INSTALACION DE UN SERVER DE BD POSTGRES, APACHE 2.0.55+JBOSS 4.0.4
PARA TRABAJO EN UN ENTORNO DE DESARROLLO J2EE
CON NETBEANS 6 Y UML BAJO UBUNTU 7.04

Miguel Abarca Castro
Prof. Gustavo Donoso M.

INDICE DE CONTENIDOS

1. Apache 2.0.55

1.1. Paquetes necesarios

Para comenzar la instalación de apache se requiere de algunas librerías, estas se deben instalar ejecutando los siguientes paquetes:

- **$ sudo apt-get install gcc**
- **$ sudo apt-get install linux-kernel-headers**
- **$ sudo apt-get install build-essential**
- **$ sudo apt-get install g++**
- **$ sudo apt-get install automake**
- **$ sudo apt-get install cc**

1.2. Instalación

Una vez instalados estos paquetes entramos al correspondiente directorio vía consola y ejecutamos los siguientes comandos como root, el directorio /usr/local/apache2 corresponde al lugar donde deseas instalarlo, en este caso es ese

- **$ sudo ./configure –prefix=/usr/local/apache2/**
- **$ sudo make**
- **$ sudo make install**

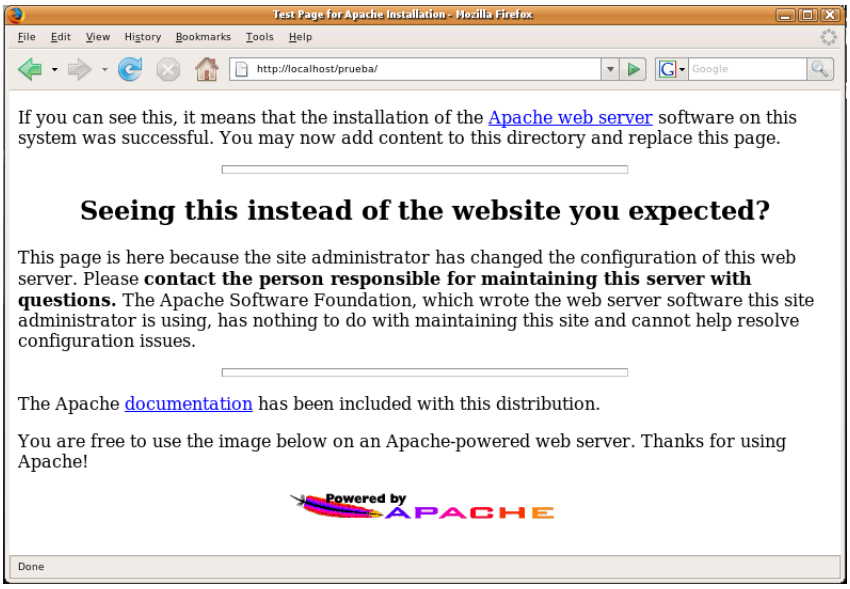
Una vez ejecutados todos estos pasos deberíamos tener el servidor web instalado correctamente, para iniciar el servidor debemos entrar al directorio correspondiente donde instalamos y realizar los siguientes pasos:

- **$ cd /usr/local/apache2/**
- **$ cd bin**
- **$ sudo ./apachectl start**

Cuando hallamos iniciado el servidor, en nuestro navegador web ingresar la siguiente ruta:

- **http://localhost**
- **http://127.0.0.1**

Si vemos el logo de apache tenemos todo instalado correctamente.



Es bueno señalar también que como apache abre el puerto 80, podemos acceder a nuestro servidor de forma interna y externa por medio de nuestra IP local o publica, para la forma publica debemos abrir el puerto en nuestro router.

2. JDK+JRE+JEE

2.1. Archivos necesarios

Los archivos necesarios para realizar la instalación de los paquetes JAVA deben ser descargados de la pagina www.java.sun.com y se deben bajar los instaladores binarios, los nombres de estos son los siguientes:

- **jdk-6u3-linux-i586.bin**
- **jre-6u3-linux-i586.bin**
- **java-tools-bundle-update3-beta-linux.sh**

2.2. Instalación de Java Runtime Environment (JRE)

Solo se deben seguir estos pasos para la correcta instalación:

a) En una terminal, nos vamos a la carpeta donde hemos descargado **.bin**:

$ cd <carpeta>

b) Le damos permisos de ejecución al archivo:

$ chmod +x jre-6-linux-i586.bin

c) Luego lo instalamos:

$ sudo ./jre-6-linux-i586.bin

d) Seguimos las instrucciones que van apareciendo en pantalla.

e) Movemos la carpeta creada después de la instalación (llamada jre1.6.0) a una más apropiada, sino existe el directorio jvm lo creamos con sudo:

$ sudo mv jre1.6.0 /usr/lib/jvm

f) Seteamos el nuevo Java como una de las “alternativas de java”:

$ sudo update-alternatives --install “/usr/bin/java” “java” “/usr/lib/jvm/jre1.6.0/bin/java” 1

g) Ahora seteamos la “nueva alternativa” como la real de Java:

