamento das tabelas: Usuários, Clientes e Tipos.

Tecnologias utilizadas:

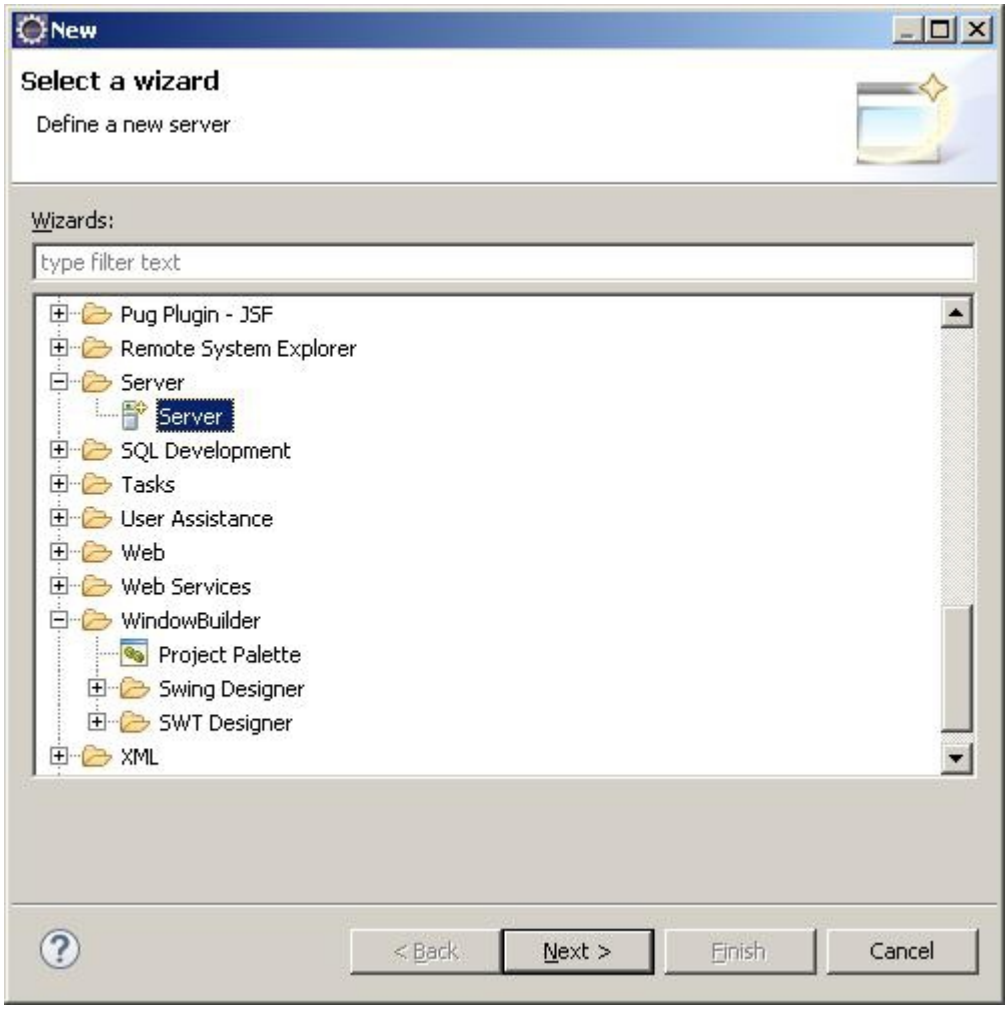
PostgreSql 9.3
Eclipse 4.5.1
JSF 2.2
PrimeFaces 5.3
Hibernate 5.1
Pug Plugin 1.0.1 (Desenvolvido por Alberto Henrique Sousa)

1) Script do banco de dados:

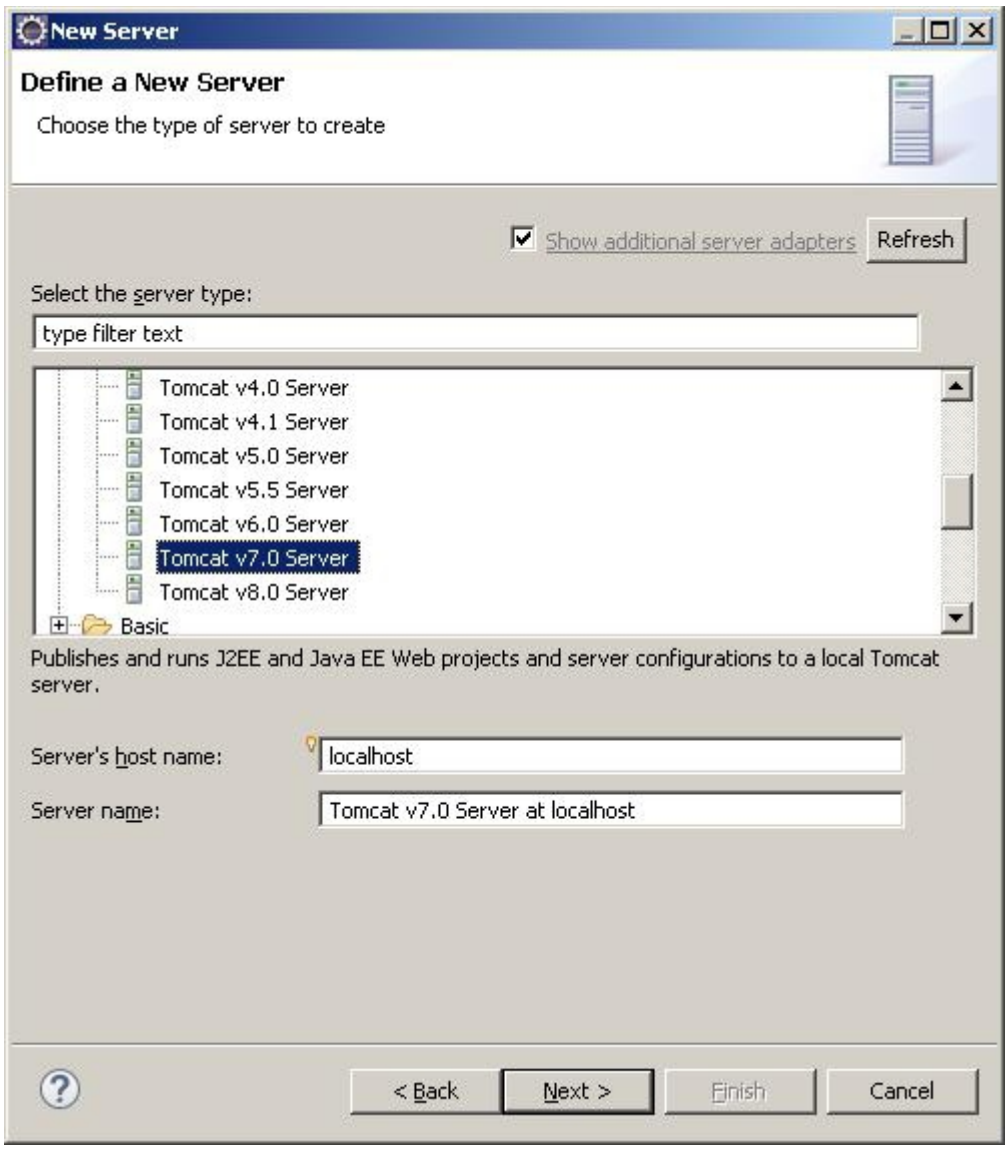
```
CREATE DATABASE "curso-jsf"
  WITH ENCODING='UTF8'
       CONNECTION LIMIT=-1;

CREATE TABLE usuarios
(
  id serial NOT NULL,
  nome character varying(30),
  senha character varying(100),
  CONSTRAINT pk_usuarios_id PRIMARY KEY (id)
)
WITH (
  OIDS=FALSE
);

CREATE TABLE tipos
(
  id serial NOT NULL,
  descricao character varying(80),
  CONSTRAINT tipos_pkey PRIMARY KEY (id)
)
WITH (
  OIDS=FALSE
);

CREATE TABLE clientes
(
  id serial NOT NULL,
  nome character varying(80),
  tipos_id integer,
  CONSTRAINT pk_clientes_id PRIMARY KEY (id),
  CONSTRAINT fk_tipos_id FOREIGN KEY (tipos_id)
      REFERENCES tipos (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);

CREATE INDEX fki_tipos_id
  ON clientes
  USING btree
  (tipos_id);
```

2) Server Tomcat

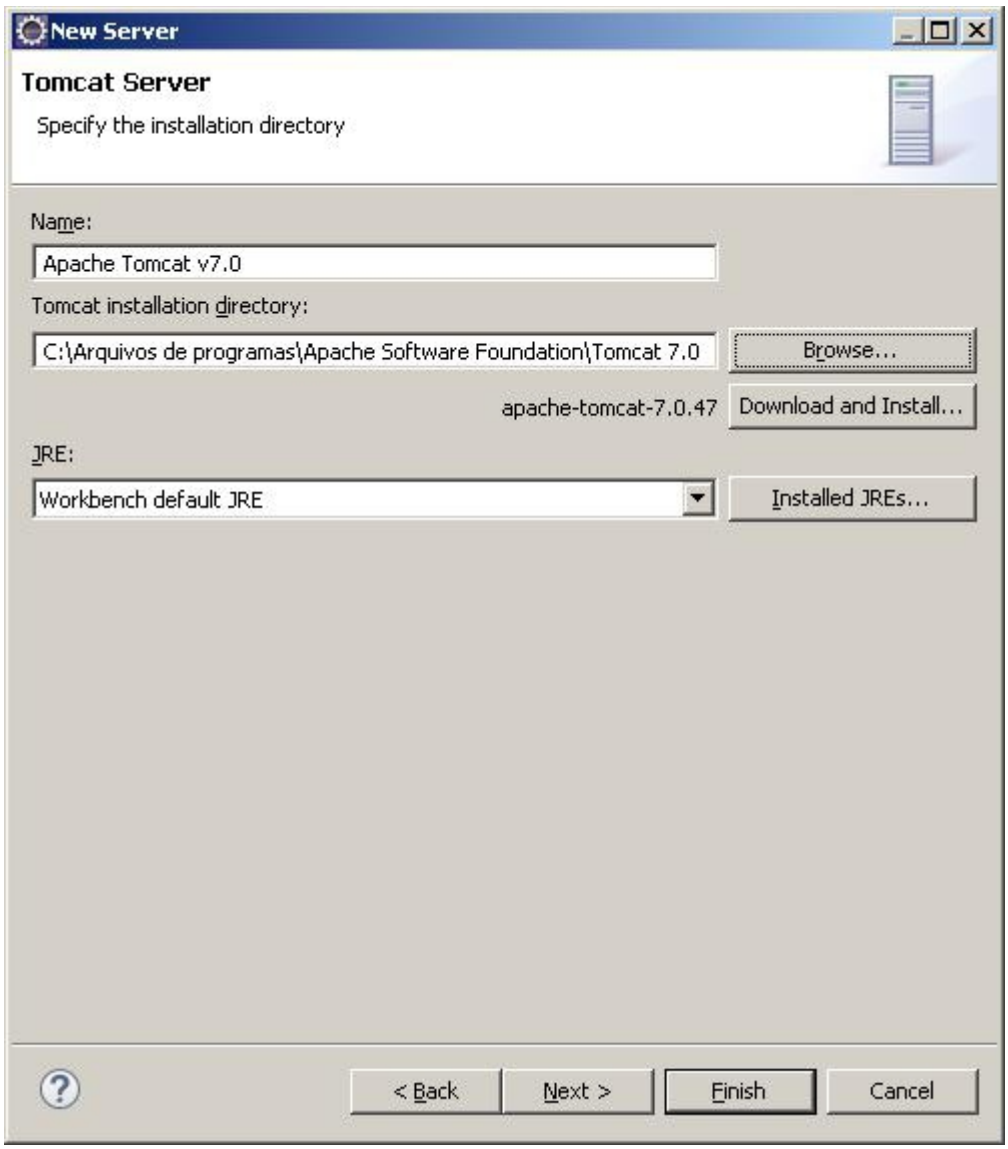
Abra o Eclipse e crie o “Server Tomcat”.

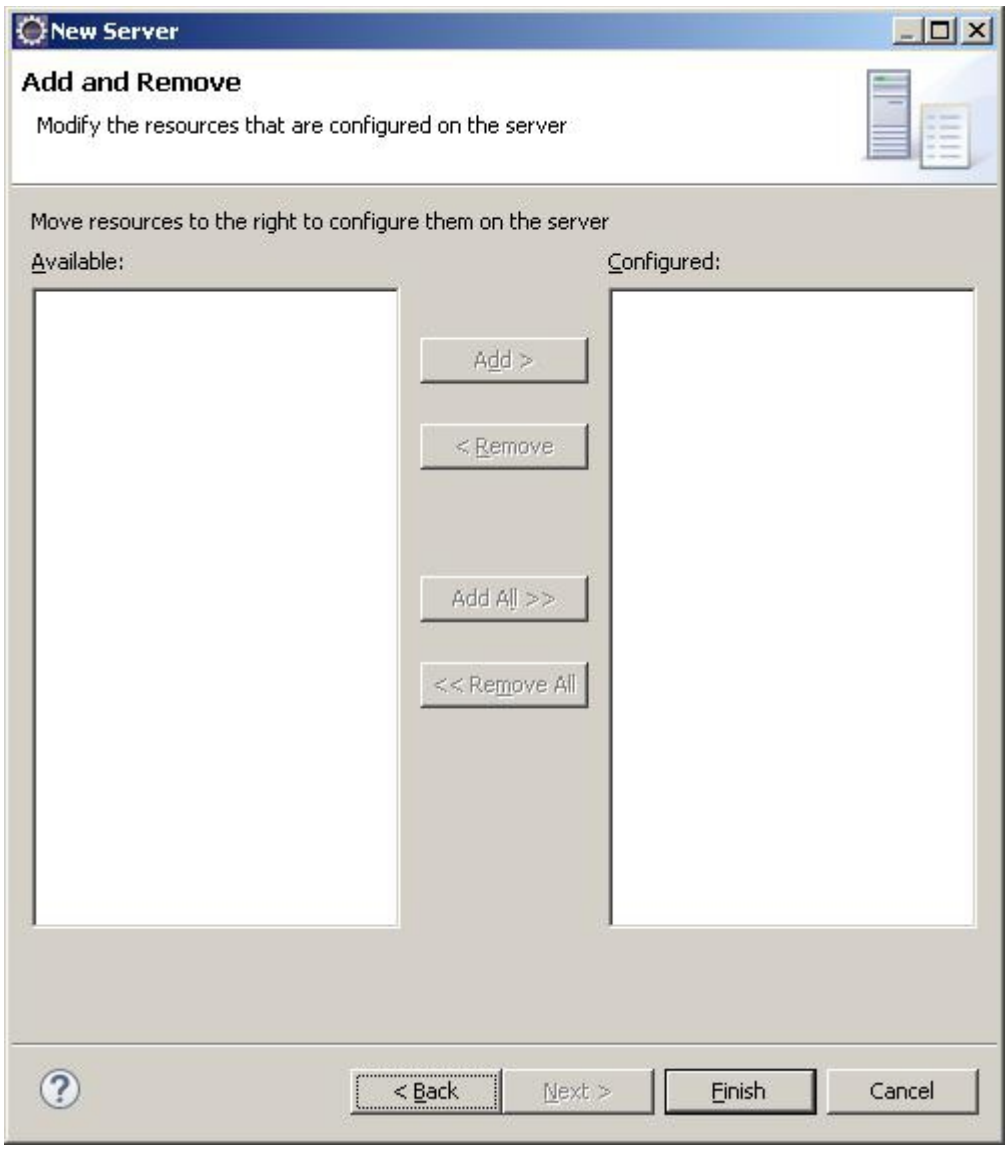


* Pule essa etapa caso já tenha o Tomcat 7 configurado na sua workspace.

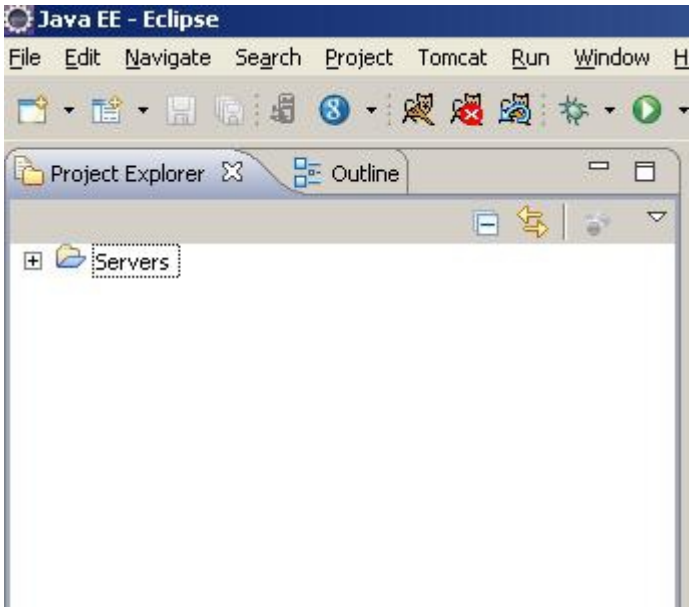


Escolha a versão 7 do Tomcat.


New Server
Tomcat Server
Specify the installation directory
Name:
Apache Tomcat v7.0
Tomcat installation directory:
C:\Arquivos de programas\Apache Software Foundation\Tomcat 7.0
Browse...
apache-tomcat-7.0.47
Download and Install...
JRE:
Workbench default JRE
Installed JREs...
< Back
Next >
Finish
Cancel



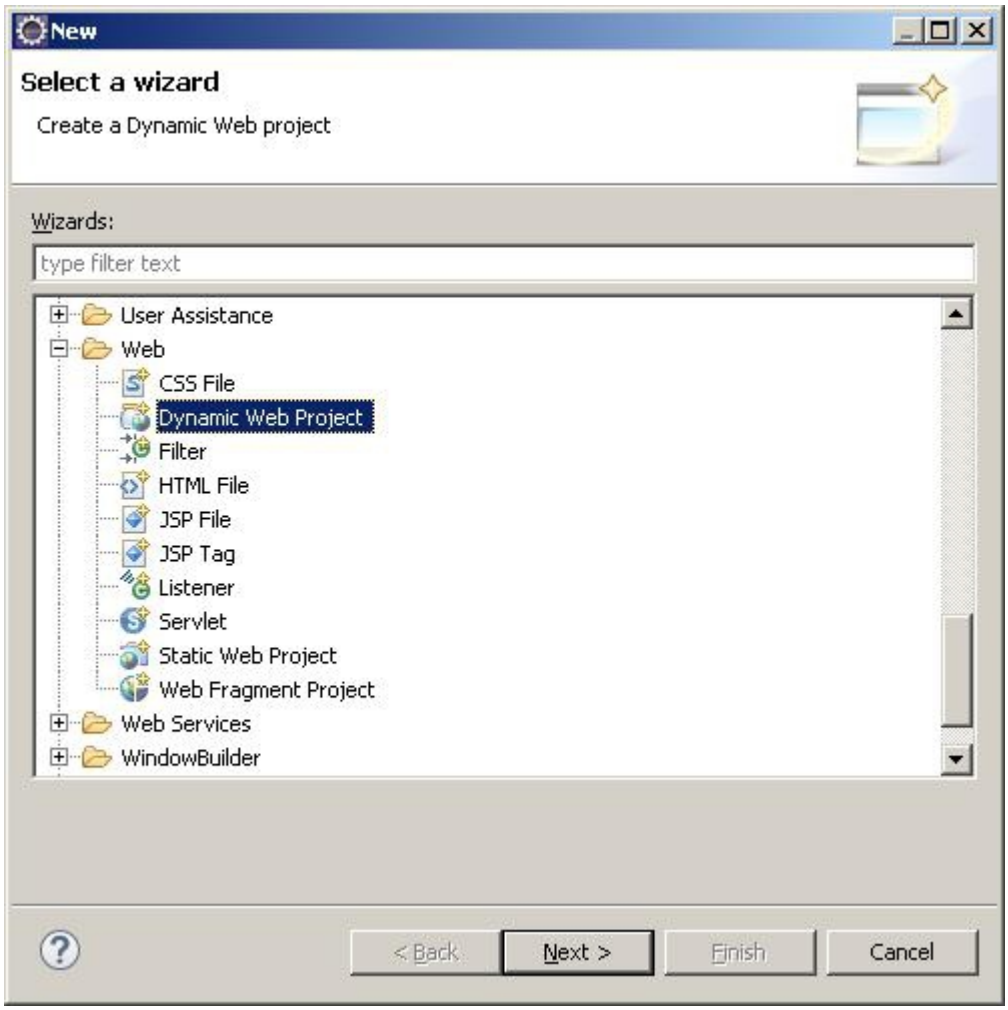
Como o projeto ainda não foi criado não teremos a vinculação nessa etapa.

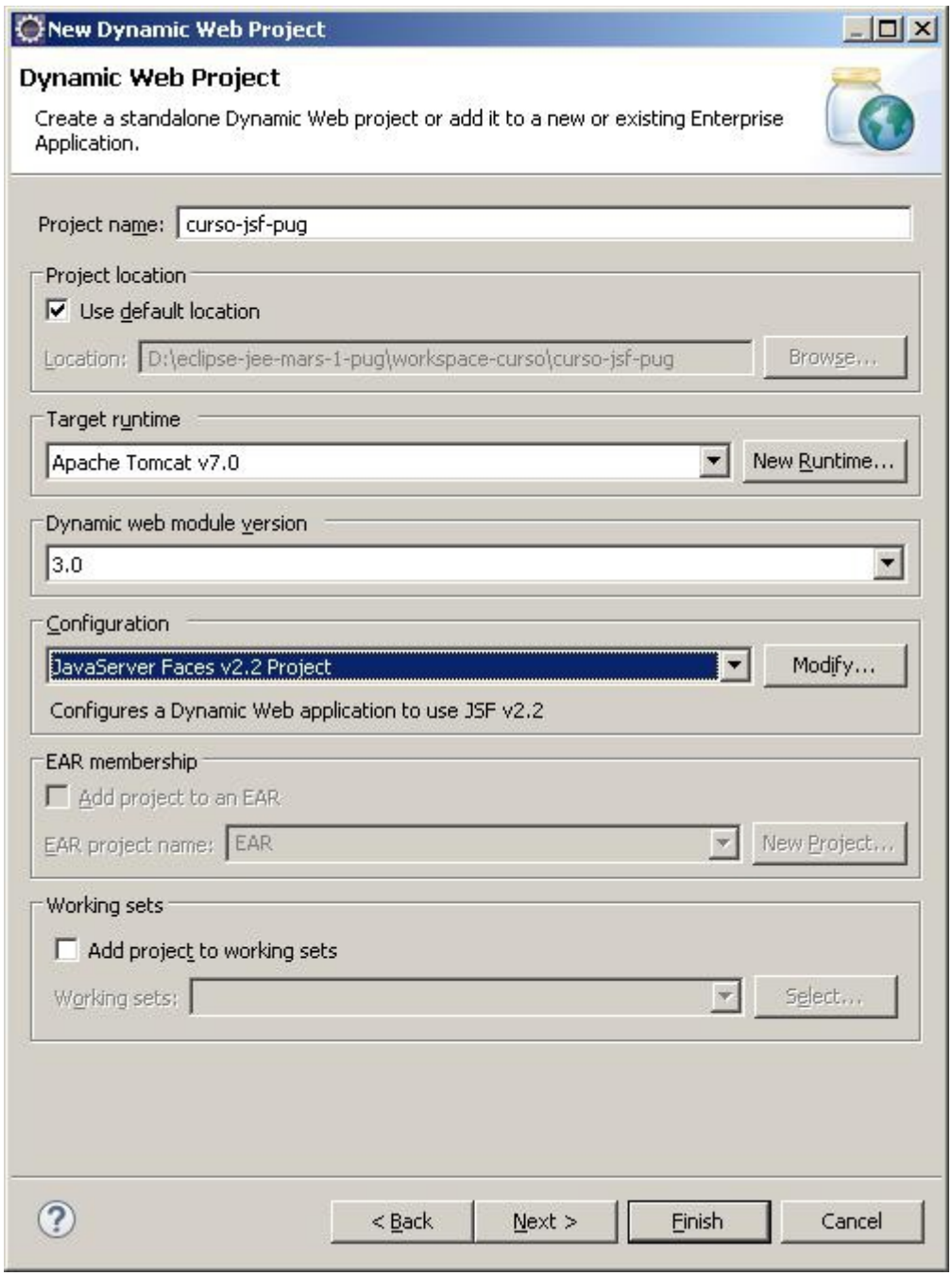


Servidor configurado até aqui.

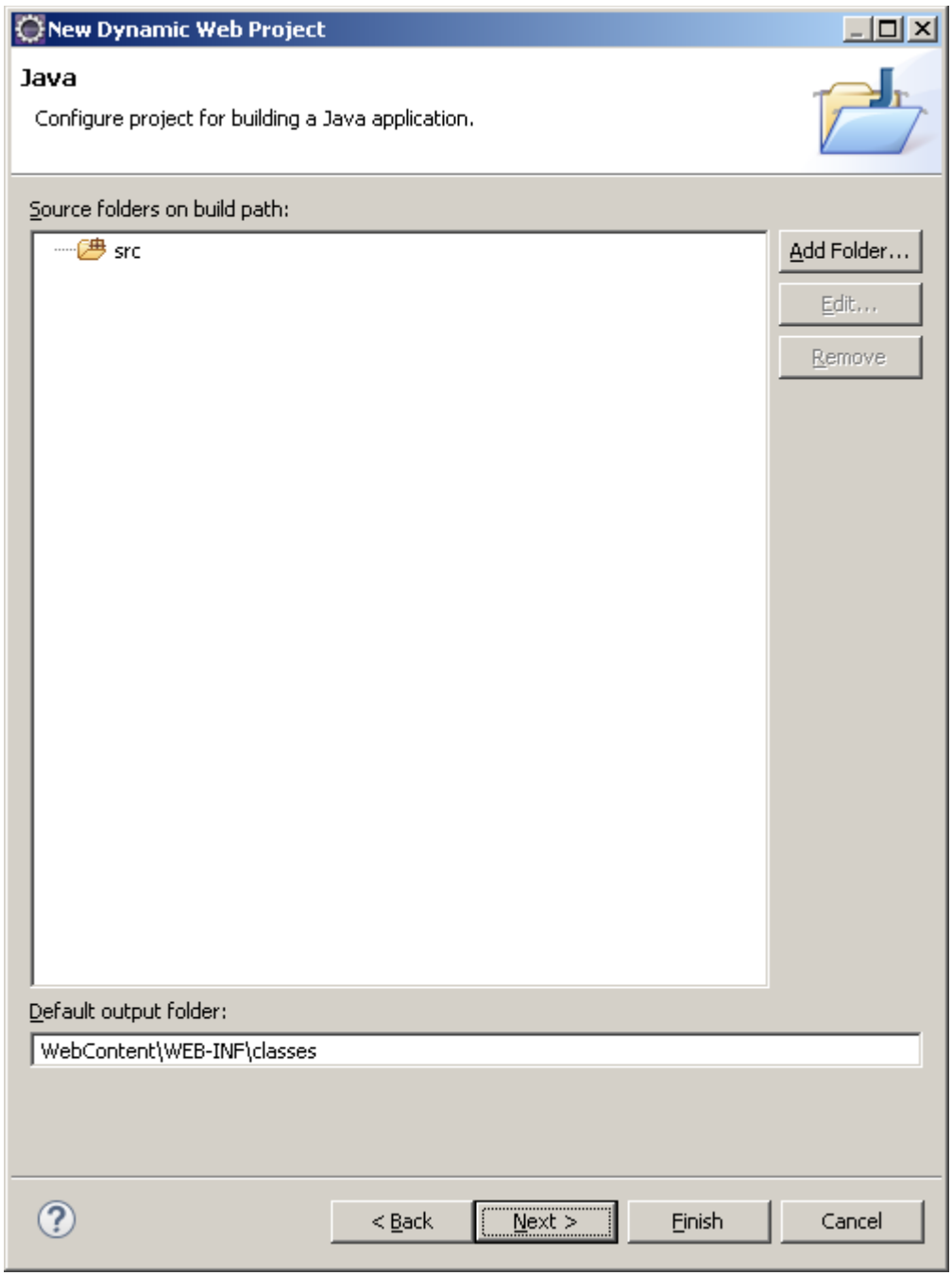
3) Novo Projeto

Crie um novo projeto do tipo Web → Dynamic Web Projetct.

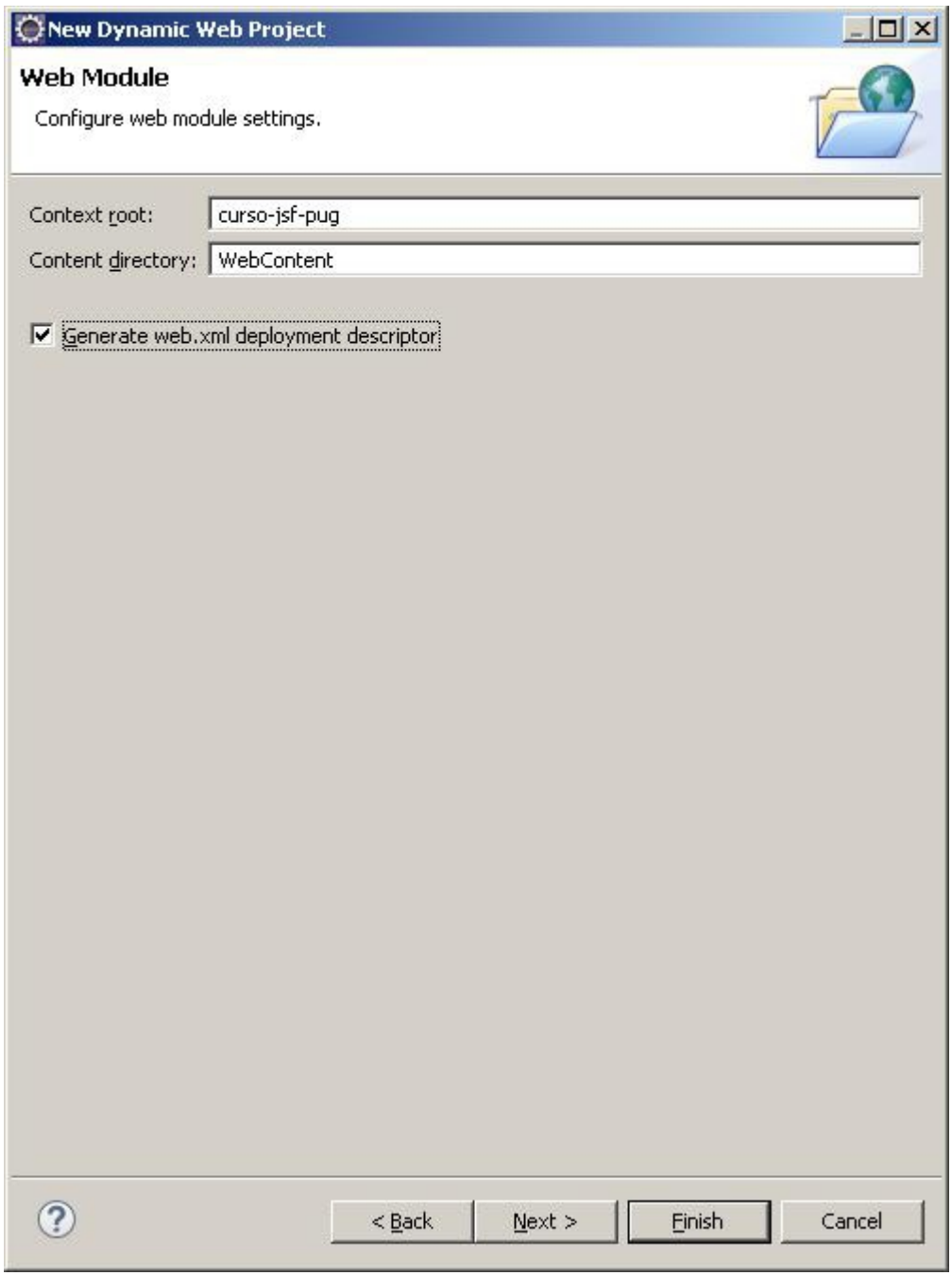




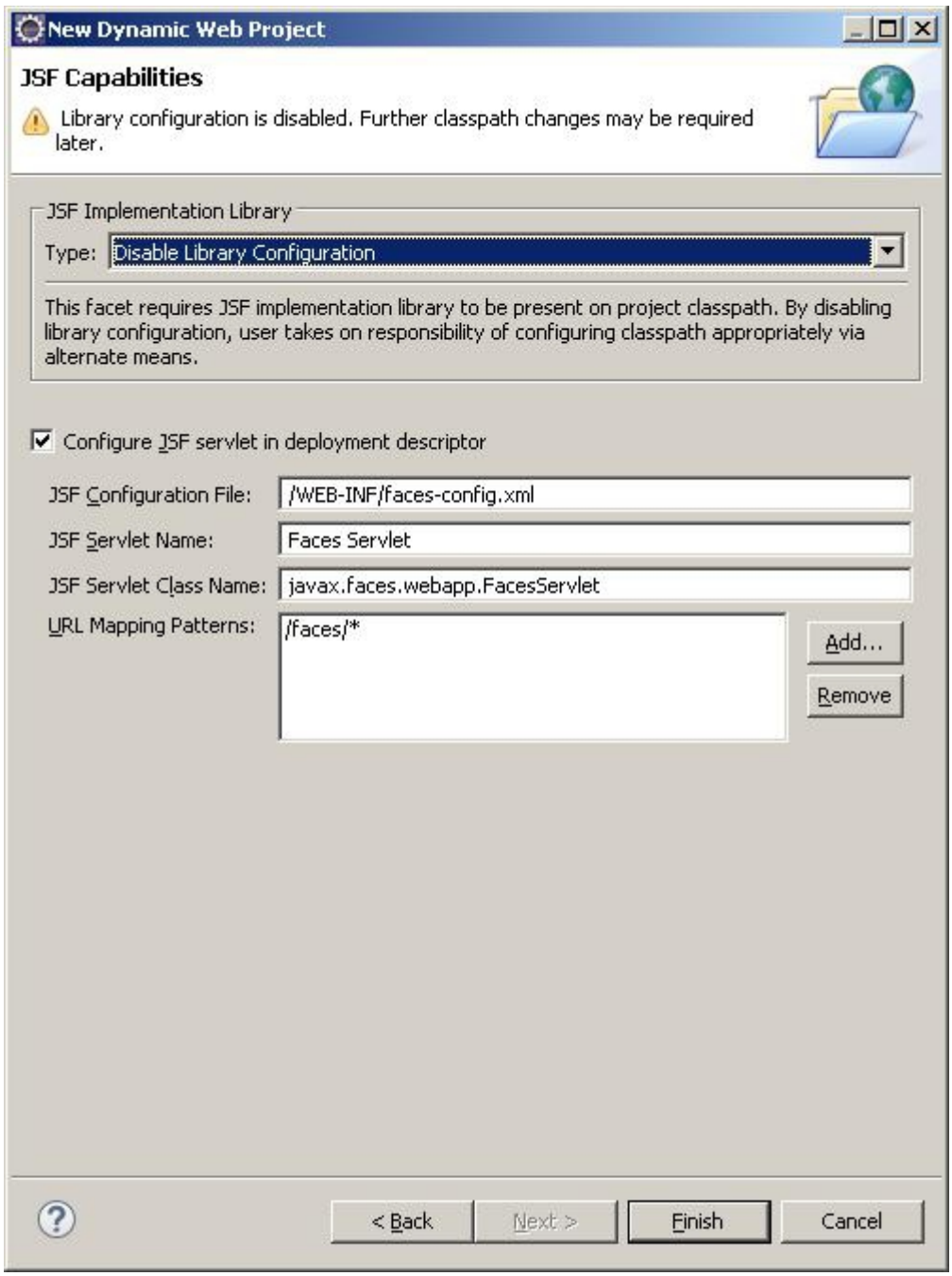
“Project name”: curso-jsf-pug.
“Target runtime”: Escolha o servidor que foi configurado no tópico anterior.
“Configuration”: “JavaServer Faces v2.2 Project”.



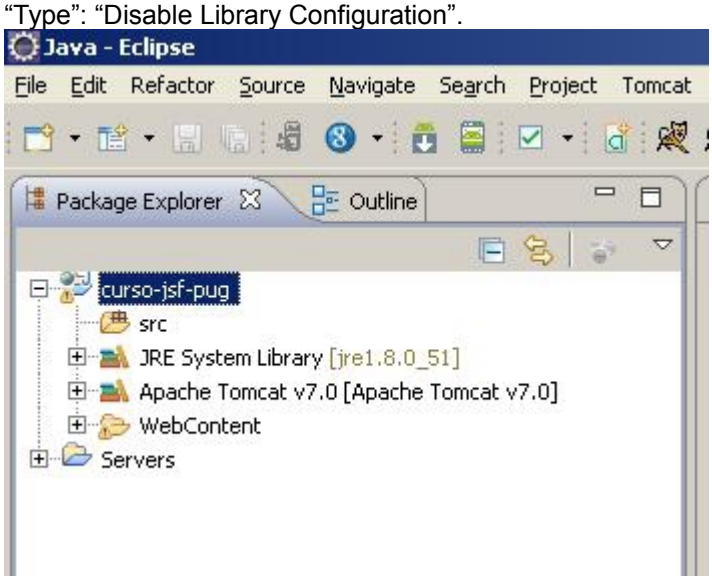
“Default output folder”: Substitua “build” por “WebContent\WEB-INF”.



Marque a opção “Generate web.xml deployment descriptor”.

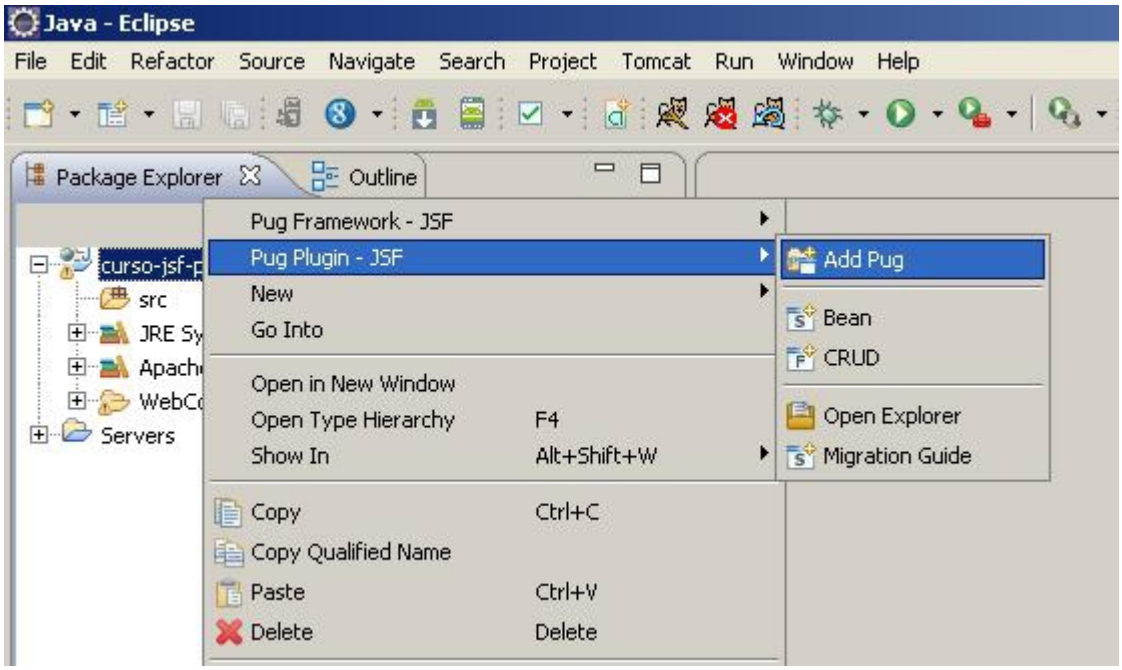


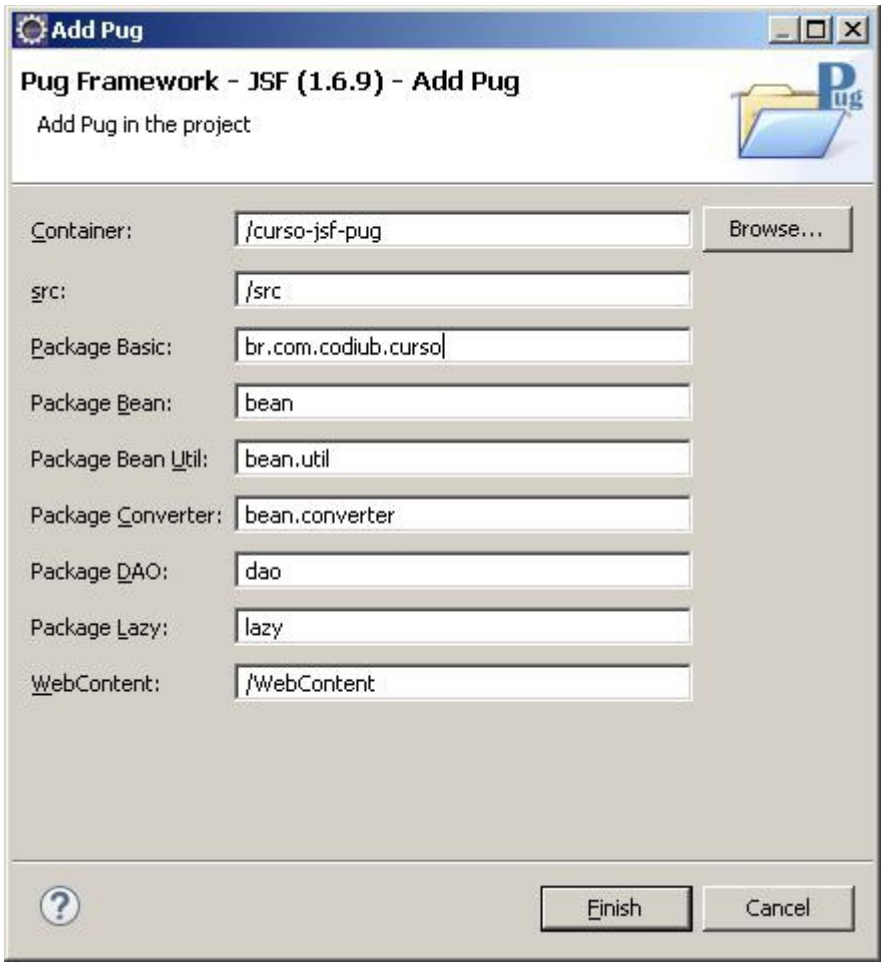
"Type": "Disable Library Configuration".