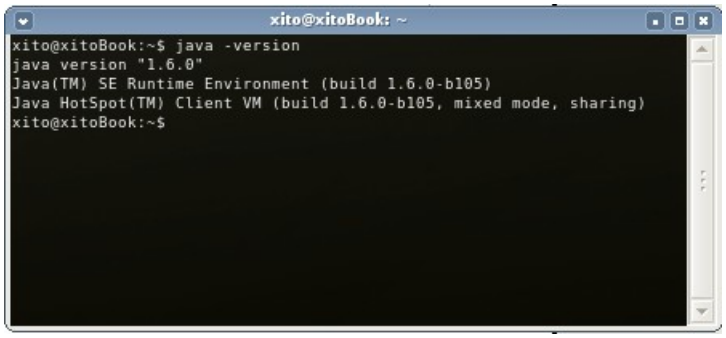
$ sudo update-alternatives --set java /usr/lib/jvm/jre1.6.0/bin/java

h) Para comprobar si tenemos la versión 1.6.0, tipeamos en el terminal:

$ java -version

Después de esta intrucción, tendrá que aparecer algo así:

java version “1.6.0”
Java(TM) SE Runtime Environment (build 1.6.0-b105)
Java HotSpot(TM) Client VM (build 1.6.0-b105, mixed mode)



2.3. Instalación de Java Development Kit (JDK)

a) En un terminal, nos vamos a la carpeta donde hemos descargado .bin:

$ cd <carpeta>

b) Le damos permisos de ejecución al archivo:

$ chmod +x jdk-6ul-linux-i586.bin

c) Luego instalamos:

$ sudo ./jdk-6u1-linux-i586.bin

d) Seguimos las instrucciones que van apareciendo en pantalla, que son 2 o 3.

e) Movemos la carpeta creada después de la instalación (llamada jdk1.6.0_01) a una más apropiada:

$ sudo mv jdk1.6.0_01 /usr/lib/jvm

f) Seteamos el nuevo Java como una de las “alternativas de Java” (todo en la misma linea):

$ sudo update-alternatives --install “/usr/bin/java” “java” “/usr/lib/jvm/jdk1.6.0_01/bin/java” 1

g) Ahora, seteamos la “nueva alternativa” como la real de Java:

$ sudo update-alternatives --set java /usr/lib/jvm/jdk1.6.0_01/bin/java

h) Para comprobar si tenemos la versión 1.6.0, tipeamos en terminal:

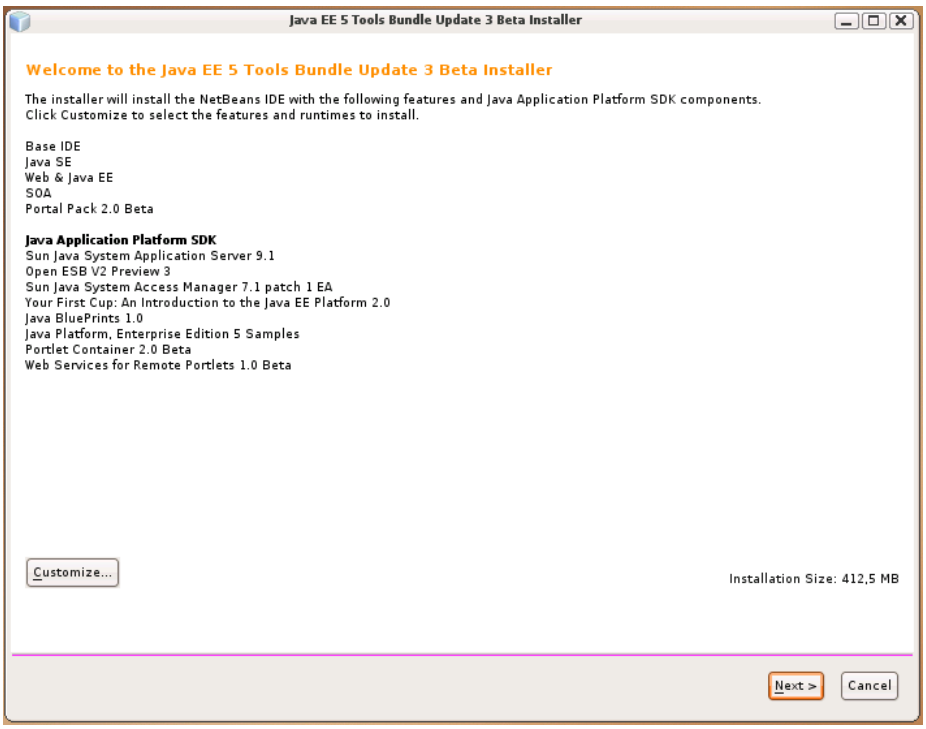
$ java -version

2.4. Instalación de Java Enterprise Edition (JEE)

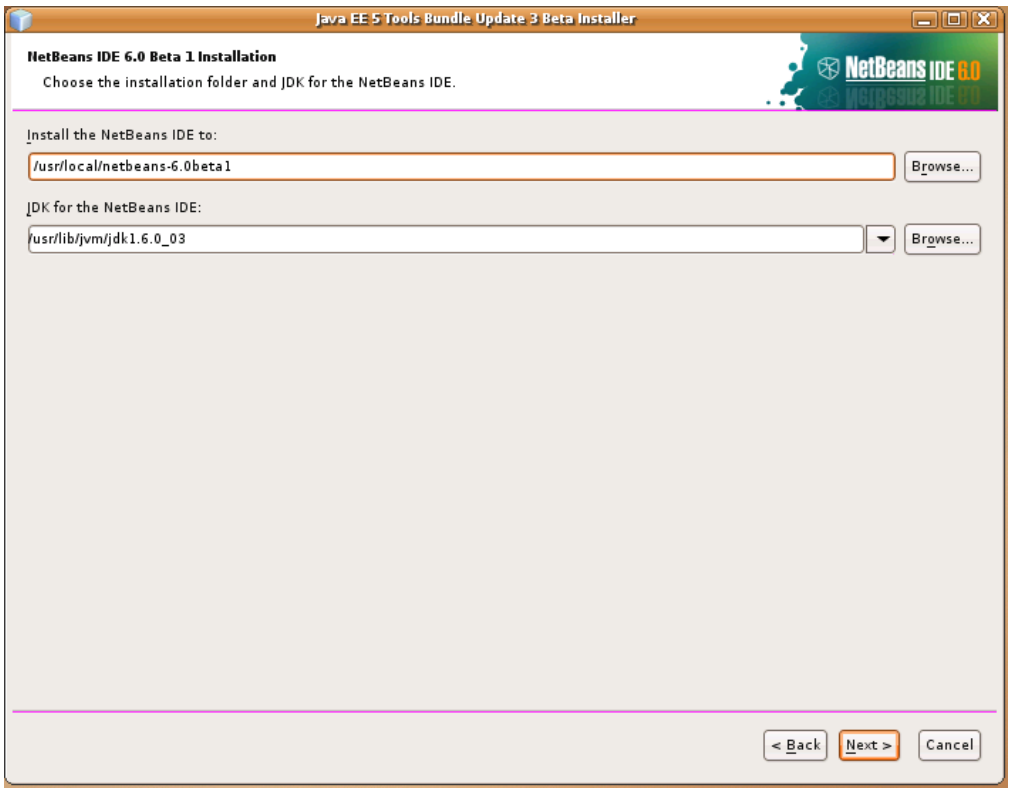
Via consola solamente entrar al directorio donde se descargo y ejecutar el siguiente comando

- **$ sudo sh ./java-tools-bundle-update3-beta-linux.sh**

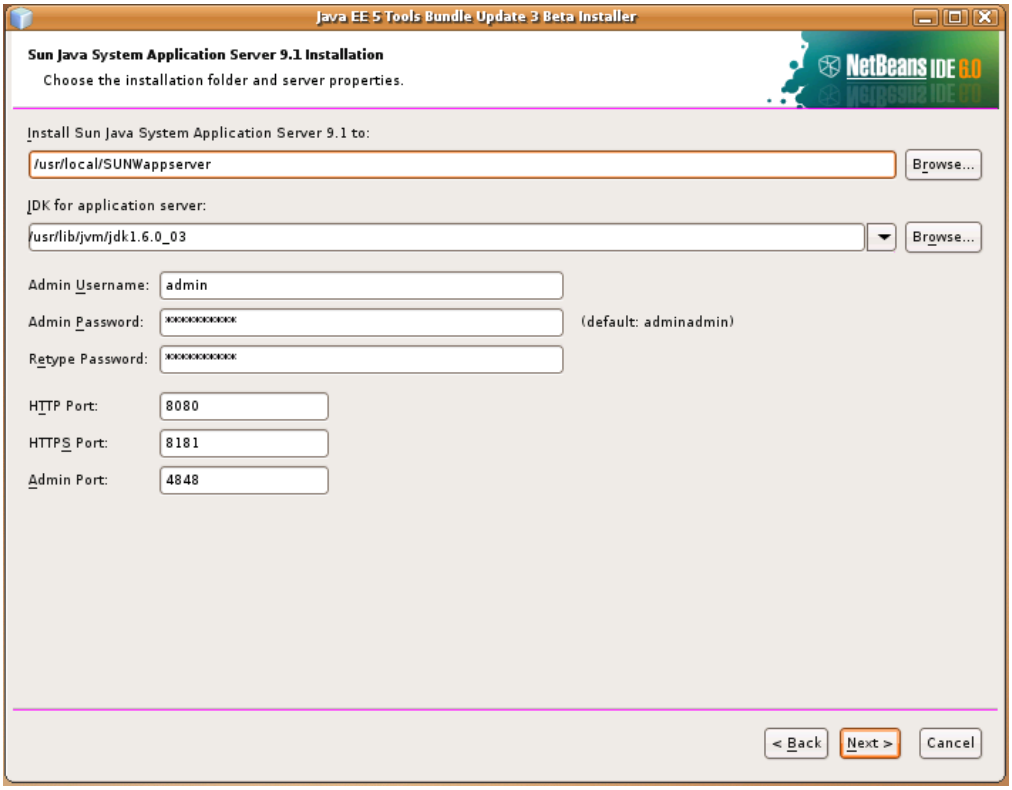
Cuando comience la instalación aparcera la siguiente ventana:



Seleccionar los componentes que desean instalar presionando el botón “Customize” y presionar NEXT para continuar la instalación y seleccionar directorios de instalación.



Presionar NEXT y deberemos poder iniciar la instalación, si seleccionaron instalar el Java System Aplication Server, les aparecerá la siguiente ventana en la cual no deben cambiar nada y solamente presionar NEXT para comenzar la instalación.



Con eso finalizamos la instalación de los paquetes necesarios para el desarrollo de aplicaciones JEE.

2.5. Definición de Variables Globales

La definición de estas variables globales es muy importante ya que serán requeridas al memento de ejecutar el servidor JBOSS, para eso debemos ejecutar el siguiente comando y agregar las siguientes lineas al archivo ***environment***.

- **$ sudo gedit /etc/environment**

Agregar las siguientes lineas (el directorio /usr/jvm/ corresponde al lugar donde instalaron los paquetes java).

```
JAVA_HOME=/usr/java/jdk1.6.0_03/
CLASSPATH=/usr/java/jdk1.6.0_03/lib/
```

Con esto damos por finalizada la instalación y configuración de todos los paquetes JAVA necesarios para el desarrollo JEE, a continuación nos centraremos en realizar la instalación del servidor JBOSS para poder ejecutar nuestras aplicaciones y poder acceder a ellas de forma local o externa por medio de Apache.

3.JBOSS 4.0.4

3.1. Referencia rapida

Revisión rápida de lo que debemos hacer:

1. Descargar mod_jk 1.2.x (http://tomcat.apache.org/download-connectors.cgi)
2. Cambiar la configuración del Apache para incluir la configuración del mod_jk.
3. Crear la configuración del mod_jk
4. Configurar los workers del mod_jk (los que indicaran los nodos que usa el apache)
5. Configurar las ULIs del Apache por medio del mod_jk
6. Reiniciar el Apache
7. Configurar Tomcat
8. Reiniciar JBoss
9. Probar.

3.2. Que es mod_jk?

MOD_JK es un conector que permite a nuestro JBOSS interactuar con servidores web como Apache, Netscape, iPlanet, SunOne e incluso IIS.

La principal funcionalidad de este módulo es permitir a servidores de aplicaciones o al servidor JBOSS enlazarse con un servidor web. Este servidor web, típicamente el servidor HTTP Apache, introduce una mayor gestión en las conexiones de los clientes y mayor la seguridad en las transacciones del sistema. Así mismo se puede enlazar varias instancias al servidor web permitiendo así una mayor tolerancia a errores y aligerar la carga en los servidores JAVA.

3.3. Archivos necesarios

Para poder iniciar la instalación requerimos que se descarguen los archivos de sus respectivos sitios web oficiales.

- **jboss-4.0.4.GA.zip (**www.jboss.org)
- **mod_jk-1.2.25-httpd-2.2.4.so (versiones menores tienen errores)**
  (http://apache.freeby.pctools.cl/tomcat/tomcat-connectors/jk/binaries/linux/jk-1.2.25/)

Utilizaremos esta versión de JBOSS debido a que es la mas estable probada, y porque cumple con todos los requerimientos necesarios para el desarrollo de aplicaciones JEE. Además el modulo mod_jk ke usaremos es el compatible con nuestra versión de apache y no contiene errores.

3.4. Integración con Apache 2.0.55

Para poder tener éxito en la integración de estos dos servidores se recomienda mucho cuidado en la creación de los nuevos archivos y en la modificación de los archivos de configuración de Apache.

PASO #1: Descargar el Apache

Una vez descargado e instalado el apache, no se necesita ninguna configuracion especial. Llamaremos APACHE_HOME a la carpeta o directorio donde se encuentra instalado (siendo esta la ruta completa, en mi caso /home/xito/apache2).

PASO #2: Descargar mod_jk 1.2.x (mod_jk-1.2.25-httpd-2.2.4.so) o de la pagina http://apache.freeby.pctools.cl/tomcat/tomcat-connectors/jk/binaries/linux/jk-1.2.25/i386/

Una vez descargado el archivo mod_jk debemos renombrarlo por **mod_jk.so** y colocarlo en la carpeta APACHE_HOME/modules.

PASO #3: Configurar Apache para que use modjk al final del archivo APACHE_HOME/conf/httpd.conf agregar las siguientes lineas:

```
# Include mod_jk configuration file
Include conf/mod-jk.conf
```

PASO #4: Creamos el archivo modjk con la configuración.

Para esto en la carpeta APACHE_HOME/conf, creamos el archivo **mod-jk.conf** y colocamos lo siguiente:

```
# Load mod_jk module
# Specify the filename of the mod_jk lib
LoadModule jk_module modules/mod_jk.so

# Where to find workers.properties
JkWorkersFile conf/workers.properties

# Where to put jk logs
JkLogFile logs/mod_jk.log

# Set the jk log level [debug/error/info]
JkLogLevel info

# Select the log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"

# JkOptions indicates to send SSK KEY SIZE
# Note: Changed from +ForwardURICompat.
# See http://tomcat.apache.org/security-jk.html
JkOptions +ForwardKeySize +ForwardURICompatUnparsed -ForwardDirectories

# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"

# Mount your applications
JkMount /__application__/* loadbalancer

# You can use external file for mount points.
# It will be checked for updates each 60 seconds.
# The format of the file is: /url=worker
# /examples/*=loadbalancer
JkMountFile conf/uriworkermap.properties

# Add shared memory.
# This directive is present with 1.2.10 and
# later versions of mod_jk, and is needed for
# for load balancing to work properly
# Note: Replaced JkShmFile logs/jk.shm due to SELinux issues. Refer to
# https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=225452
JkShmFile run/jk.shm

# Add jkstatus for managing runtime data
<Location /jkstatus/>
JkMount status
Order deny,allow
Deny from all
Allow from 127.0.0.1
</Location>
```

Ahora el mod_jk esta listo para aceptar peticiones del JBOSS, ahora necesitamos configurar los workers.