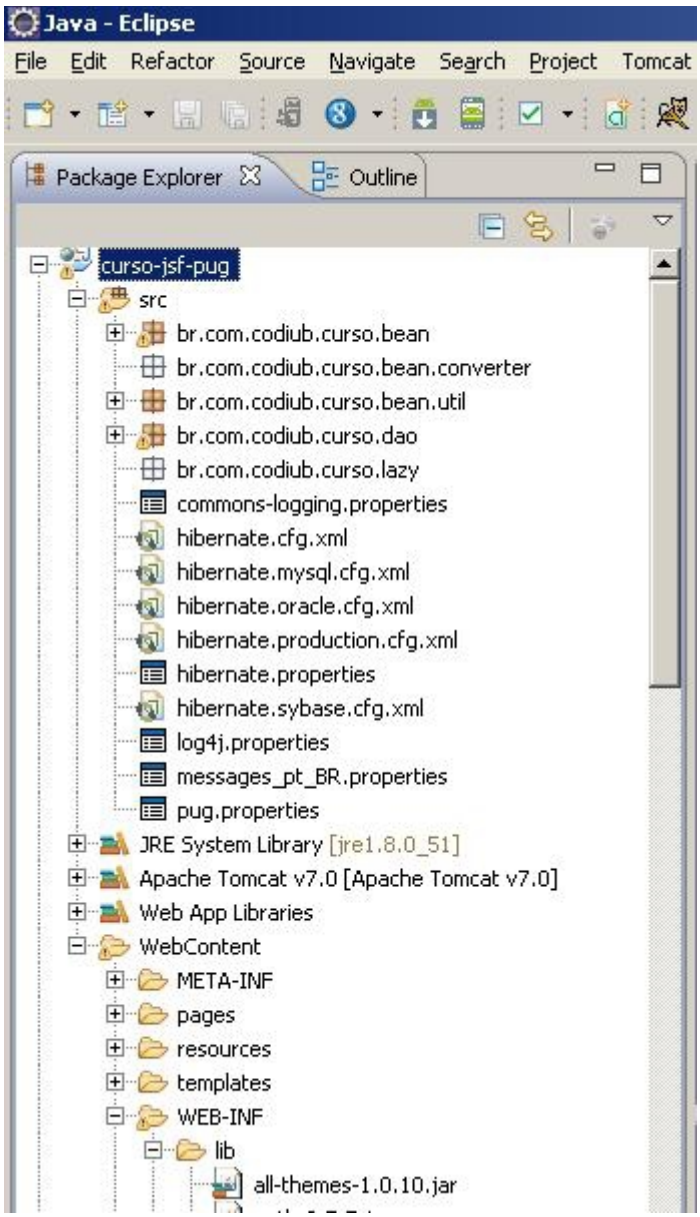
4) Adicionando o Pug

Adicione o Pug no projeto recém-criado.





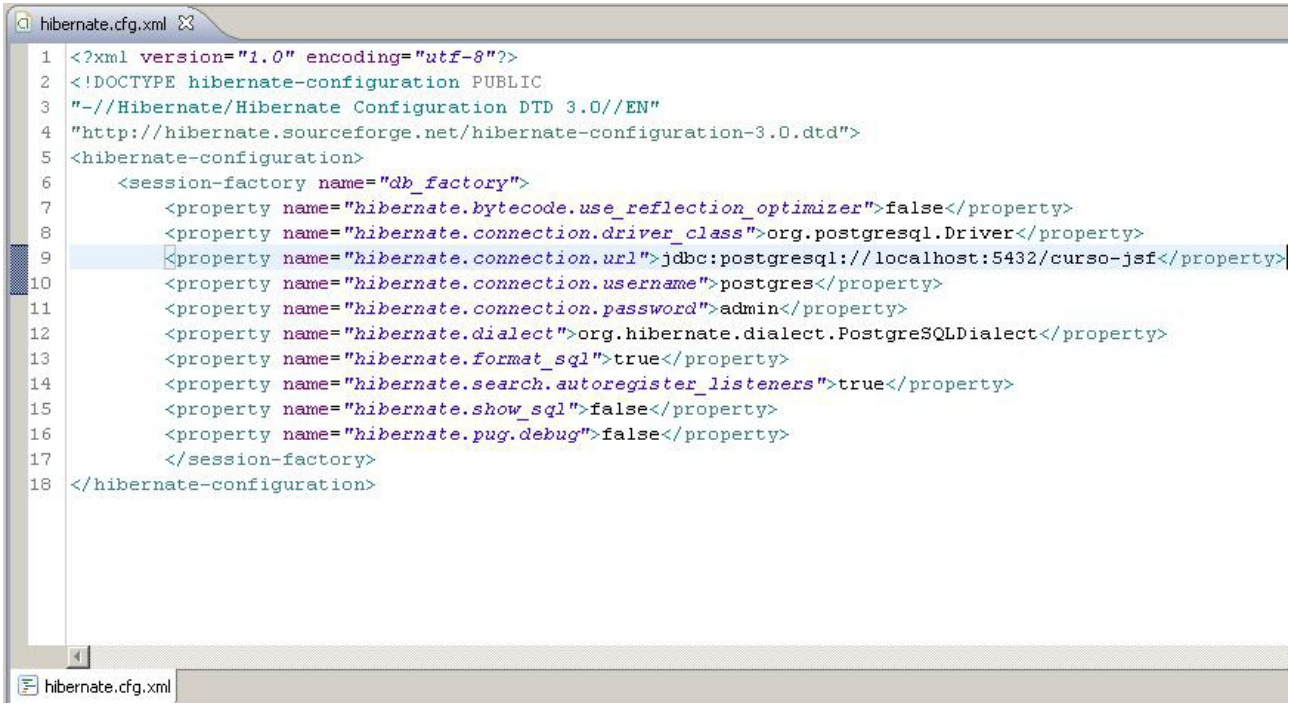
“Package Basic”: “br.com.codiub.curso”.



Após adicionar o Pug o projeto conterá a estrutura para uso do JSF + PrimeFaces + Hibernate.

5) Configurando Conexão com o Banco

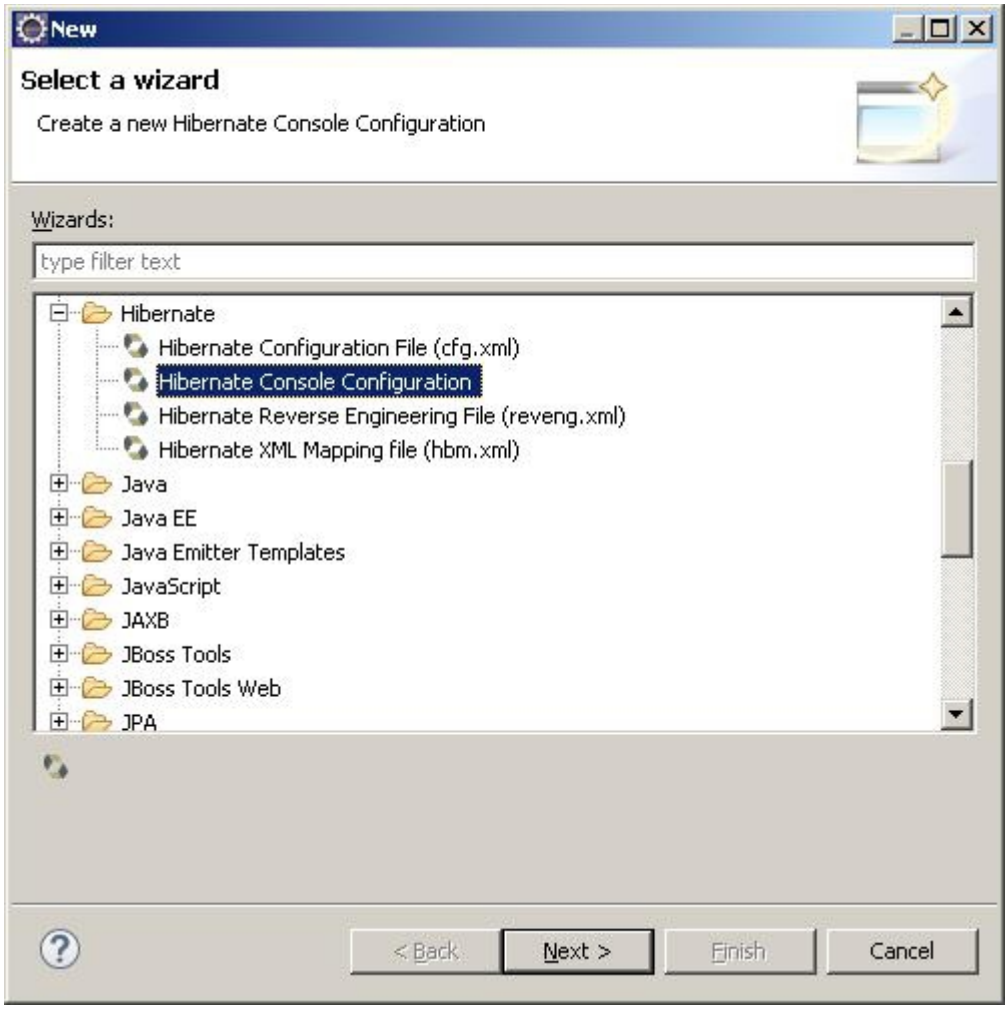
“hibernate.cfg.xml”: Modifique a URL, username e password conforme as configurações do seu database.

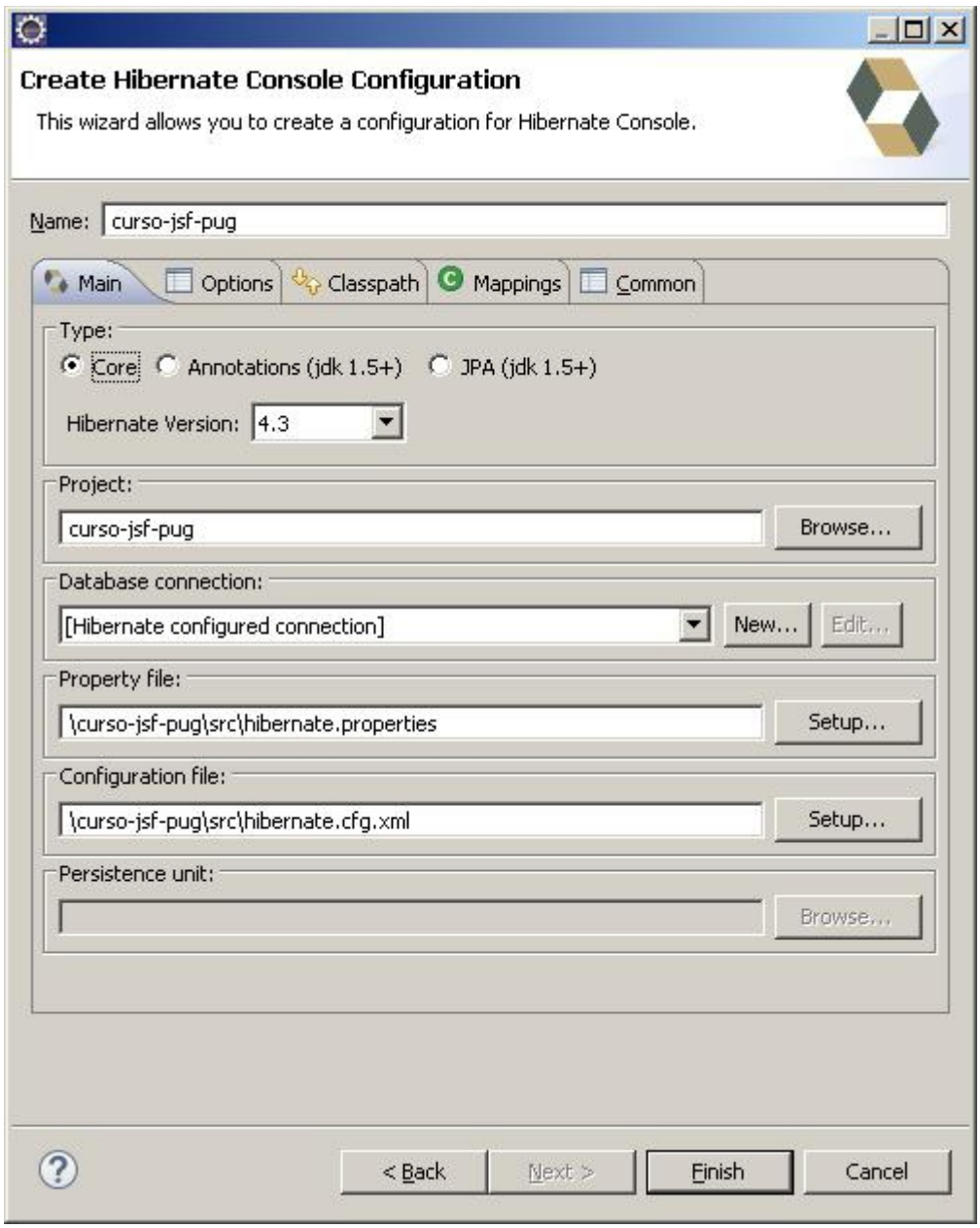


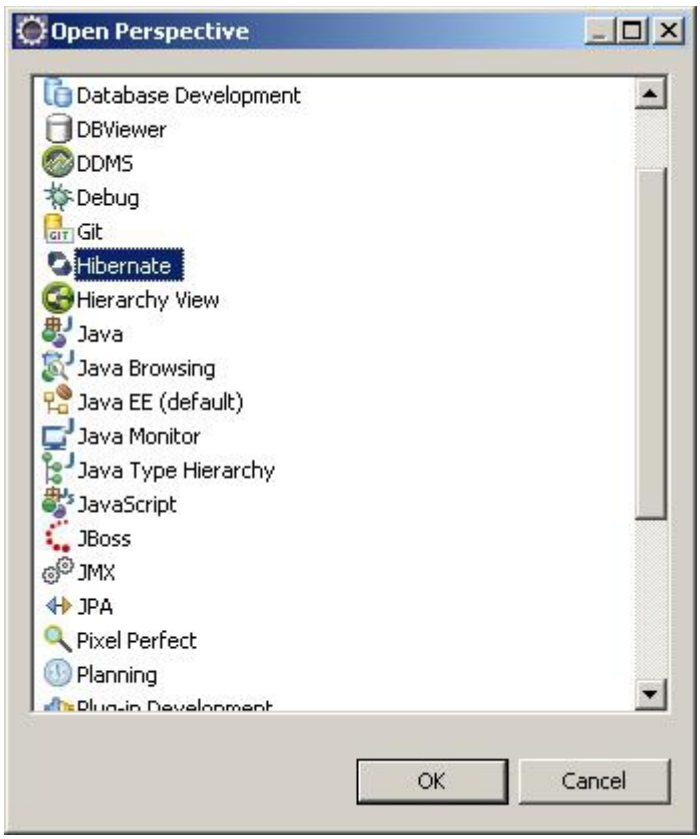
O Pug insere alguns arquivos “cfg.xml” de conexão com o banco, o padrão é o “hibernate.cfg.xml”, você pode mudar o padrão através do arquivo “pug.properties”. Esse tipo de configuração permite trabalhar com conexões diferentes, tais como: desenvolvimento e produção.

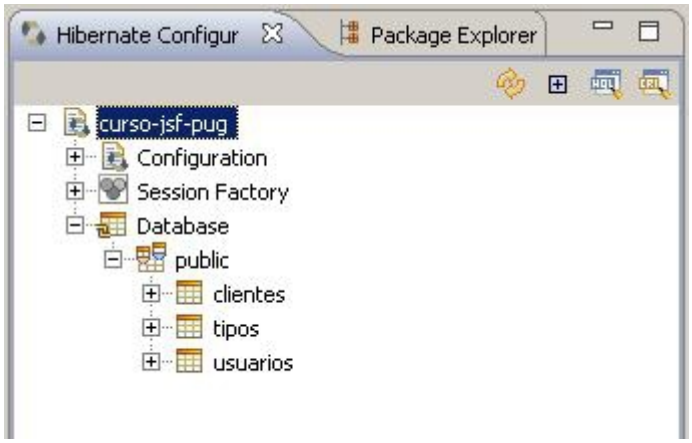
6) Console e Engenharia Reversa do Hibernate

É necessário definir uma configuração de console para que o Plugin Hibernate Tools possa se conectar com a base de dados e assim realizar a engenharia reversa para criação das classes de objetos de mapeamento com o database.



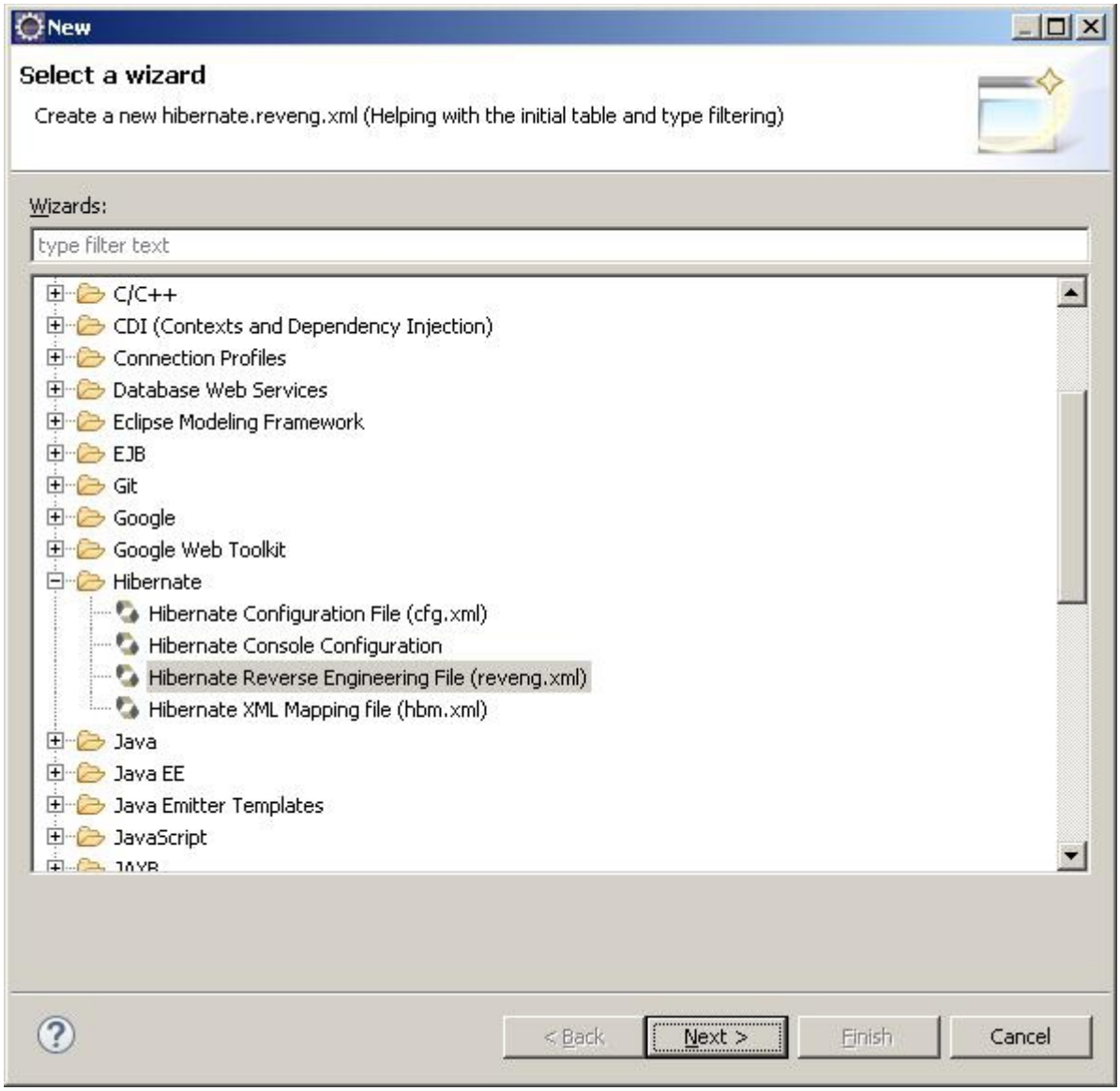

Create Hibernate Console Configuration
This wizard allows you to create a configuration for Hibernate Console.
Name: curso-jsf-pug
Main
Options
Classpath
Mappings
Common
Type:
Core
Annotations (jdk 1.5+)
JPA (jdk 1.5+)
Hibernate Version: 4.3
Project:
curso-jsf-pug
Browse...
Database connection:
[Hibernate configured connection]
New...
Edit...
Property file:
\curso-jsf-pug\src\hibernate.properties
Setup...
Configuration file:
\curso-jsf-pug\src\hibernate.cfg.xml
Setup...
Persistence unit:
Browse...
< Back
Next >
Finish
Cancel

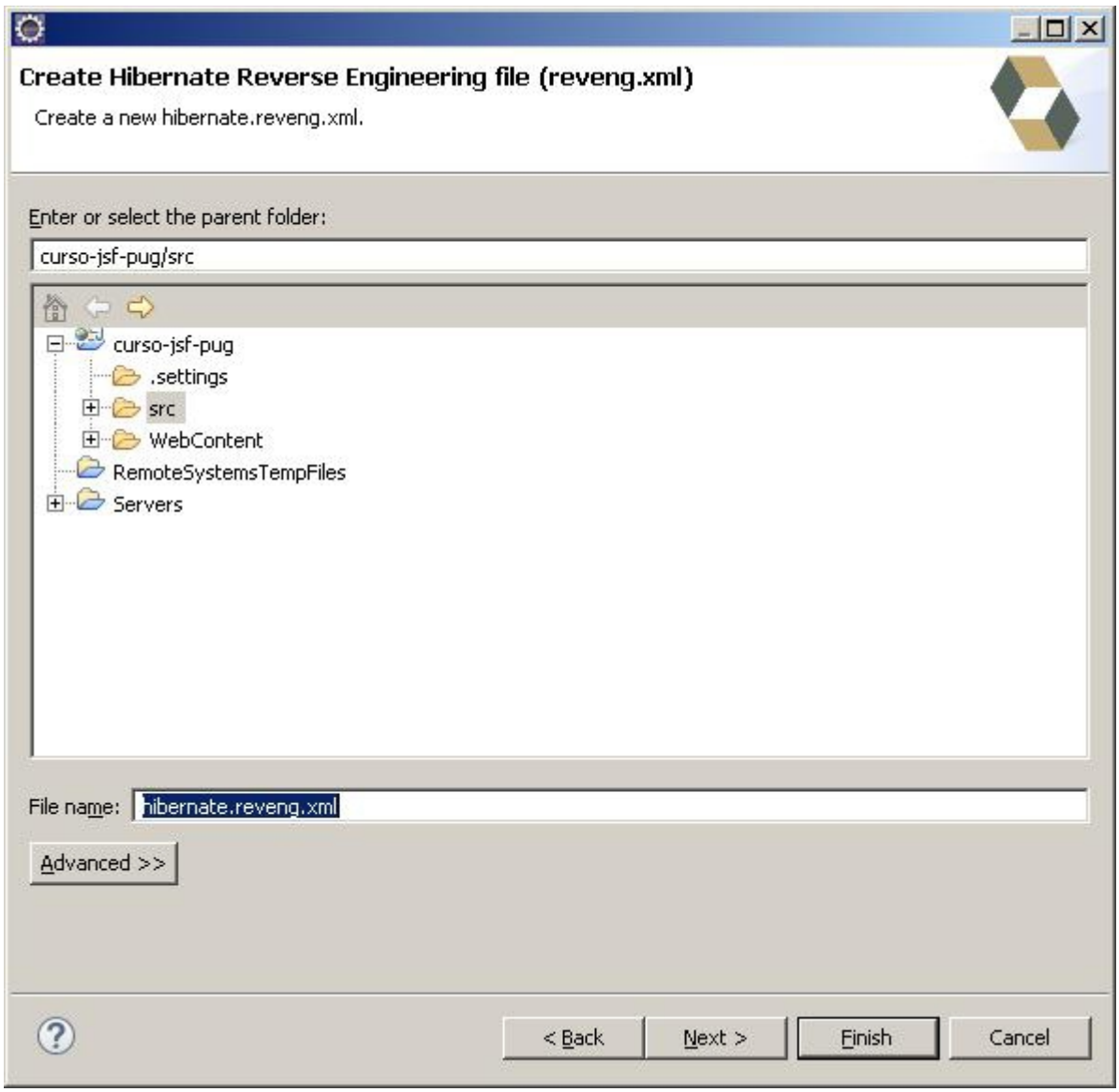


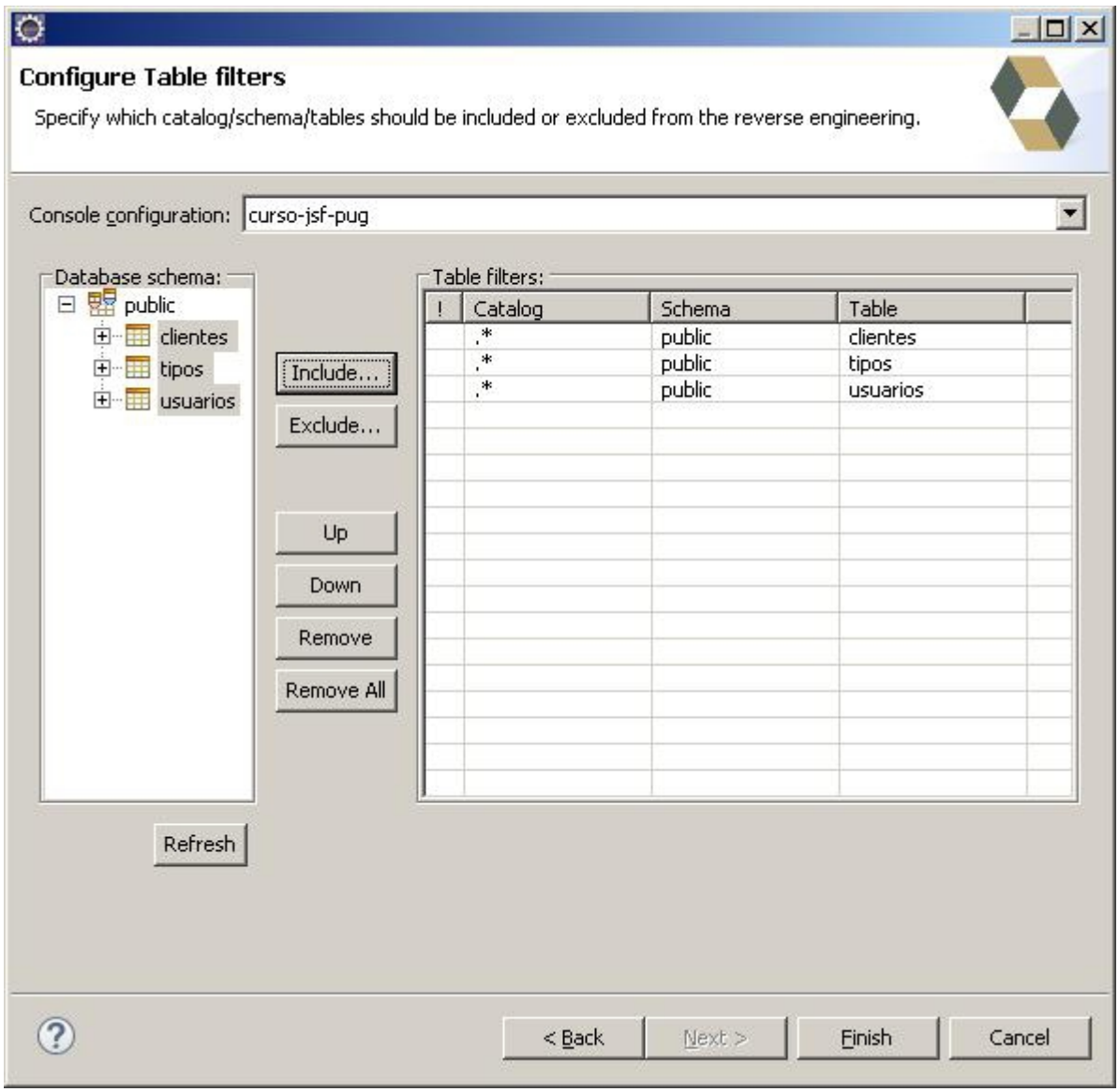


Agora o Hibernate Tools é capaz de fazer uma leitura da estrutura do database.

Para obter a engenharia reversa do database para criação das classes de objetos é necessário criar o “reveng.xml”.




Create Hibernate Reverse Engineering file (reveng.xml)
Create a new hibernate.reveng.xml.
Enter or select the parent folder:
curso-jsf-pug/src
curso-jsf-pug
.settings
src
WebContent
RemoteSystemsTempFiles
Servers
File name:
hibernate.reveng.xml
Advanced >>
< Back
Next >
Finish
Cancel

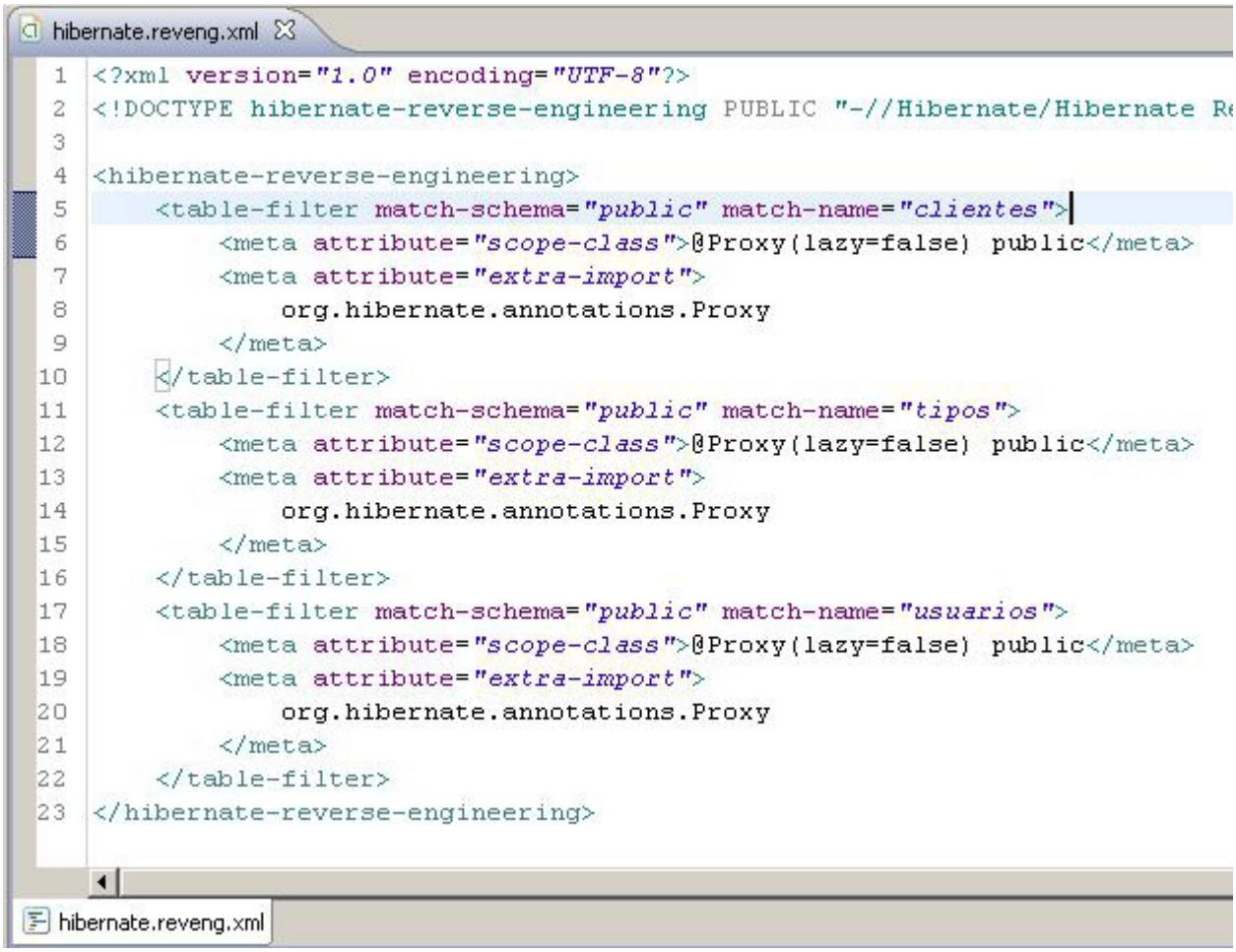


Inclua as três tabelas em “Table filters”.

hibernate.reveng.xml

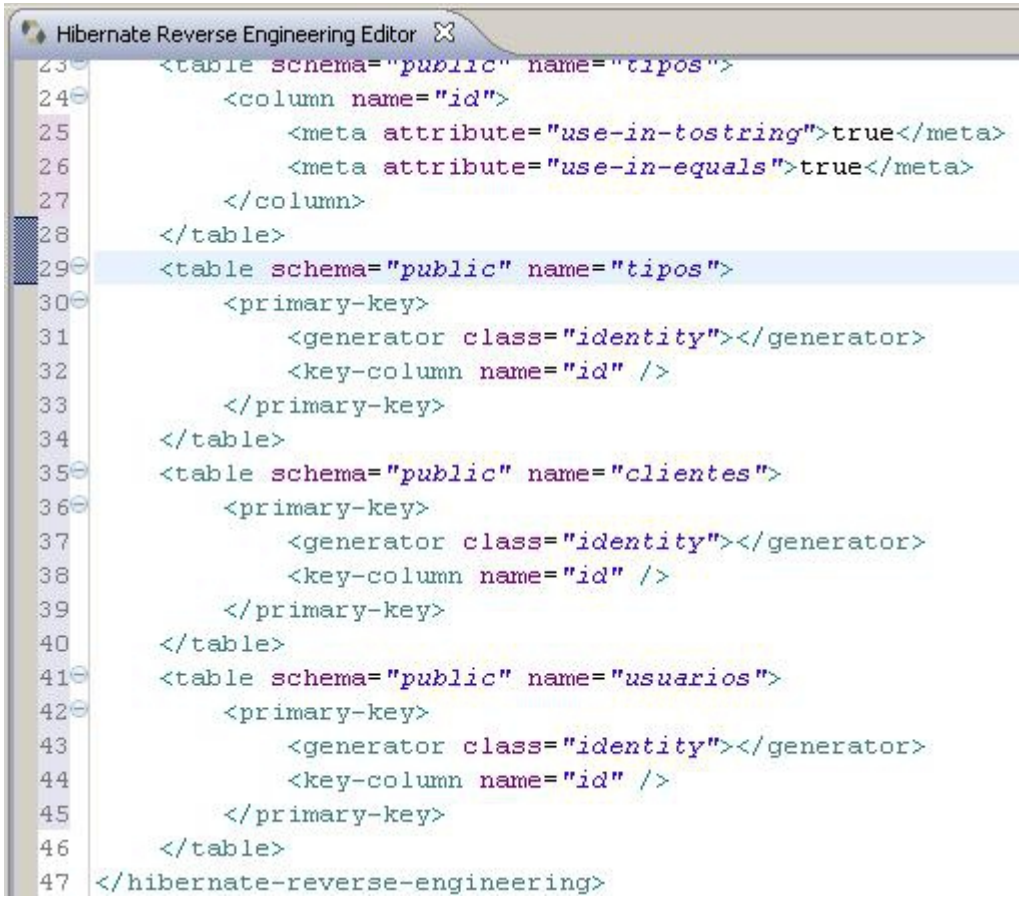
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hib
3
4  <hibernate-reverse-engineering>
5    <table-filter match-schema="public" match-name="clientes"/>
6    <table-filter match-schema="public" match-name="tipos"/>
7    <table-filter match-schema="public" match-name="usuarios"/>
8  </hibernate-reverse-engineering>
```

Será necessário incluir algumas configurações para o funcionamento correto com o JSF e Hibernate.



hibernate.reveng.xml

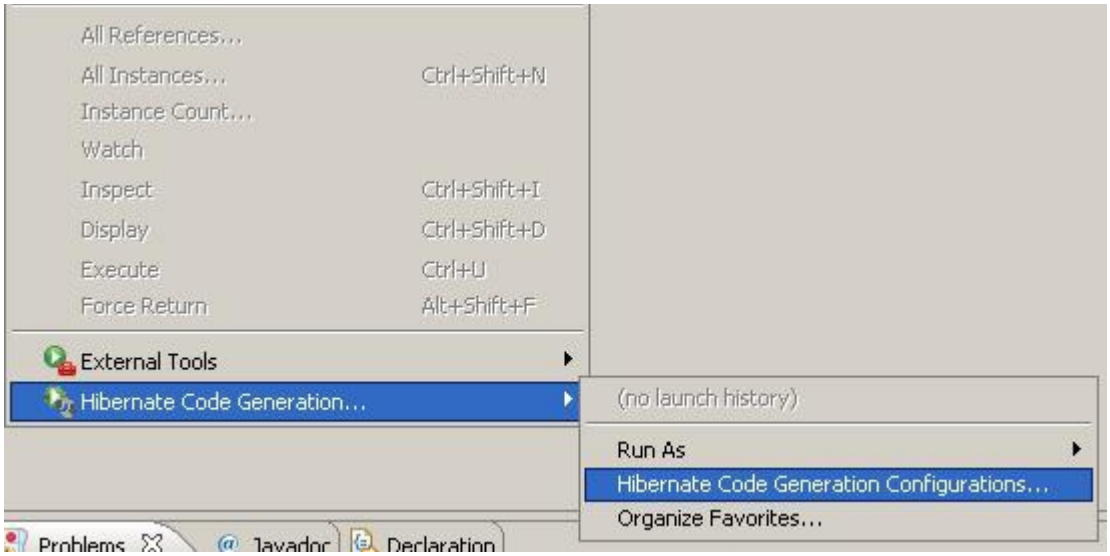
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate

<hibernate-reverse-engineering>
    <table-filter match-schema="public" match-name="clientes">
        <meta attribute="scope-class">@Proxy(lazy=false) public</meta>
        <meta attribute="extra-import">
            org.hibernate.annotations.Proxy
        </meta>
    </table-filter>
    <table-filter match-schema="public" match-name="tipos">
        <meta attribute="scope-class">@Proxy(lazy=false) public</meta>
        <meta attribute="extra-import">
            org.hibernate.annotations.Proxy
        </meta>
    </table-filter>
    <table-filter match-schema="public" match-name="usuarios">
        <meta attribute="scope-class">@Proxy(lazy=false) public</meta>
        <meta attribute="extra-import">
            org.hibernate.annotations.Proxy
        </meta>
    </table-filter>
    <table schema="public" name="tipos">
        <column name="id">
            <meta attribute="use-in-tostring">true</meta>
            <meta attribute="use-in-equals">true</meta>
        </column>
    </table>
</hibernate-reverse-engineering>
```



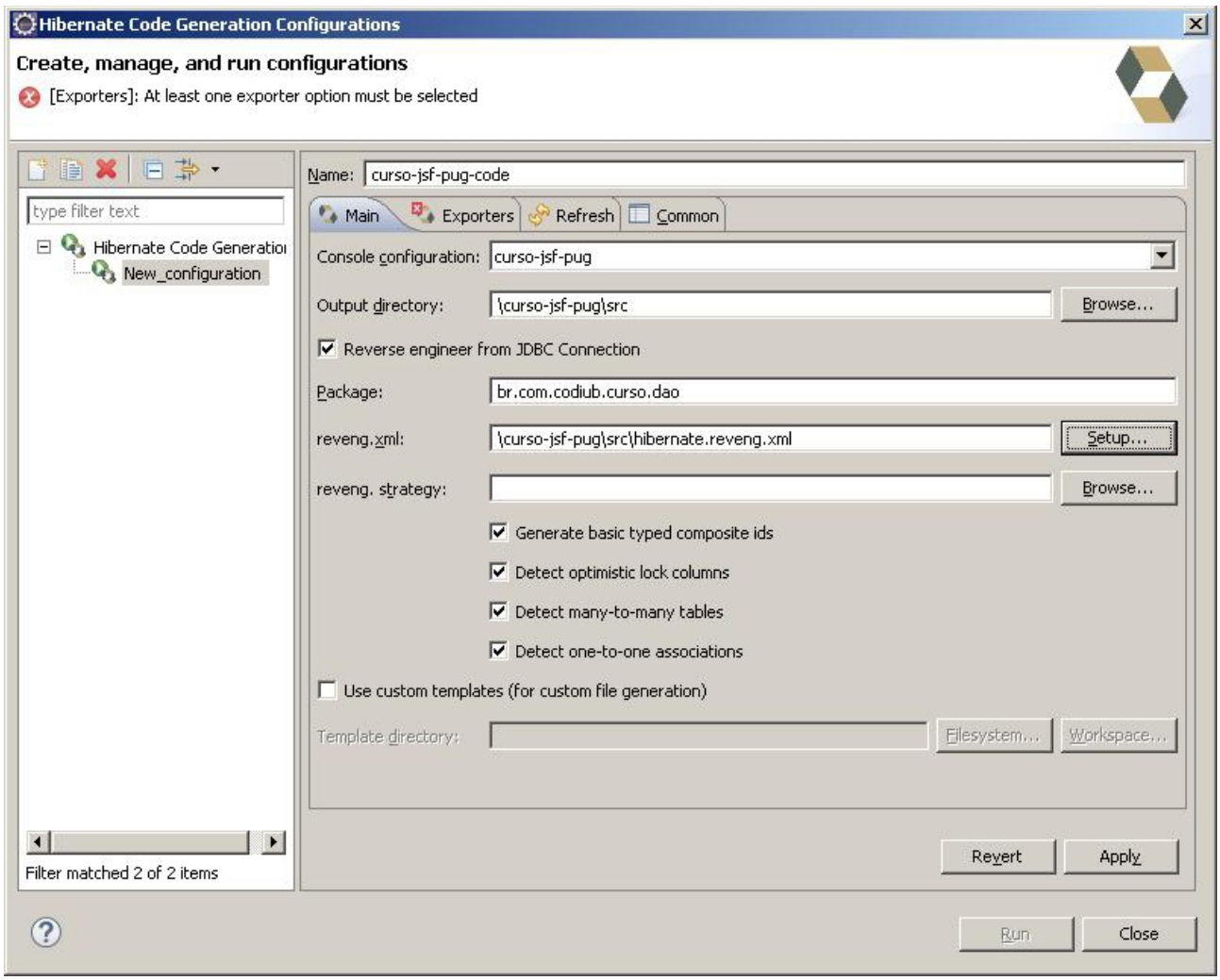
Esse tipo de configuração vai depender do tipo de database, pois cada um tem as suas particularidades de tipos de campos. Segue a codificação usada no “hibernate.reveng.xml”:

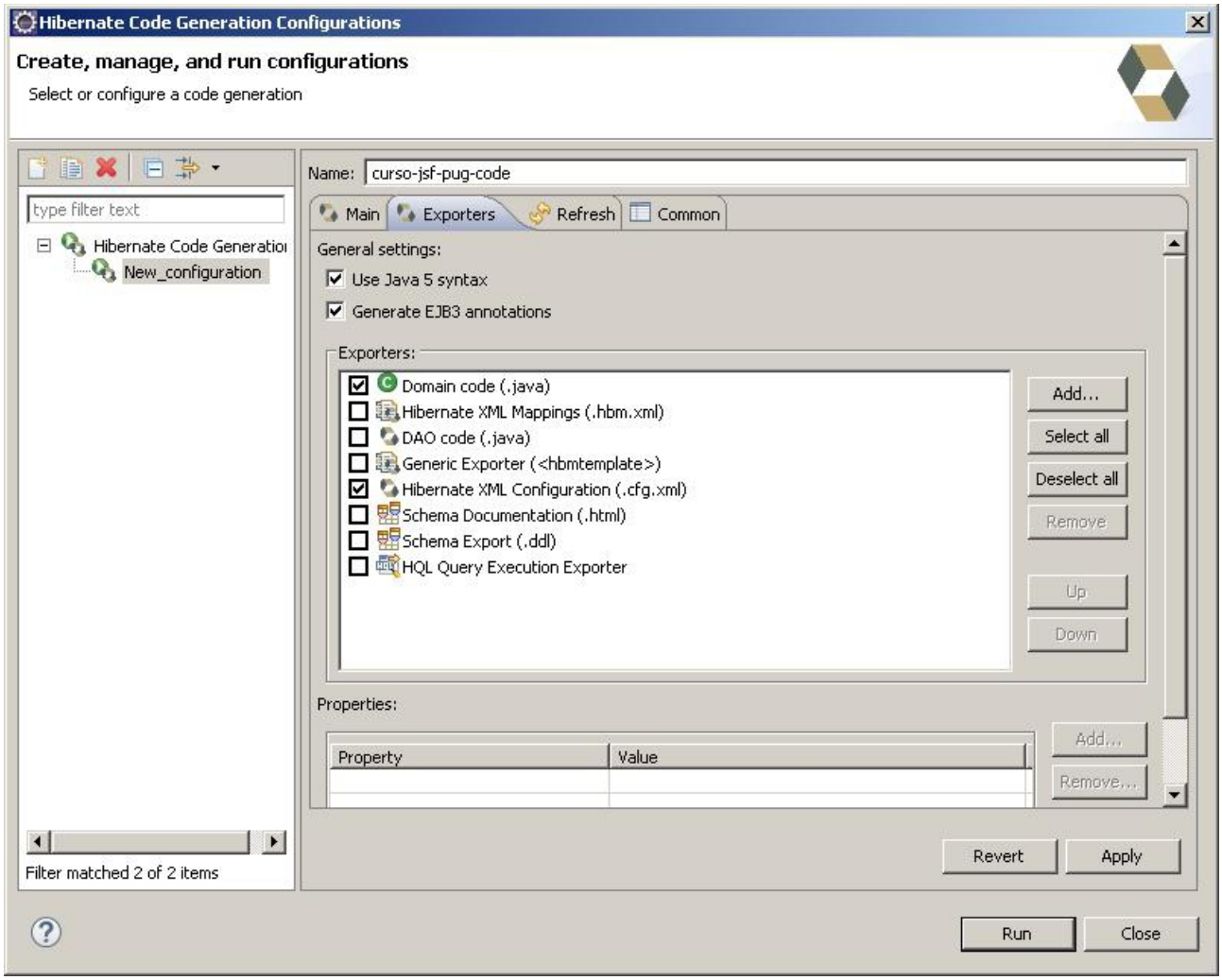
```
<hibernate-reverse-engineering>
	<table-filter match-schema="public" match-name="clientes">
		<meta attribute="scope-class">@Proxy(lazy=false) public</meta>
		<meta attribute="extra-import">
			org.hibernate.annotations.Proxy
		</meta>
	</table-filter>
	<table-filter match-schema="public" match-name="tipos">
		<meta attribute="scope-class">@Proxy(lazy=false) public</meta>
		<meta attribute="extra-import">
			org.hibernate.annotations.Proxy
		</meta>
	</table-filter>
	<table-filter match-schema="public" match-name="usuarios">
		<meta attribute="scope-class">@Proxy(lazy=false) public</meta>
		<meta attribute="extra-import">
			org.hibernate.annotations.Proxy
		</meta>
	</table-filter>
	<table schema="public" name="tipos">
		<column name="id">
			<meta attribute="use-in-tostring">true</meta>
			<meta attribute="use-in-equals">true</meta>
		</column>
	</table>
	<table schema="public" name="tipos">
		<primary-key>
			<generator class="identity"></generator>
```

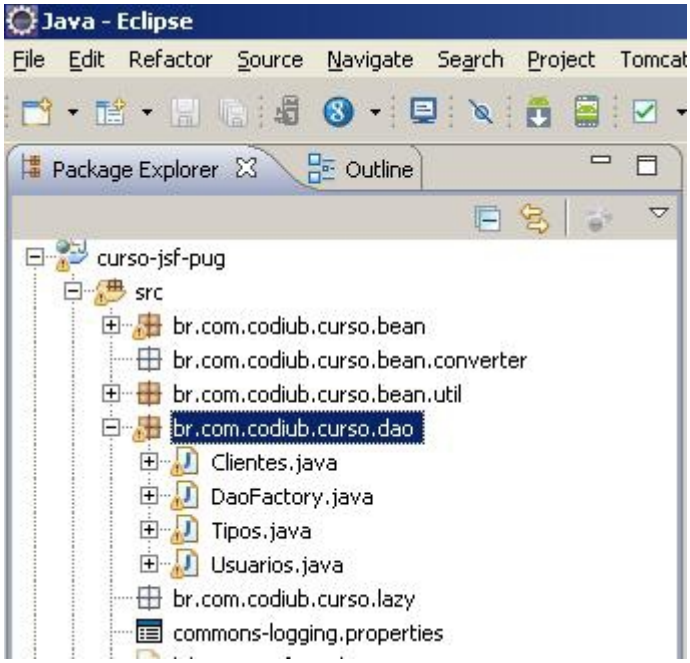
```
                  <key-column name="id" />
            </primary-key>
      </table>
      <table schema="public" name="clientes">
            <primary-key>
                  <generator class="identity"></generator>
                  <key-column name="id" />
            </primary-key>
      </table>
      <table schema="public" name="usuarios">
            <primary-key>
                  <generator class="identity"></generator>
                  <key-column name="id" />
            </primary-key>
      </table>
</hibernate-reverse-engineering>
```



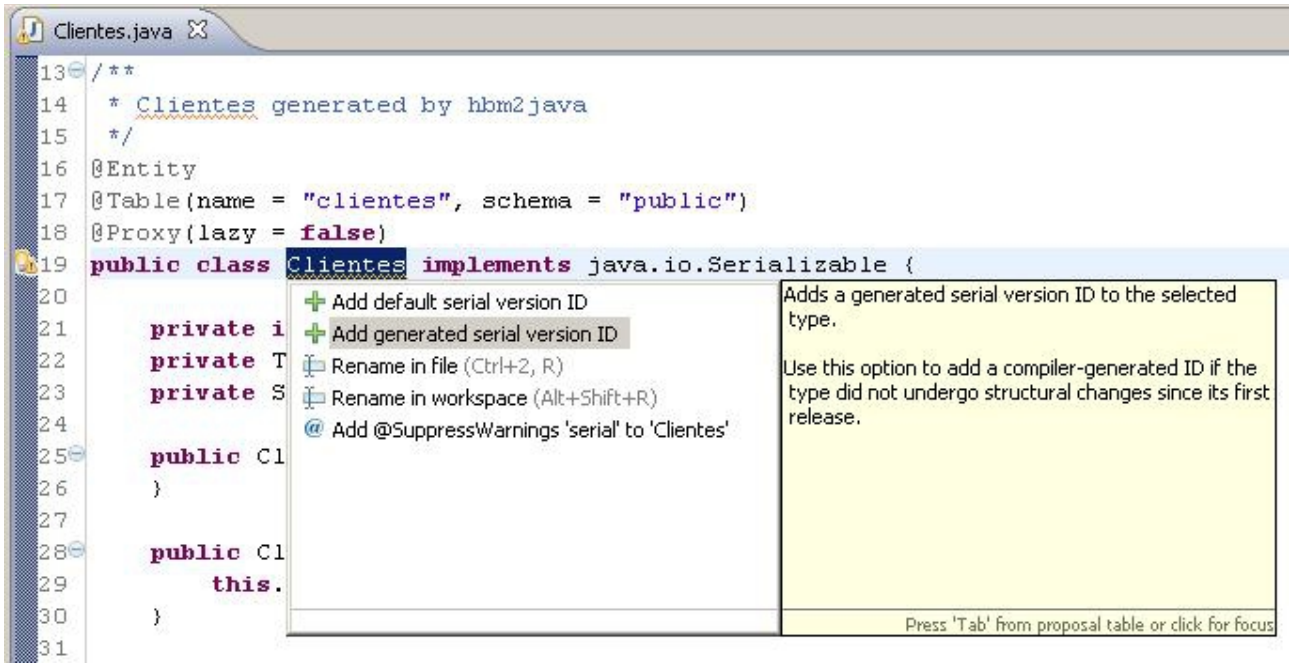
Agora podemos solicitar ao Hibernate Tools que faça a criação das classes de objetos de mapeamento através do menu Run → Hibernate Code Generation.

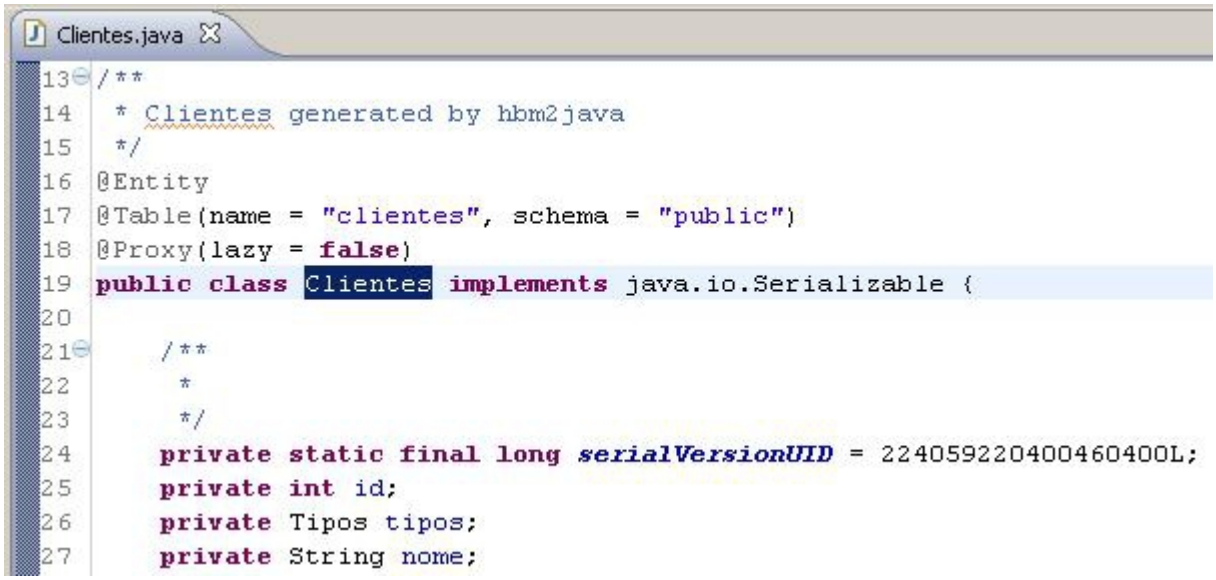

Hibernate Code Generation Configurations
Create, manage, and run configurations
[Exporters]: At least one exporter option must be selected
type filter text
Hibernate Code Generatio
New_configuration
Filter matched 2 of 2 items
Name: curso-jsf-pug-code
Main
Exporters
Refresh
Common
Console configuration: curso-jsf-pug
Output directory: \curso-jsf-pug\src
Browse...
Reverse engineer from JDBC Connection
Package: br.com.codiub.curso.dao
reveng.xml: \curso-jsf-pug\src\hibernate.reveng.xml
Setup...
reveng. strategy:
Browse...
Generate basic typed composite ids
Detect optimistic lock columns
Detect many-to-many tables
Detect one-to-one associations
Use custom templates (for custom file generation)
Template directory:
Filesystem...
Workspace...
Revert
Apply
Run
Close


Hibernate Code Generation Configurations
Create, manage, and run configurations
Select or configure a code generation
type filter text
Hibernate Code Generatio
New_configuration
Filter matched 2 of 2 items
Name: curso-jsf-pug-code
Main
Exporters
Refresh
Common
General settings:
Use Java 5 syntax
Generate EJB3 annotations
Exporters:
Domain code (.java)
Hibernate XML Mappings (.hbm.xml)
DAO code (.java)
Generic Exporter (<hbmtemplate>)
Hibernate XML Configuration (.cfg.xml)
Schema Documentation (.html)
Schema Export (.ddl)
HQL Query Execution Exporter
Add...
Select all
Deselect all
Remove
Up
Down
Properties:
Property
Value
Add...
Remove...
Revert
Apply
Run
Close


Java - Eclipse
File Edit Refactor Source Navigate Search Project Tomcat
Package Explorer
Outline
curso-jsf-pug
src
br.com.codiub.curso.bean
br.com.codiub.curso.bean.converter
br.com.codiub.curso.bean.util
br.com.codiub.curso.dao
Clientes.java
DaoFactory.java
Tipos.java
Usuarios.java
br.com.codiub.curso.lazy
commons-logging.properties

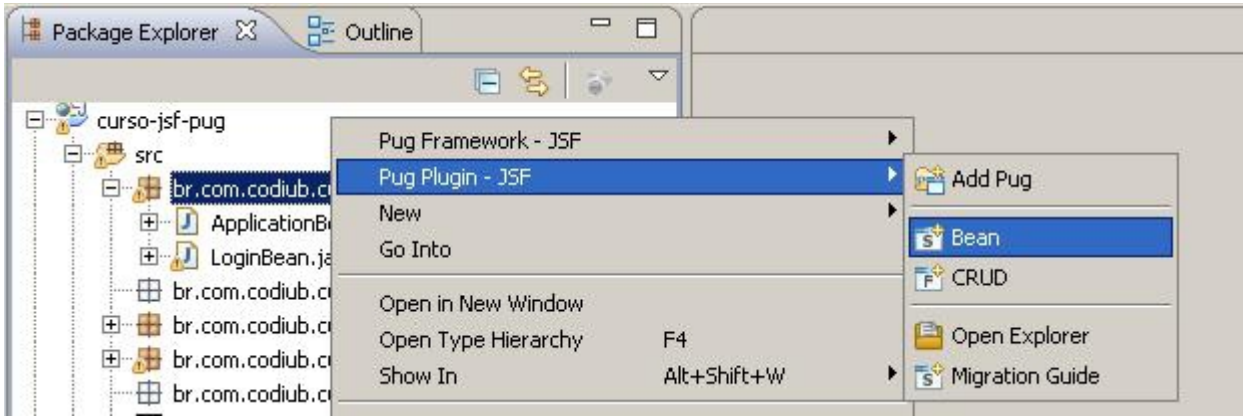
Abra as classes geradas pelo hibernate no pacote “br.com.codiub.curso.dao” e mande adicionar o serial version ID, é importante para o correto funcionamento de serialização do JSF.

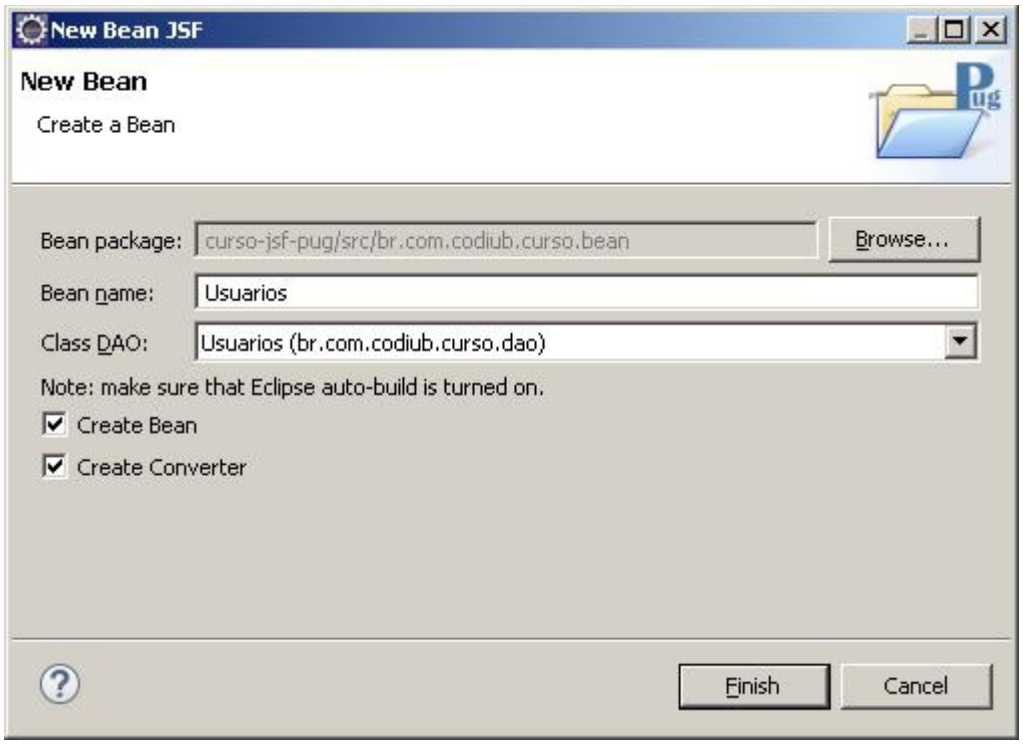




7) Classes Beans

A classe Bean é a responsável pela manipulação do lado do Server e de interação com o XHTML do lado do Client. Assim iremos criar um Bean para manipular cada classe de mapeamento. O Pug facilita esse processo criando uma estrutura de integração.

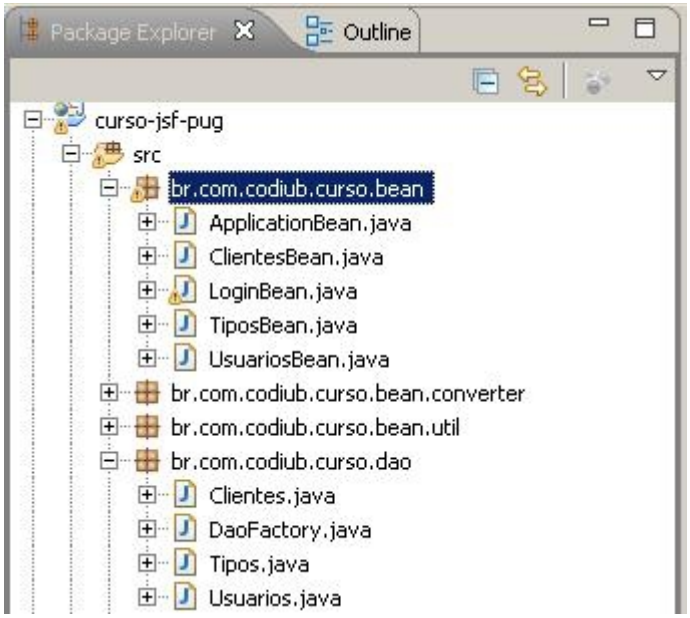


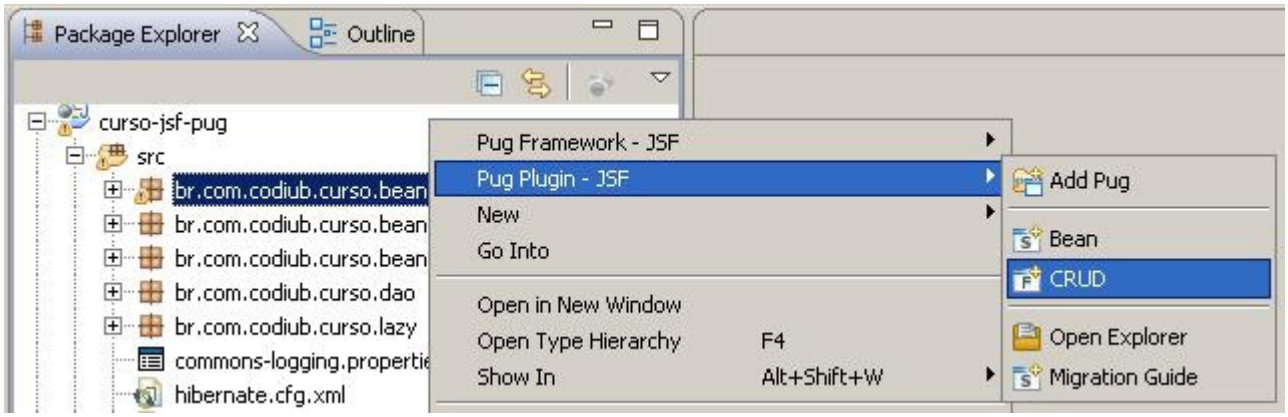


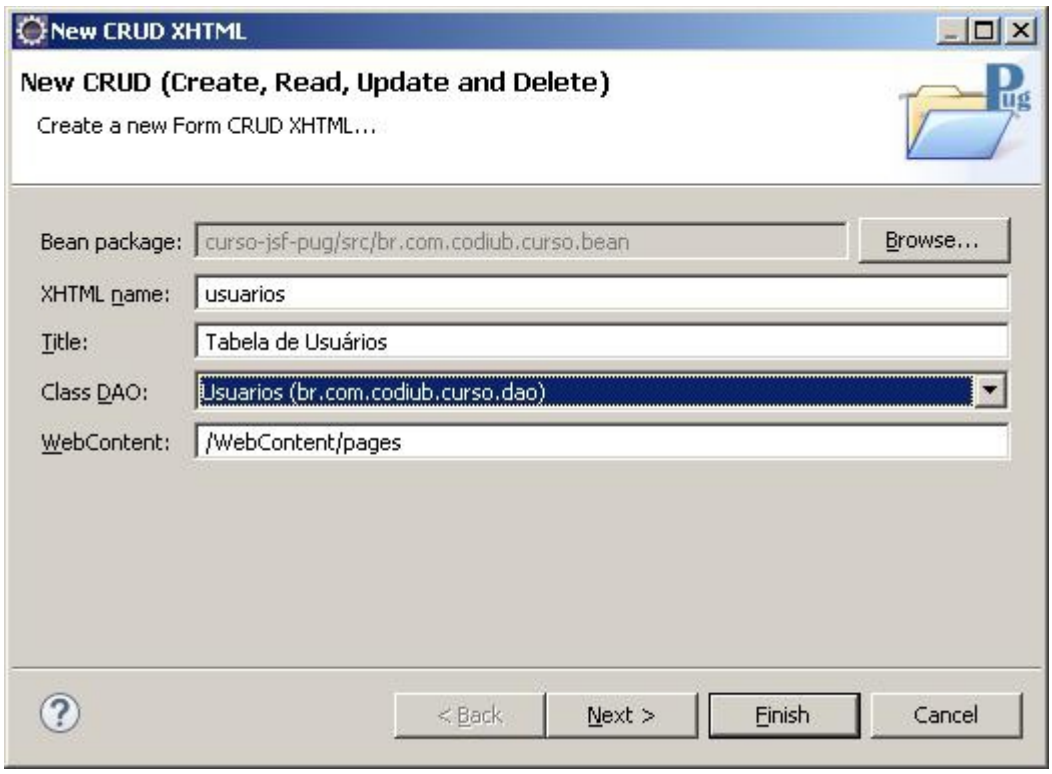
Crie um Bean para: Usuarios, Clientes e Tipos.