PASO #5: Configurar workers
En la carpeta APACHE_HOME/conf, crear el archivo **workers.properties** y escribir lo siguiente:

```
# Define list of workers that will be used

# for mapping requests
# The configuration directives are valid
# for the mod_jk version 1.2.18 and later
#
worker.list=loadbalancer,status

# Define Node1
# modify the host as your host IP or DNS name.
worker.node1.port=8009
worker.node1.host=localhost
worker.node1.type=ajp13
worker.node1.lbfactor=1
# worker.node1.connection_pool_size=10 (1)

# Define Node2
# modify the host as your host IP or DNS name.
worker.node2.port=8009
worker.node2.host= localhost
worker.node2.type=ajp13
worker.node2.lbfactor=1
# worker.node1.connection_pool_size=10 (1)
```

```
# Load-balancing behaviour
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=node1,node2

# Status worker for managing load balancer
worker.status.type=status
```

Deben modificar localhost por si IP o DNS eso depende de su configuracion del servidor, pero si el servidor esta en su mismo ekipo dejenlo asi mismo.

PASO #6: Crear los URI

crear un archivo que se llame **uriworkermap.properties** en la carpeta APACHE_HOME/conf

```
# Simple worker configuration file
#

# Mount the Servlet context to the ajp13 worker
/jmx-console=loadbalancer
/jmx-console/*=loadbalancer
/web-console=loadbalancer
/web-console/*=loadbalancer
```

Esto configurara el mod_jk para responder peticiones para /jmx-console y /web-console a Apache.

PASO #7: Reiniciar el Apache. Si es ke ya tenemos el servicio corriendo debemos detenerlo de la siguiente forma $ sudo APACHE_HOME/bin/apachectl stop de lo contrario lo hacemos correr

```
$ sudo APACHE/bin/apachectl start
```

PASO #8: Configure Tomcat en JBOSS

Para completar la configuracion, necesitamos nombrar cada nodo de acuerdo a la configuración especificada en los workers. Para esto editamos el archivo JBOSS_HOME/server/all/deploy/jbossweb-tomcat50.sar/server.xml (reemplazar /all de acuerdo a la configuracion que estamos ocupando)

Buscar la etiqueta <Engine….> y agregar el atributo **jvmRoute**:

```
<Engine name="jboss.web" defaultHost="localhost" jvmRoute="node1"> . </Engine>
```

Este nombre debe coincidir con el declarado en el worker que creamos.

Además asegurarse que el esta linea este sin comentario:

```
<!— A AJP 1.3 Connector on port 8009 —>
<Connector port="8009″ address="${jboss.bind.address}" emptySessionPath="true"
enableLookups="false" redirectPort="8443″ protocol="AJP/1.3″/>
```

PASO #10: Activate the UseJK en el JBoss

Para esto editamos el archivo JBOSS_HOME/server/all/deploy/jbossweb-tomcat50.sar/META-INF/jboss-service.xml.

(reemplazar /all de acuerdo a la configuracion que estamos ocupando)

Encontrar la etiqueta <attribute> con el nombre UseJK, y modificamos su valor a "true":

```
<attribute name="UseJK">true</attribute>
```

La versión del Tomcat puede ser 50 o 55 depende de la versión del JBOSS que estemos ocupando.

PASO #11: Definir variable Global JBOSS_HOME

Agregamos la siguiente linea al archivo /etc/environment.

El directorio /usr/local/jboss-4.0.4.GA/ corresponde al cual nosotros elegimos cuando lo instalamos (descomprimimos)

```
$ sudo gedit /etc/environment
JBOSS_HOME=/usr/local/jboss-4.0.4.GA/
```

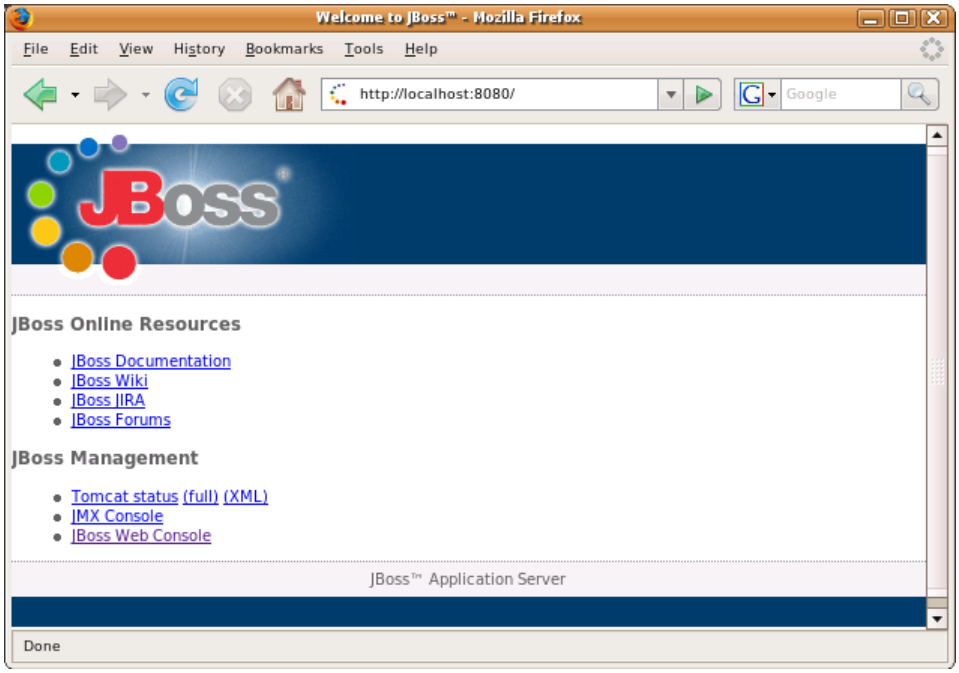
Reniciamos el PC

PASO #12: Iniciamos JBOSS y Apache

o lo hacemos correr de la siguiente forma

```
$ sudo cd /APACHE_HOME/bin/ ./apachectl start
$ cd $JBOSS_HOME/bin
$ sh ./run.sh
```

Si todo quedo bien configurado deberíamos poder entrar por acá http://localhost:8080 http://localhost:8080/web-console/ o http://localhost:8080/ y ver una pantalla como la siguiente.