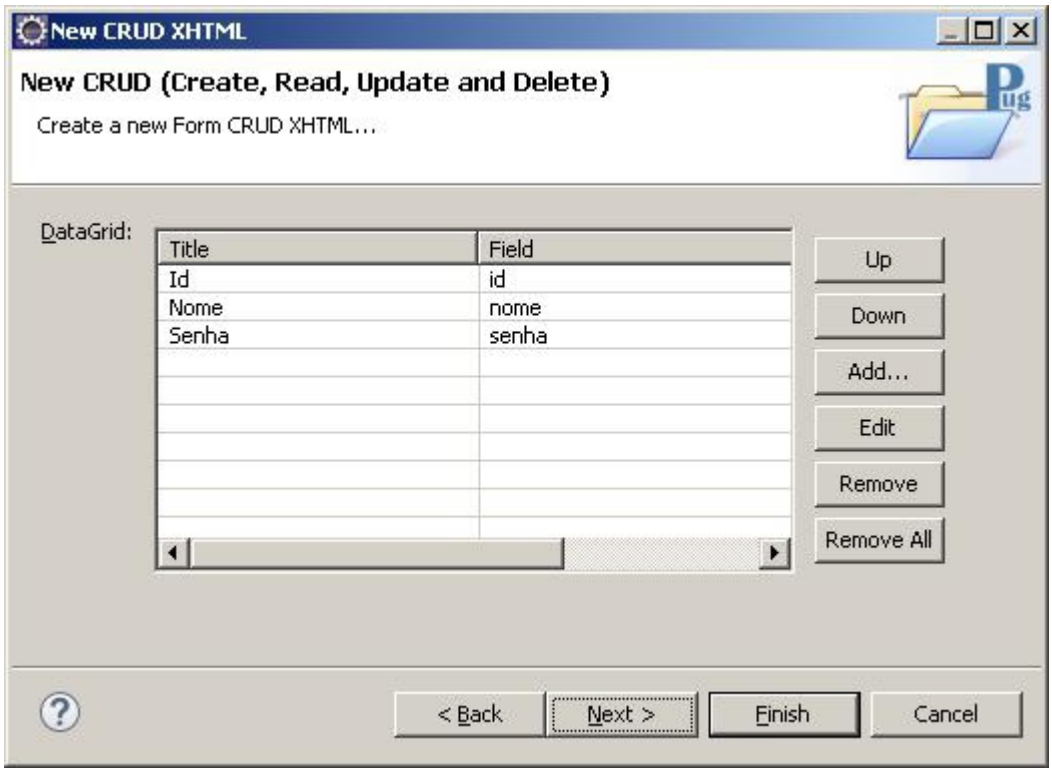
8) XHTML

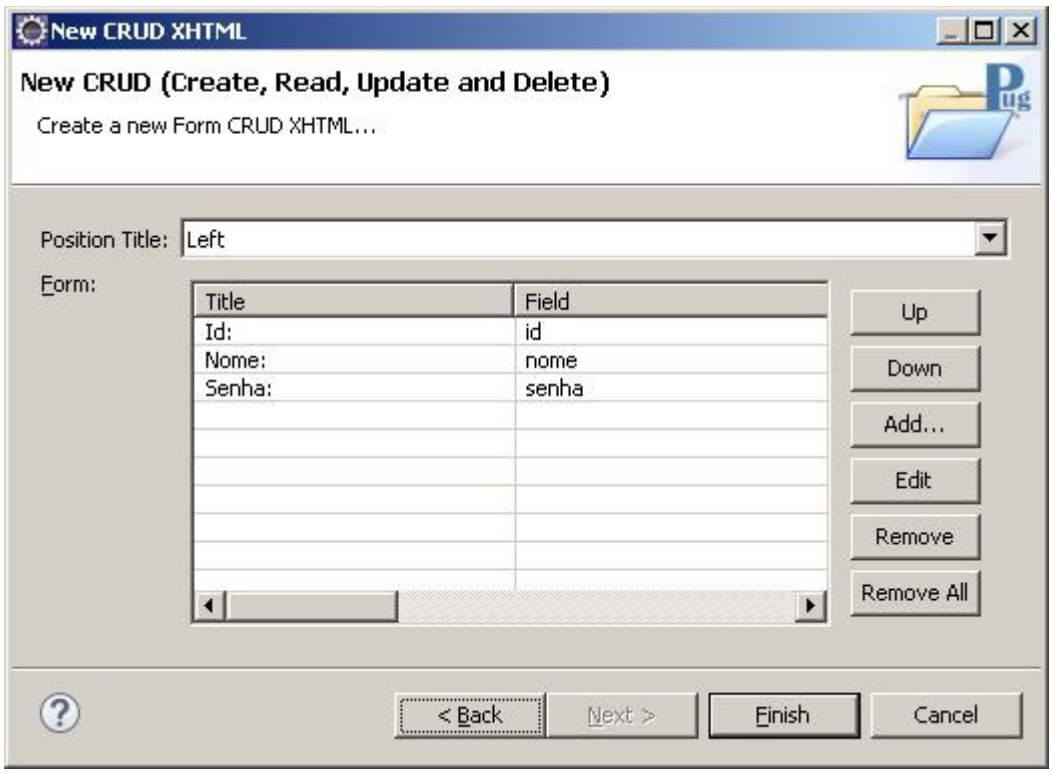
A interface Client é definida através de XHTML, onde teremos a estrutura HTML com componentes do PrimeFaces e integração com os Beans criados no tópico anterior. O Pug cria um CRUD (Create, Read, Update e Delete) contemplando as operações básicas para o usuário.

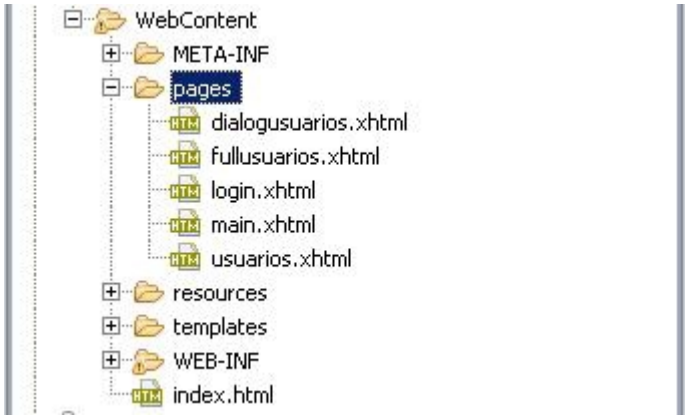


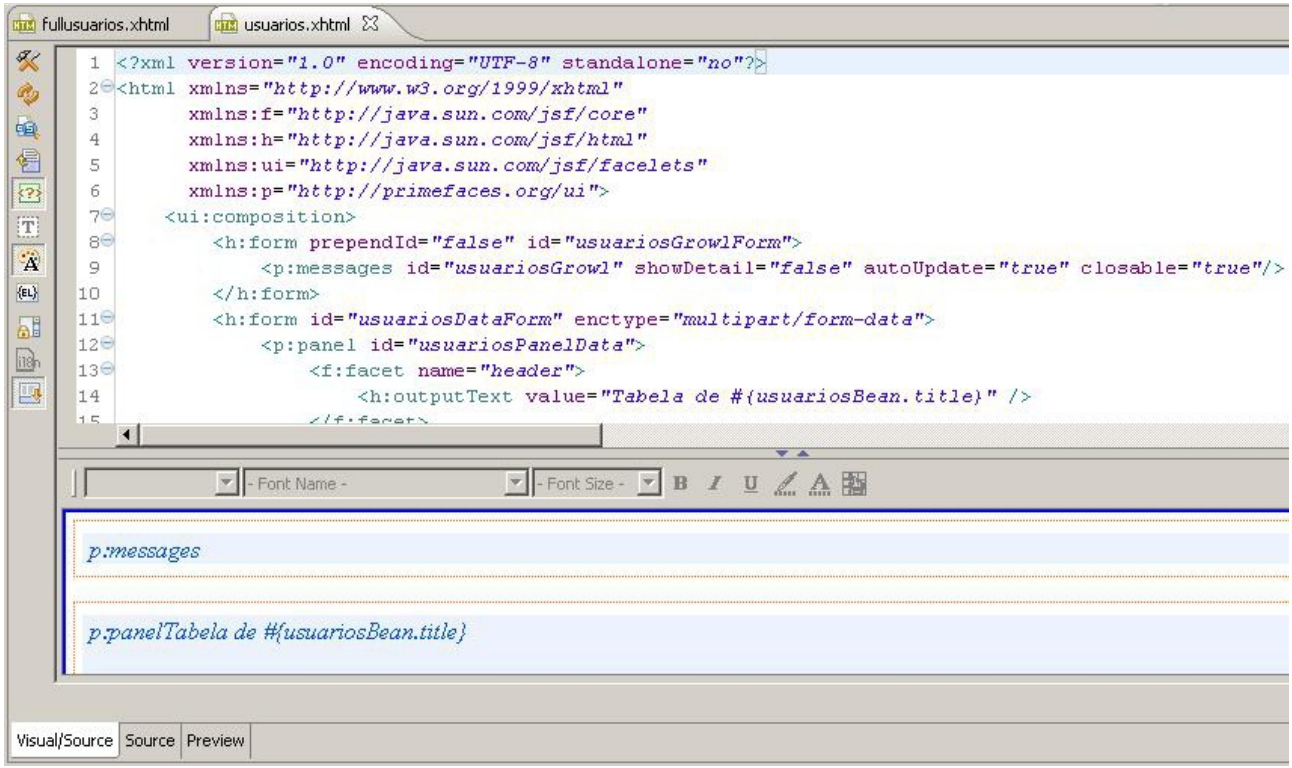


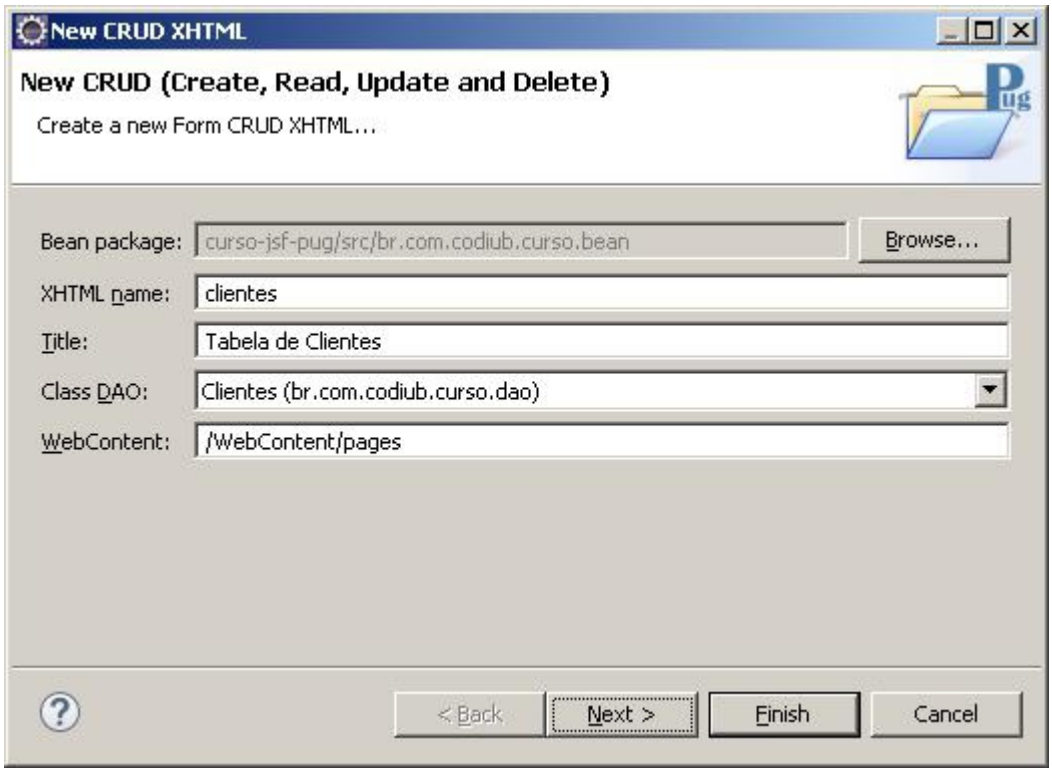

New CRUD XHTML
New CRUD (Create, Read, Update and Delete)
Create a new Form CRUD XHTML...
Bean package:
curso-jsf-pug/src/br.com.codiub.curso.bean
Browse...
XHTML name:
usuarios
Title:
Tabela de Usuários
Class DAO:
Usuarios (br.com.codiub.curso.dao)
WebContent:
/WebContent/pages
< Back
Next >
Finish
Cancel

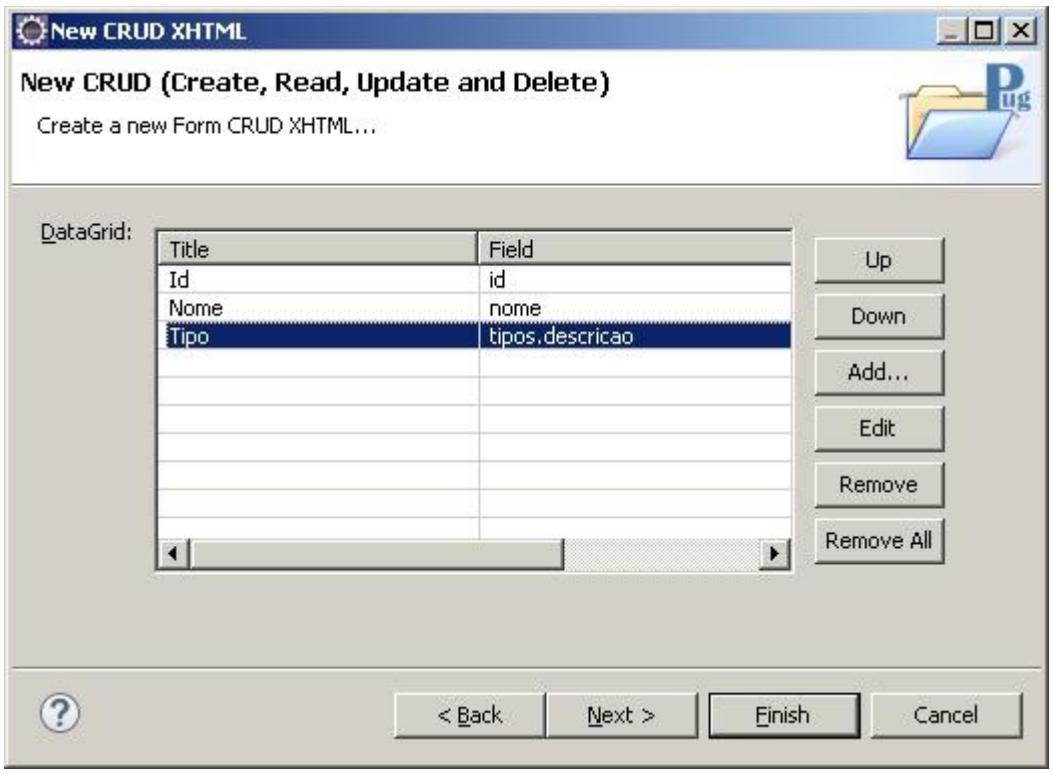

New CRUD XHTML
New CRUD (Create, Read, Update and Delete)
Create a new Form CRUD XHTML...
DataGrid:
Title
Field
Id
id
Nome
nome
Senha
senha
Up
Down
Add...
Edit
Remove
Remove All
< Back
Next >
Finish
Cancel

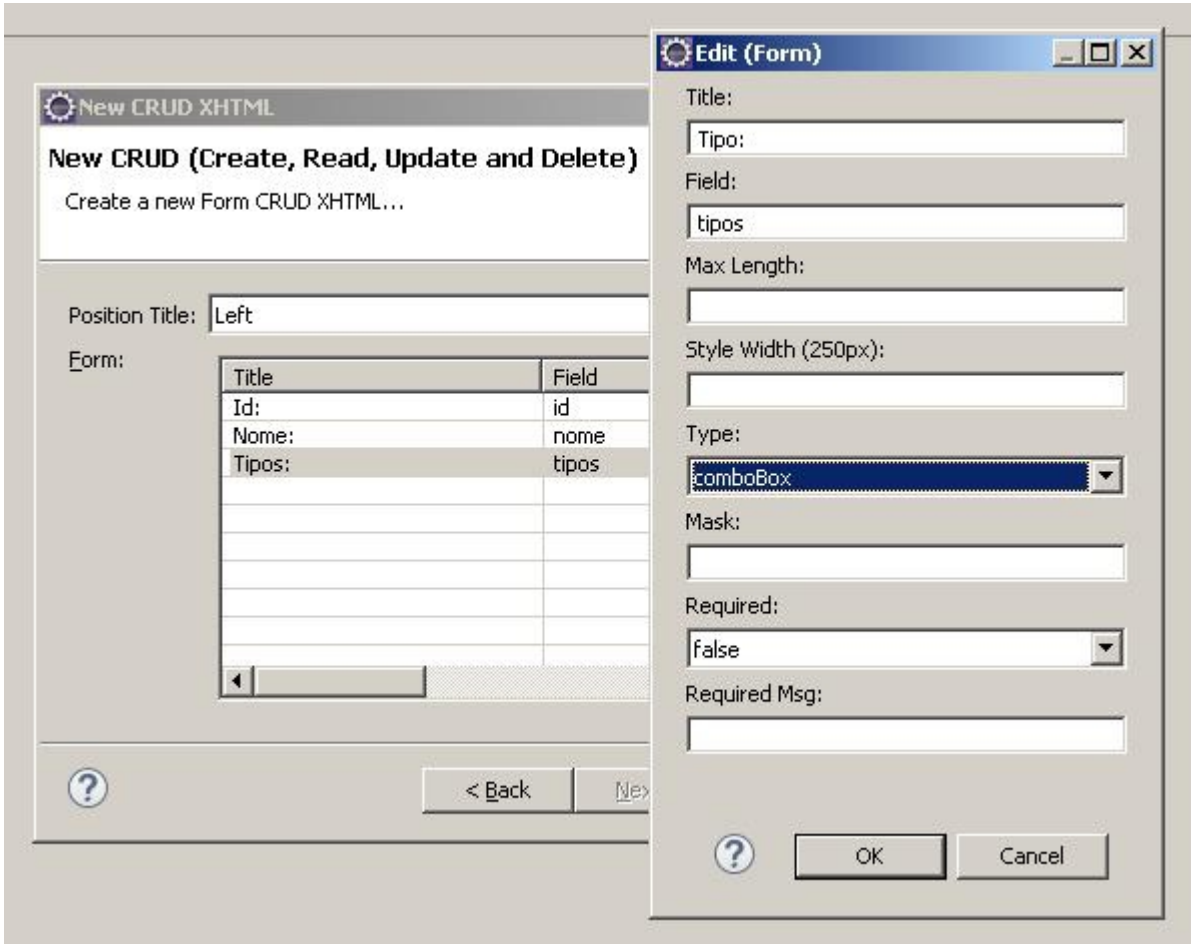

New CRUD XHTML
New CRUD (Create, Read, Update and Delete)
Create a new Form CRUD XHTML...
Position Title:
Left
Form:
Title
Field
Id:
id
Nome:
nome
Senha:
senha
Up
Down
Add...
Edit
Remove
Remove All
< Back
Next >
Finish
Cancel

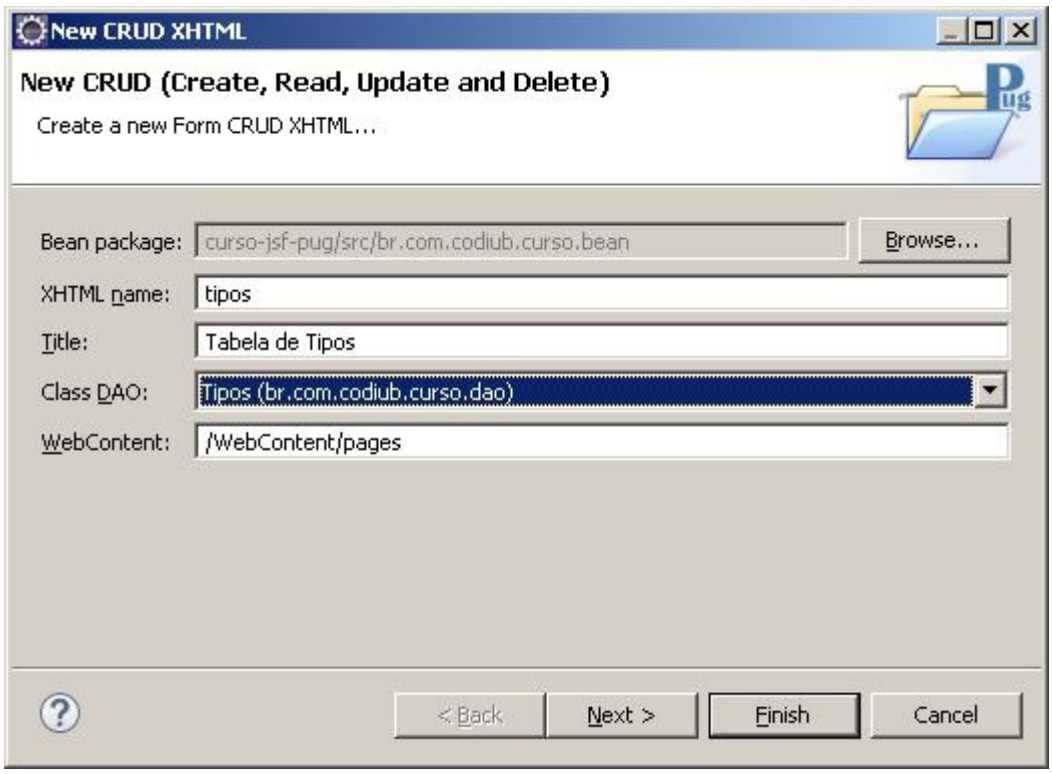

WebContent
META-INF
pages
dialogusuarios.xhtml
fullusuarios.xhtml
login.xhtml
main.xhtml
usuarios.xhtml
resources
templates
WEB-INF
index.html

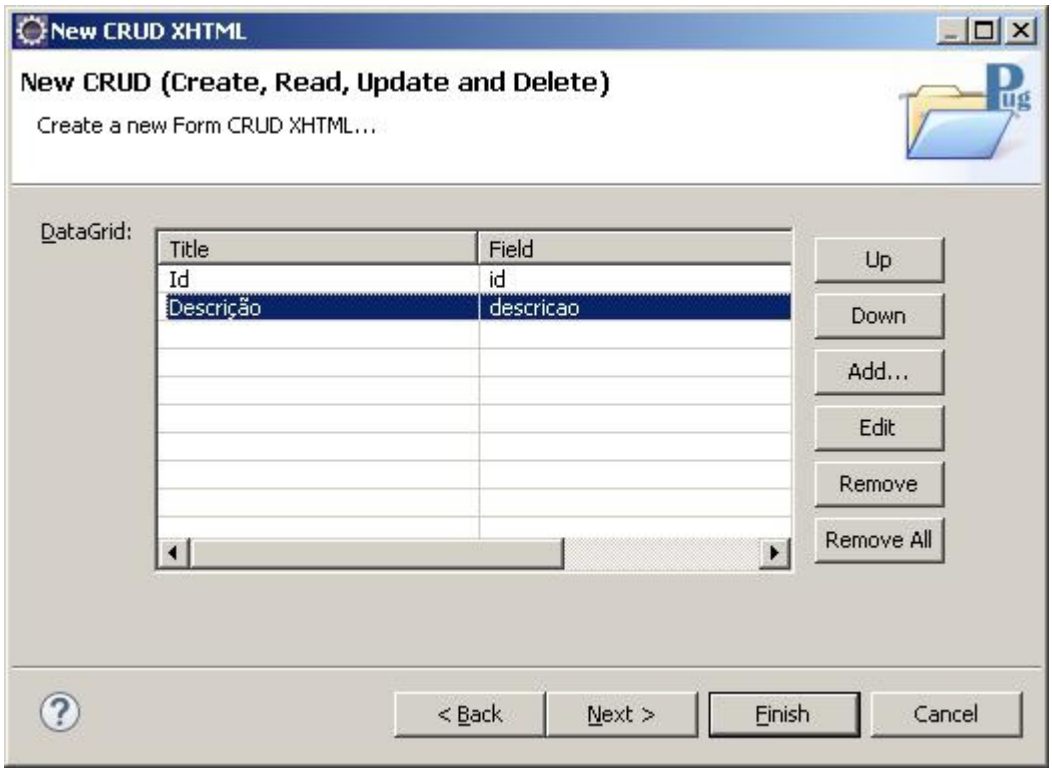

fullusuarios.xhtml
usuarios.xhtml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:h="http://java.sun.com/jsf/html"
xmlns:ui="http://java.sun.com/jsf/facelets"
xmlns:p="http://primefaces.org/ui">
<ui:composition>
<h:form prependId="false" id="usuariosGrowlForm">
<p:messages id="usuariosGrowl" showDetail="false" autoUpdate="true" closable="true"/>
</h:form>
<h:form id="usuariosDataForm" enctype="multipart/form-data">
<p:panel id="usuariosPanelData">
<f:facet name="header">
<h:outputText value="Tabela de #{usuariosBean.title}" />
- Font Name -
- Font Size -
p:messages
p:panelTabela de #{usuariosBean.title}
Visual/Source
Source
Preview

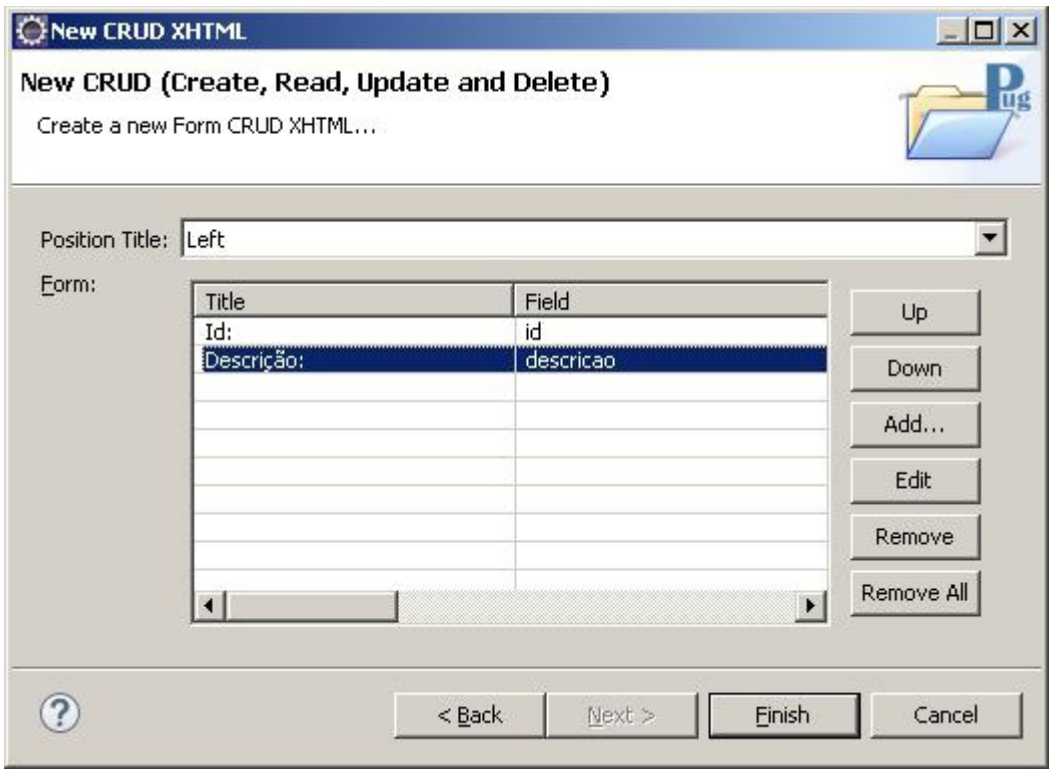

New CRUD XHTML
New CRUD (Create, Read, Update and Delete)
Create a new Form CRUD XHTML...
Bean package:
curso-jsf-pug/src/br.com.codiub.curso.bean
Browse...
XHTML name:
clientes
Title:
Tabela de Clientes
Class DAO:
Clientes (br.com.codiub.curso.dao)
WebContent:
/WebContent/pages
< Back
Next >
Finish
Cancel


New CRUD XHTML
New CRUD (Create, Read, Update and Delete)
Create a new Form CRUD XHTML...
DataGrid:
Title
Field
Id
id
Nome
nome
Tipo
tipos.descricao
Up
Down
Add...
Edit
Remove
Remove All
< Back
Next >
Finish
Cancel


New CRUD XHTML
New CRUD (Create, Read, Update and Delete)
Create a new Form CRUD XHTML...
Position Title:
Left
Form:
Title
Field
Id:
id
Nome:
nome
Tipos:
tipos
< Back
Edit (Form)
Title:
Tipo:
Field:
tipos
Max Length:
Style Width (250px):
Type:
comboBox
Mask:
Required:
false
Required Msg:
OK
Cancel

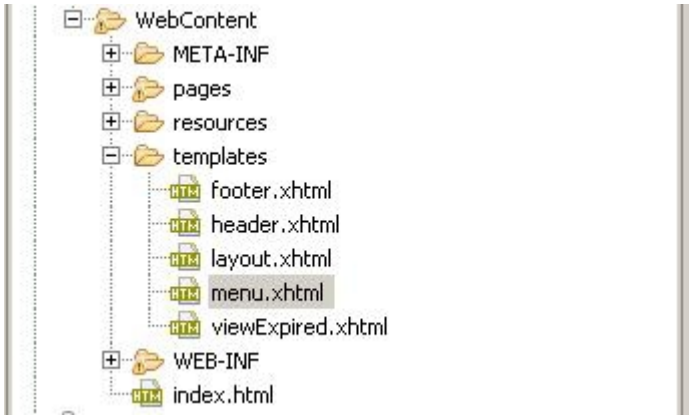

New CRUD XHTML
New CRUD (Create, Read, Update and Delete)
Create a new Form CRUD XHTML...
Bean package:
curso-jsf-pug/src/br.com.codiub.curso.bean
Browse...
XHTML name:
tipos
Title:
Tabela de Tipos
Class DAO:
Tipos (br.com.codiub.curso.dao)
WebContent:
/WebContent/pages
< Back
Next >
Finish
Cancel


New CRUD XHTML
New CRUD (Create, Read, Update and Delete)
Create a new Form CRUD XHTML...
DataGrid:
Title
Field
Id
id
Descrição
descricao
Up
Down
Add...
Edit
Remove
Remove All
< Back
Next >
Finish
Cancel



9) MENU

Para que o usuário possa acessar os CRUD editaremos o “menu.xhtml” e incluiremos as chamadas.



menu.xhtml

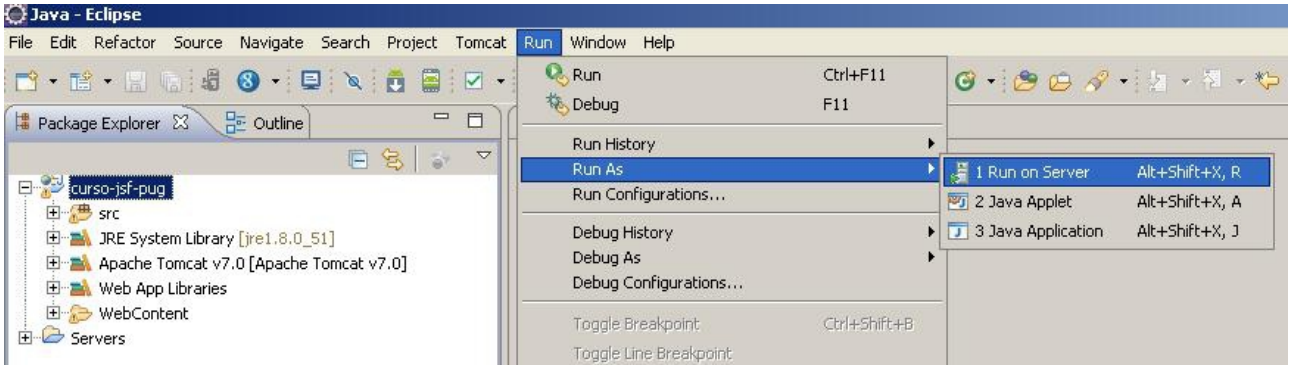
```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <html xmlns="http://www.w3.org/1999/xhtml"
3        xmlns:f="http://java.sun.com/jsf/core"
4        xmlns:h="http://java.sun.com/jsf/html"
5        xmlns:ui="http://java.sun.com/jsf/facelets"
6        xmlns:p="http://primefaces.org/ui">
7      <ui:composition>
8          <p:menubar>
9              <p:menuitem value="Home" icon="ui-icon-home" url="/" />
10
11             <p:submenu label="#{messages.label_menu_cadastros}">
12                 <p:menuitem value="Clientes" url="/pages/fullclientes.xhtml"/>
13                 <p:menuitem value="Tipos" url="/pages/fulltipos.xhtml"/>
14                 <p:menuitem value="Usuários" url="/pages/fullusuarios.xhtml"/>
15             </p:submenu>
16
17         </p:menubar>
18     </ui:composition>
19 </html>
```

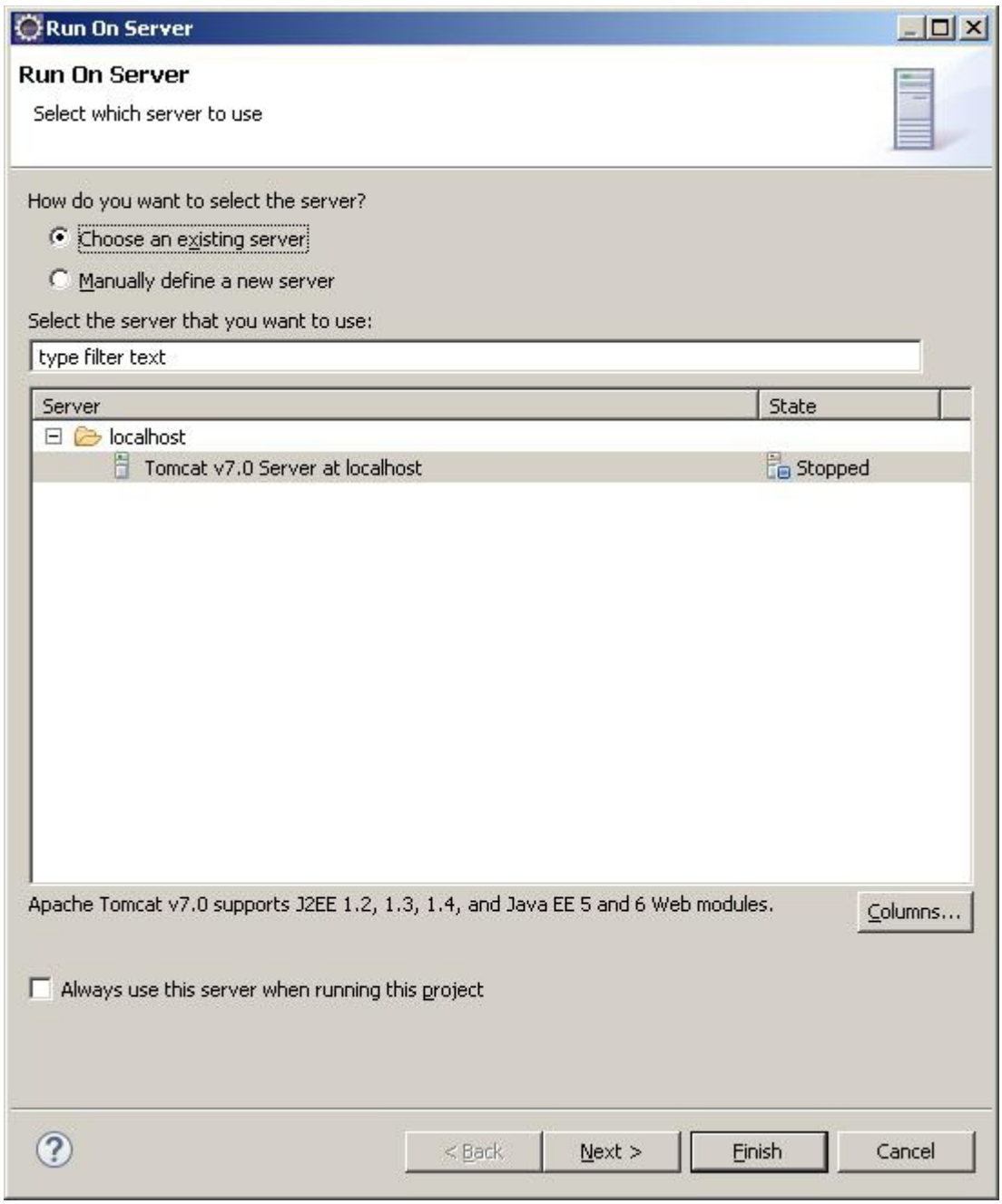
Codificação:

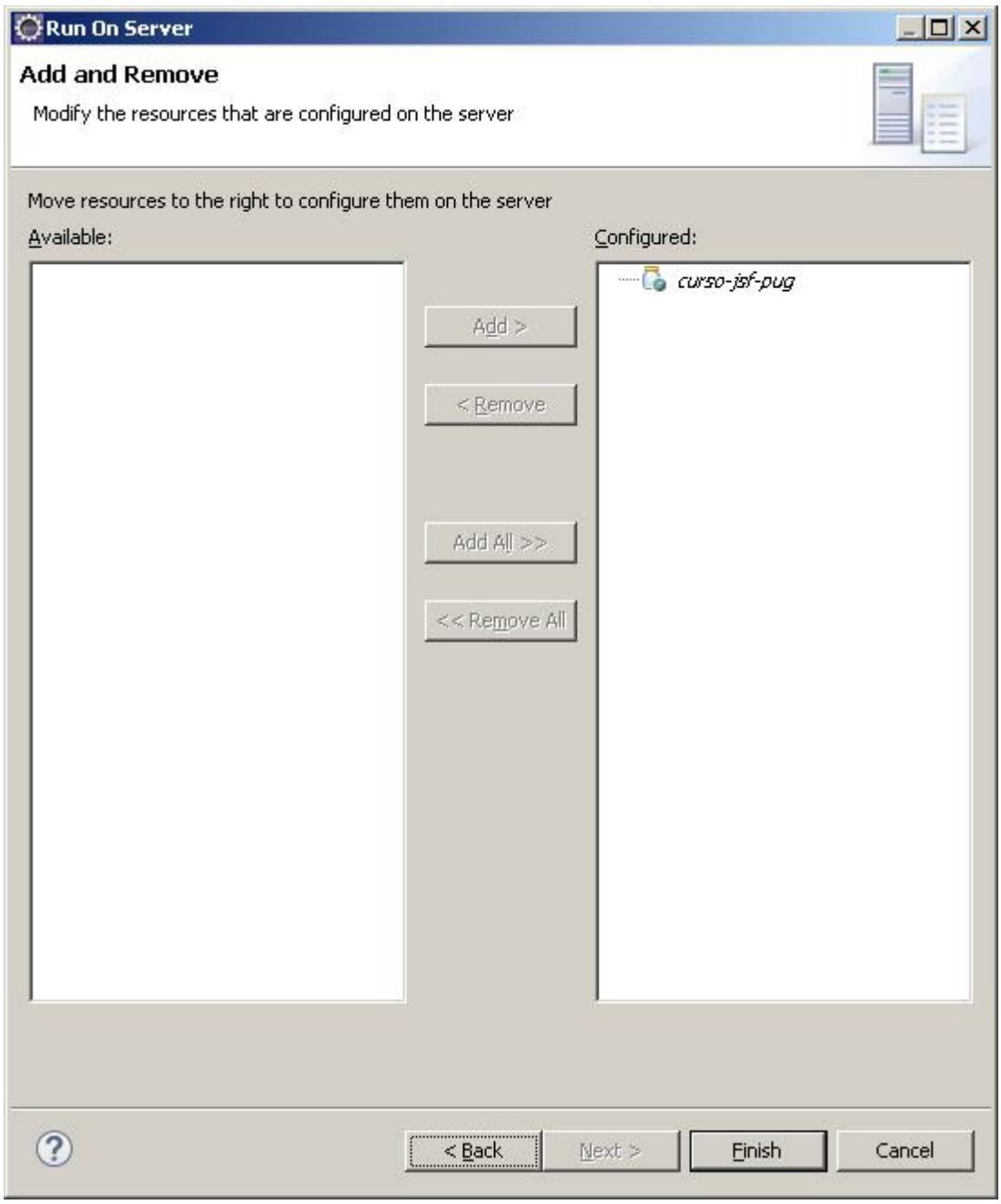
```
<p:submenu label="#{messages.label_menu_cadastros}">
    <p:menuitem value="Clientes" url="/pages/fullclientes.xhtml"/>
    <p:menuitem value="Tipos" url="/pages/fulltipos.xhtml"/>
    <p:menuitem value="Usuários" url="/pages/fullusuarios.xhtml"/>
</p:submenu>
```

10) Executando o projeto

Já podemos executar o projeto para ver como está.