4. Cuentas de Usuario JBOSS

4.1. Creación de una cuenta

Para comenzar la configuración de las cuentas del servidor Apache+JBOSS deberemos realizar dos operaciones:

- **Modificacion del archivo httpd.conf, para activacion de la carpeta public_html**
- **Crear usuarios nuevos**

Para la modificacion del httpd.con, debemos ingresar al directorio donde instalamos apache que lo llamaremos APACHE_HOME, y modificar el archivo de la siguiente forma:

- **$ cd APACHE_HOME/conf/**
- **$ sudo gedit httpd.conf**

Y buscaremos las siguientes lineas

```
# UserDir: The name of the directory that is appended onto a user's home
# directory if a ~user request is received.
#
UserDir public_html

#
# Control access to UserDir directories.  The following is an example
# for a site where these directories are restricted to read-only.
#
#<Directory /home/*/public_html>
#    AllowOverride FileInfo AuthConfig Limit Indexes
#    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
#    <Limit GET POST OPTIONS PROPFIND>
#        Order allow,deny
#        Allow from all
#    </Limit>
#    <LimitExcept GET POST OPTIONS PROPFIND>
#        Order deny,allow
#        Deny from all
#    </LimitExcept>
#</Directory>
```

Y quitamos las almoadillas de `#<Directory /home/*/public_html>` , por lo que deberia quedar asi:

```
<Directory /home/*/public_html>
    AllowOverride FileInfo AuthConfig Limit Indexes
    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
    <Limit GET POST OPTIONS PROPFIND>
        Order allow,deny
        Allow from all
    </Limit>
    <LimitExcept GET POST OPTIONS PROPFIND>
        Order deny,allow
        Deny from all
    </LimitExcept>
</Directory>
```

Con esto estamos permitiendo que que se puedan revisar los archivos via web de todos los usuarios que crearemos.

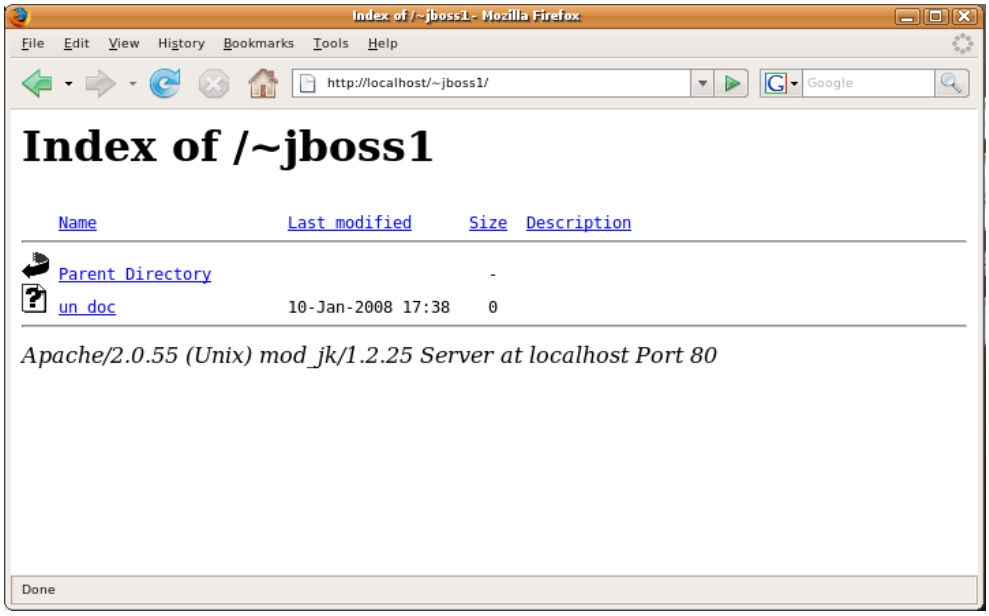
A continuación, via consola creamos una nueva carpeta en nuestro $HOME llamada **public_html** para comprobar que la configuración quedo correcta y reiniciar nuestro servidor apache para esto realizamos los siguientes pasos.