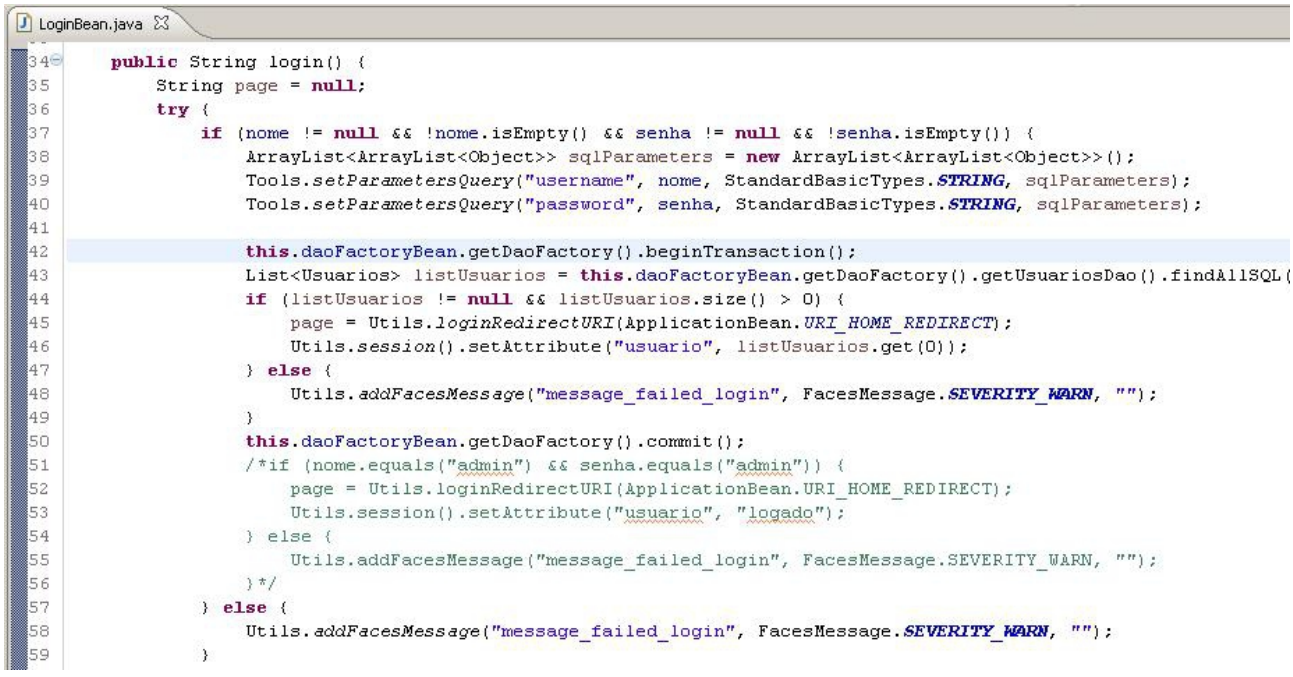

Run On Server
Run On Server
Select which server to use
How do you want to select the server?
Choose an existing server
Manually define a new server
Select the server that you want to use:
type filter text
Server
State
localhost
Tomcat v7.0 Server at localhost
Stopped
Apache Tomcat v7.0 supports J2EE 1.2, 1.3, 1.4, and Java EE 5 and 6 Web modules.
Columns...
Always use this server when running this project
< Back
Next >
Finish
Cancel


Run On Server
Add and Remove
Modify the resources that are configured on the server
Move resources to the right to configure them on the server
Available:
Configured:
curso-jsf-pug
Add >
< Remove
Add All >>
<< Remove All
< Back
Next >
Finish
Cancel

Utilize o usuário: **admin** e senha: **admin** para esse acesso inicial.

11) Validar Login

A validação do acesso no tópico anterior estava fixada no código do Bean “LoginBean”, faça a edição da classe e modifique as instruções de validação para buscar do banco de dados.



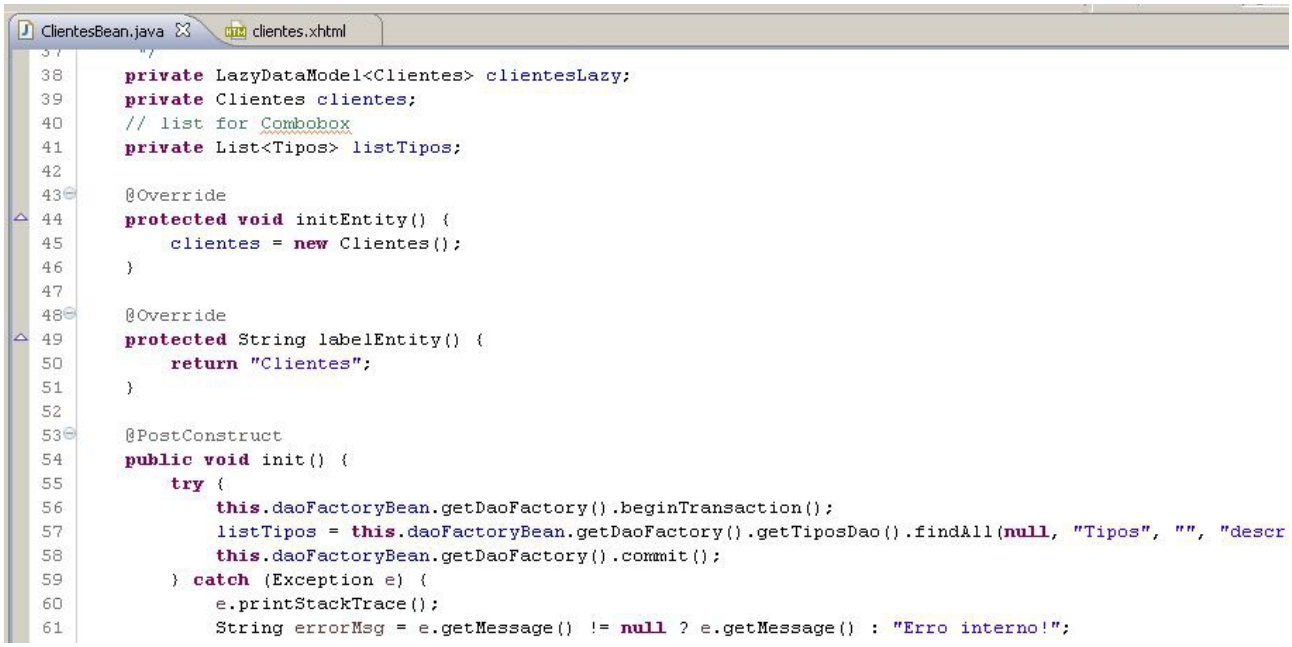
Codificação:

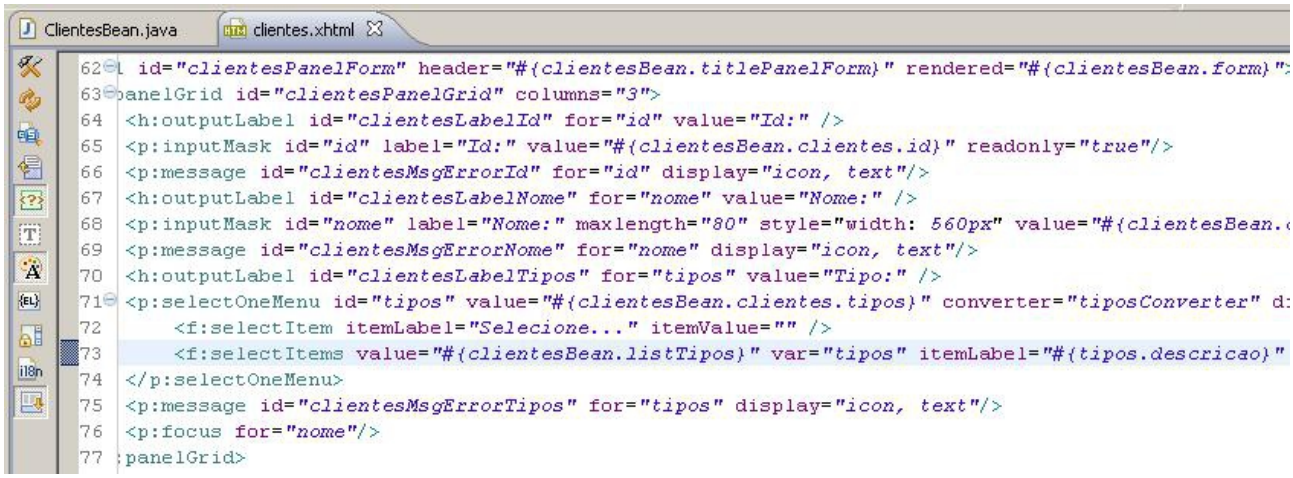
```
                    if (nome != null && !nome.isEmpty() && senha != null && !
senha.isEmpty()) {
                        ArrayList<ArrayList<Object>> sqlParameters = new
ArrayList<ArrayList<Object>>();
                        Tools.setParametersQuery("username", nome,
StandardBasicTypes.STRING, sqlParameters);
                        Tools.setParametersQuery("password", senha,
StandardBasicTypes.STRING, sqlParameters);
```

```
                        this.daoFactoryBean.getDaoFactory().beginTransaction();
                        List<Usuarios> listUsuarios =
this.daoFactoryBean.getDaoFactory().getUsuariosDao().findAllSQL(null, "from
Usuarios where nome = :username and senha = :password", 0, 0, sqlParameters);
                        if (listUsuarios != null && listUsuarios.size() > 0) {
                              page =
Utils.loginRedirectURI(ApplicationBean.URI_HOME_REDIRECT);
                              Utils.session().setAttribute("usuario",
listUsuarios.get(0));
                        } else {
                              Utils.addFacesMessage("message_failed_login",
FacesMessage.SEVERITY_WARN, "");
                        }
                        this.daoFactoryBean.getDaoFactory().commit();
                        /*if (nome.equals("admin") && senha.equals("admin")) {
                              page =
Utils.loginRedirectURI(ApplicationBean.URI_HOME_REDIRECT);
                              Utils.session().setAttribute("usuario", "logado");

                        } else {
                              Utils.addFacesMessage("message_failed_login",
FacesMessage.SEVERITY_WARN, "");
                        }*/
                  } else {
                        Utils.addFacesMessage("message_failed_login",
FacesMessage.SEVERITY_WARN, "");
                  }
```

12) Corrigindo o ComboBox do CRUD de Clientes

O CRUD de clientes está com um problema, pois a codificação gerada pelo Pug deixa apenas um modelo de estrutura do qual tem que ser adaptada. Faça a edição do “ClientesBean”, e ajuste as referências do Combobox.





Segue as codificações:

(ClientesBean):

```
@ManagedBean
@ViewScoped
public class ClientesBean extends BaseBean implements Serializable {

	/**
	 *
	 */
	private static final long serialVersionUID = -7083238750827492631L;
	/**
	 *
	 */
	private LazyDataModel<Clientes> clientesLazy;
	private Clientes clientes;
	// list for Combobox
	private List<Tipos> listTipos;

	@Override
	protected void initEntity() {
		clientes = new Clientes();
```

```
	}

	@Override
	protected String labelEntity() {
		return "Clientes";
	}

	@PostConstruct
	public void init() {
		try {
			this.daoFactoryBean.getDaoFactory().beginTransaction();
			listTipos =
this.daoFactoryBean.getDaoFactory().getTiposDao().findAll(null, "Tipos", "",
"descricao", "", 0, 0);
			this.daoFactoryBean.getDaoFactory().commit();

		} catch (Exception e) {
			e.printStackTrace();
			String errorMsg = e.getMessage() != null ? e.getMessage() :
"Erro interno!";
			errorMsg += " - " +(e.getCause() != null &&
e.getCause().getMessage() != null ? e.getCause().getMessage() : "");
			Utils.addFacesMessage("message_failed_created",
FacesMessage.SEVERITY_ERROR, errorMsg);
		}
	}

	public void save() {
		try {
			String message = "";
			this.daoFactoryBean.getDaoFactory().beginTransaction();
			if (clientes.getId() != null && clientes.getId() > 0) {

	this.daoFactoryBean.getDaoFactory().getClientesDao().merge(clientes);
				message = "message_successfully_updated";
			} else {

	this.daoFactoryBean.getDaoFactory().getClientesDao().save(clientes);
				message = "message_successfully_created";
			}
			this.daoFactoryBean.getDaoFactory().commit();
			Utils.addFacesMessage(message, labelEntity());
		  setForm(false);
		} catch (Exception e) {
			e.printStackTrace();
			String errorMsg = e.getMessage() != null ? e.getMessage() :
"Erro interno!";
			errorMsg += " - " +(e.getCause() != null &&
e.getCause().getMessage() != null ? e.getCause().getMessage() : "");
			Utils.addFacesMessage("message_failed_created",
FacesMessage.SEVERITY_ERROR, errorMsg);
		}
	}

	public void delete() {
		try {
			if (clientes != null) {
				this.daoFactoryBean.getDaoFactory().beginTransaction();

	daoFactoryBean.getDaoFactory().getClientesDao().remove(clientes);
				this.daoFactoryBean.getDaoFactory().commit();
				Utils.addFacesMessage("message_successfully_deleted",
labelEntity());
```

```
                        clientes = null;
                  }
            } catch (Exception e) {
                  e.printStackTrace();
                  String errorMsg = e.getMessage() != null ? e.getMessage() :
"Erro interno!";
                  errorMsg += " - " +(e.getCause() != null &&
e.getCause().getMessage() != null ? e.getCause().getMessage() : "");
                  Utils.addFacesMessage("message_failed_deleted",
FacesMessage.SEVERITY_ERROR, errorMsg);
            }
      }

      public LazyDataModel<Clientes> getClientesLazy() {
            if (clientesLazy == null) {
                  clientesLazy = new ClientesLazy(daoFactoryBean);
            }
            return clientesLazy;
      }

      public void setClientesLazy(LazyDataModel<Clientes> clientesLazy) {
            this.clientesLazy = clientesLazy;
      }

      public Clientes getClientes() {
            return clientes;
      }

      public void setClientes(Clientes clientes) {
            this.clientes = clientes;
      }

      // Replace Foreign and foreign
      public void dialogForeign() {
            Map<String, Object> options = new HashMap<String, Object>();
        options.put("modal", true);
        options.put("draggable", true);
        options.put("resizable", false);
        options.put("contentHeight", 500);
        options.put("contentWidth", 700);
            RequestContext.getCurrentInstance().openDialog("dialogforeign",
options, null);
      }

      // Replace Foreign and foreign
      public void onSelectForeign(SelectEvent event) {
            //Foreign foreign = (Foreign) event.getObject();
            //clientes.setForeign(foreign);
    }

      public void closeDialog() {
            RequestContext.getCurrentInstance().closeDialog(clientes);
      }

      public String getTitle() {
            return labelEntity();
      }

      public List<Tipos> getListTipos() {
            return listTipos;
      }

}
```

(clientes.xhtml):

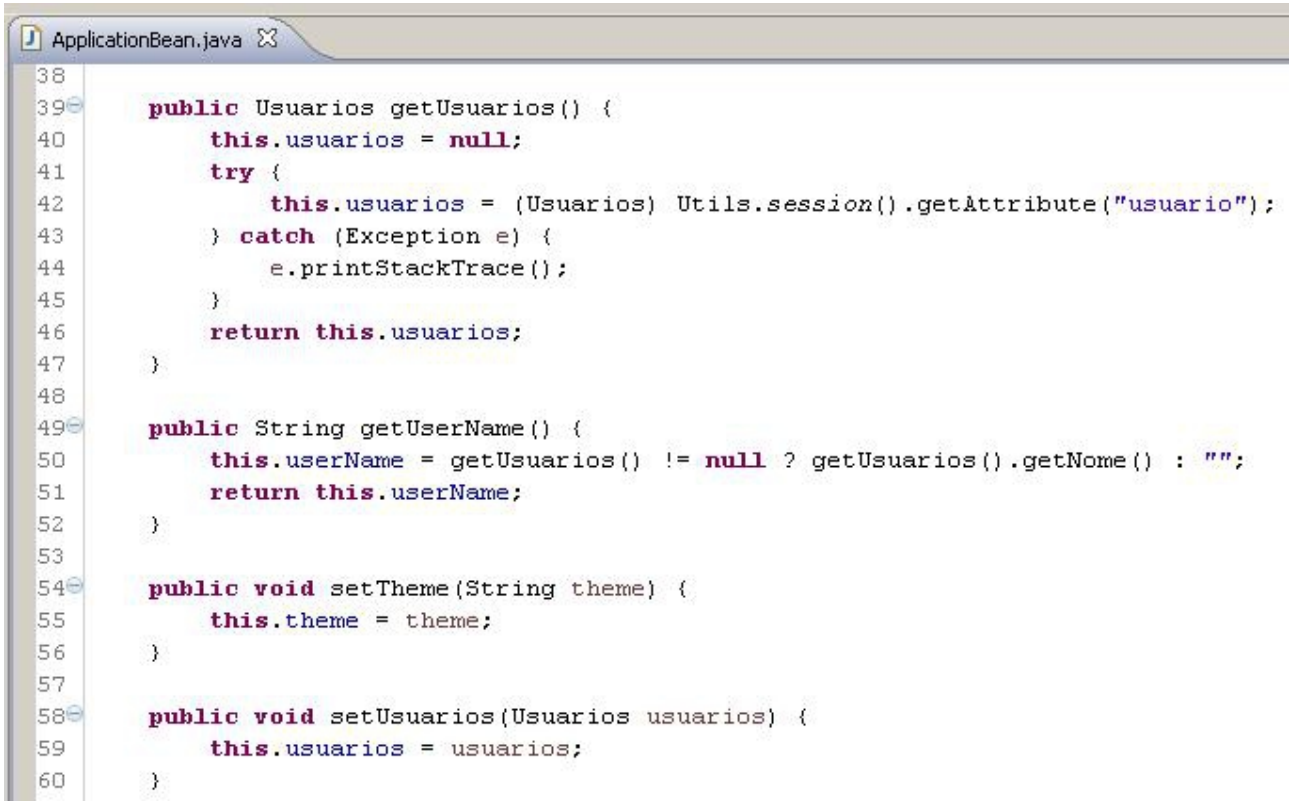
```
                                    <p:selectOneMenu id="tipos"
value="#{clientesBean.clientes.tipos}" converter="tiposConverter"
disabled="#{clientesBean.view}">
                                                <f:selectItem itemLabel="Selecione..."
itemValue="" />
                                                <f:selectItems
value="#{clientesBean.listTipos}" var="tipos" itemLabel="#{tipos.descricao}"
itemValue="#{tipos}"/>
                                    </p:selectOneMenu>
```

13) Finalizando

Nossa última etapa é um ajuste no “ApplicationBean”, como padrão a propriedade “usuarios” está definido como Object, mas vamos alterar para a classe “Usuarios”.

ApplicationBean.java

```
2  * Generator by Pug Framework 1.6.9
4  package br.com.codiub.curso.bean;
5
6  import java.io.Serializable;
14
15 @ManagedBean
16 @ApplicationScoped
17 public class ApplicationBean implements Serializable {
18
19     /**
20      *
21      */
22     private static final long serialVersionUID = -3298819292184517757L;
23     public static String URI_HOME = "/pages/main.xhtml";
24     public static String URI_HOME_REDIRECT = "/pages/main.xhtml?faces-redirect=true";
25     public static String URI_LOGIN = "/pages/login.xhtml";
26     private String theme;
27     private Usuarios usuarios;
28     private String userName;
```



Codificação:

```
@ManagedBean
@ApplicationScoped
public class ApplicationBean implements Serializable {

    /**
       *
       */
      private static final long serialVersionUID = -3298819292184517757L;
      public static String URI_HOME = "/pages/main.xhtml";
      public static String URI_HOME_REDIRECT = "/pages/main.xhtml?faces-
redirect=true";
      public static String URI_LOGIN = "/pages/login.xhtml";
      private String theme;
      private Usuarios usuarios;
      private String userName;

    public String getAppName() {
        return "Modelo JSF-Prime";
    }

    public String getTheme() {
      this.theme = "aristo";
      return this.theme;
    }

      public Usuarios getUsuarios() {
            this.usuarios = null;
            try {
                  this.usuarios = (Usuarios)
Utils.session().getAttribute("usuario");
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return this.usuarios;
```

```
	}

	public String getUserName() {
		this.userName =   getUsuarios() != null ? getUsuarios().getNome() :
"";
		return this.userName;
	}

	public void setTheme(String theme) {
		this.theme = theme;
	}

	public void setUsuarios(Usuarios usuarios) {
		this.usuarios = usuarios;
	}

	public void setUserName(String userName) {
		this.userName = userName;
	}

}
```