- **cd $HOME**
- **mkdir public_html**
- **cd APACHE_HOME/bin**
- **sudo ./apachectl stop**
- **sudo ./apachectl start**

Abrimos nuestro navegador de internet y probamos con la siguiente ruta (en este caso con el usuario jboss1 que es el usuario con el cual estamos realizando la instalación de nuestro servidor):

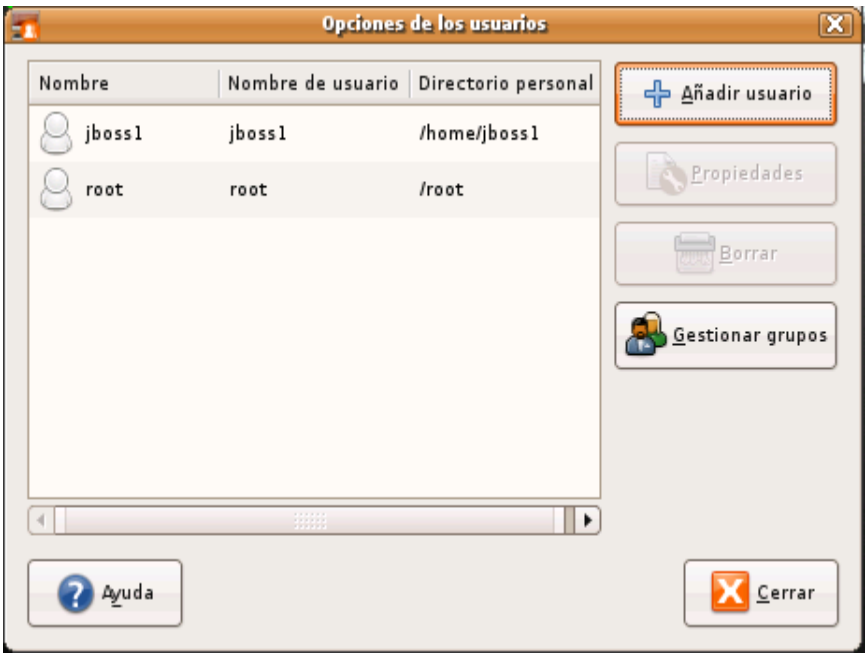
- **http://localhost/~jboss1/**

Si todo esta bien deberiamos poder ver algo como esto:



Ahora, como ya tenemos configurado nuestro apache para que se puedan ver los archivos que estan guardados en nuestra cuenta, debemos crear las cuentas de usuario para nuestro servidor JBOSS, para esto, creamos un usuario de forma normal realizando los siguientes pasos:

- **SISTEMA>ADMINISTRACION>USUARIOS Y GRUPOS**



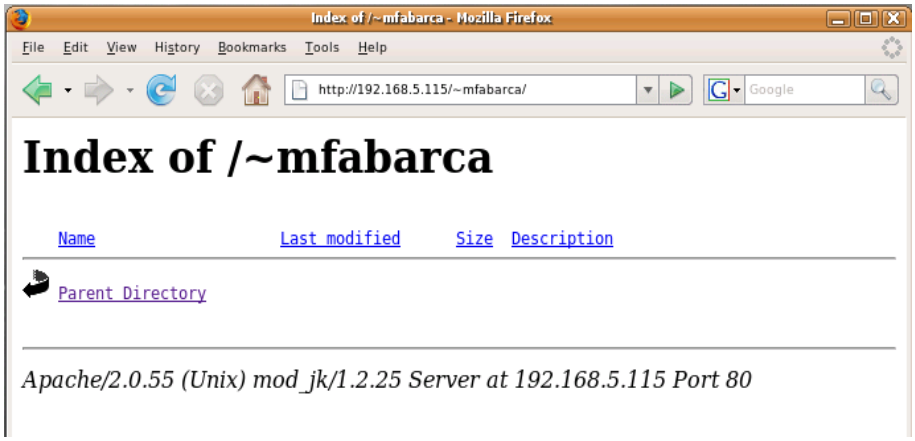
Y le damos click al botón AÑADIR USUARIO, lo que nos permitira ingresar uno nuevo

Ingresamos los datos que nos solicita y su respectiva contraseña, en este caso crearemos el usuario **mfabarca:**

Ahora debemos crear su respectiva carpeta **public_html** de la siguiente forma:

- **$ sudo bash**
- **# su - mfabarca**
- **$ mkdir public_html**
- **$ exit**
- **$ exit**

Si se lo crear de forma correcta deberiamos poder ingresar al contenido de la misma forma que con el usuario **jboss1** pero con el nombre **mfabarca**



4.2. Modificación del archivo jboss-service.xml

El servidor JBOSS por defecto solamente escanea un directorio **(JBOSS_HOME/server/[conf]/deploy/)** donde son almacenadas todas las aplicaciones J2EE, para esto debemos indicarle que no solamente debe escanear esa sino que todas las carpetas de usuarios que utilizan el servidor, para esto primeramente cada usuario deberá tener en su **public_html** un directorio llamado **deploy**, en el cual almacenaremos nuestras aplicaciones EAR, para que el JBOSS las compile, y podamos verlas a travez de la web.

Para esto, debemos realizar las siguientes instrucciones via consola para crear la carpeta deploy y modificar el archivo jboss-service.xml (utilizaremos la configuracion default):

- **$ sudo bash**
- **# su - mfabarca**
- **$ cd public_html**
- **$ mkdir deploy**
- **$ exit**
- **# exit**
- **$ cd $JBOSS_HOME/server/default/conf/**
- **$ sudo gedit jboss-service.xml**

Nos vamos al final del archivo y modificamos el atributo URLs, veremos que solo se encuentra la carpeta **deploy**, a el deberemos ingresar todas las rutas a la carpeta deploy que se encuentra en el public_html de cada uno de los usuarios, de la siguiente forma, en este caso agregaremos al usuario **mfabarca**:

```
            deploys myapp.ear from a remote location
        http://www.test.com/netboot/apps/
            scans the specified WebDAV location
 -->
<attribute name="URLs">
   deploy/,
   file:/home/mfabarca/public_html/deploy/
</attribute>
```

Cada url debe ir separada por una coma (,) tal y como aparece ahy, por ejemplo si tenemos 3 usuarios deberian ir asi:

```
<attribute name=”URLs”>
       deploy/,
       file:/home/user1/public_html/deploy/,
       file:/home/user2/public_html/deploy/,
       file:/home/user3/public_html/deploy/
</attribute>
```

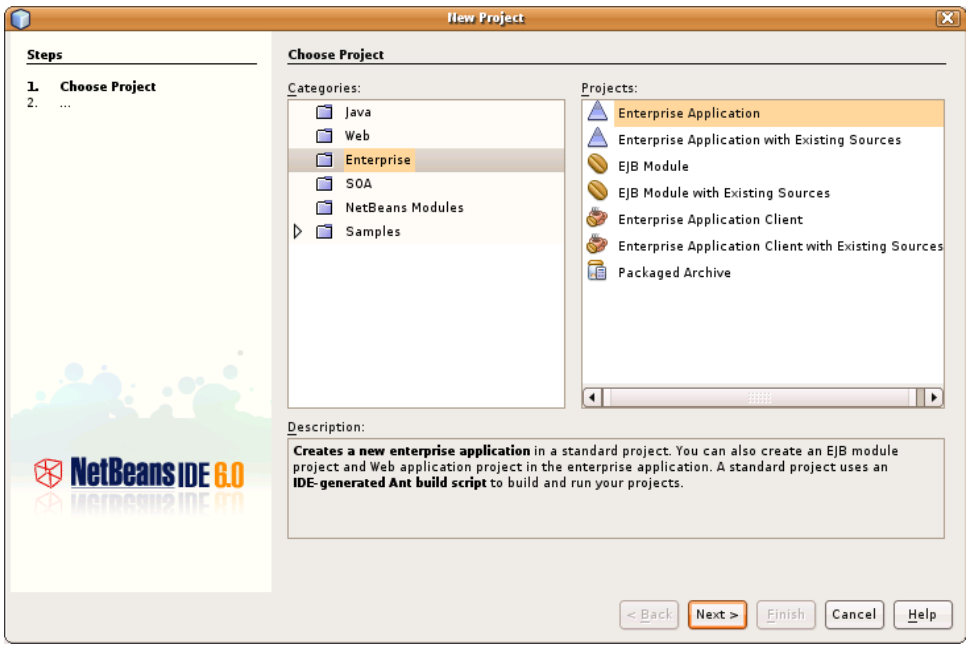
De esta forma le indicamos que todas esas rutas deben estar siempre en actualización.

NOTA: La no existencia de estas carpetas producira que no se puedan ni ver los archivos almacenados ni tampoco ver las aplicaciones EAR que necesiten sean vista via web.

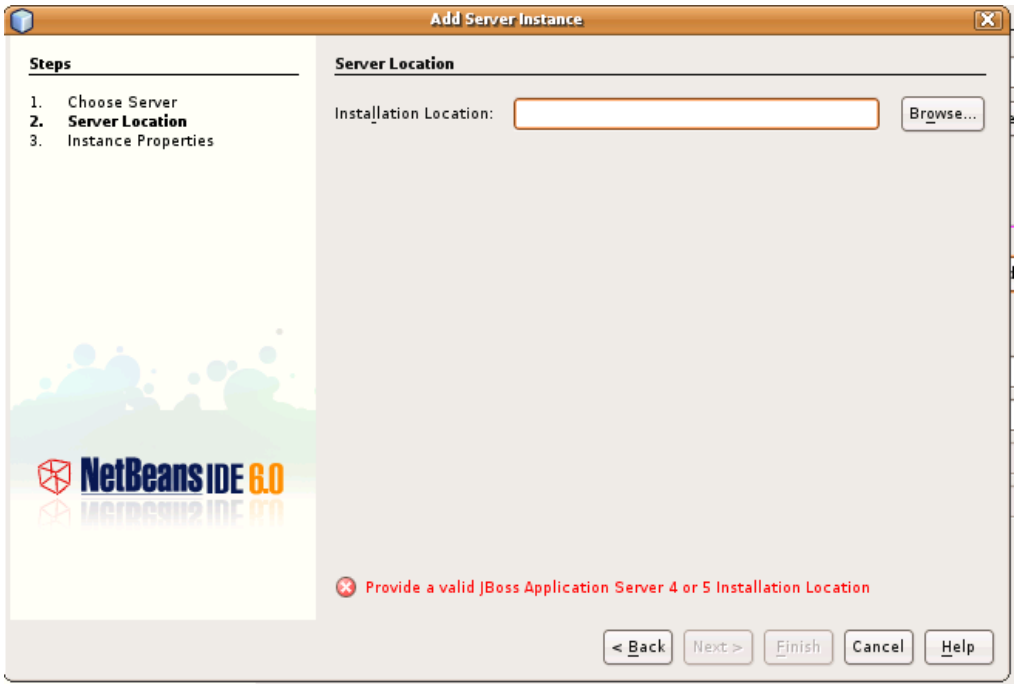
4.3. Hola Mundo!, aplicación de ejemplo

Para probar que nuestras cuentas funcionan correctamente, debemos crear una pequeña aplicación EAR de ejemplo que llamaremos HolaMundo, para esto abrimos nuestro netbeans que fue instalado cuando instalamos los paquetes bljava necesarios para el desarrollo con J2EE, asi abrimos el menu FILE y elegimos New Project y seleccionamos

- **Enterprise> Enterprise Aplication**

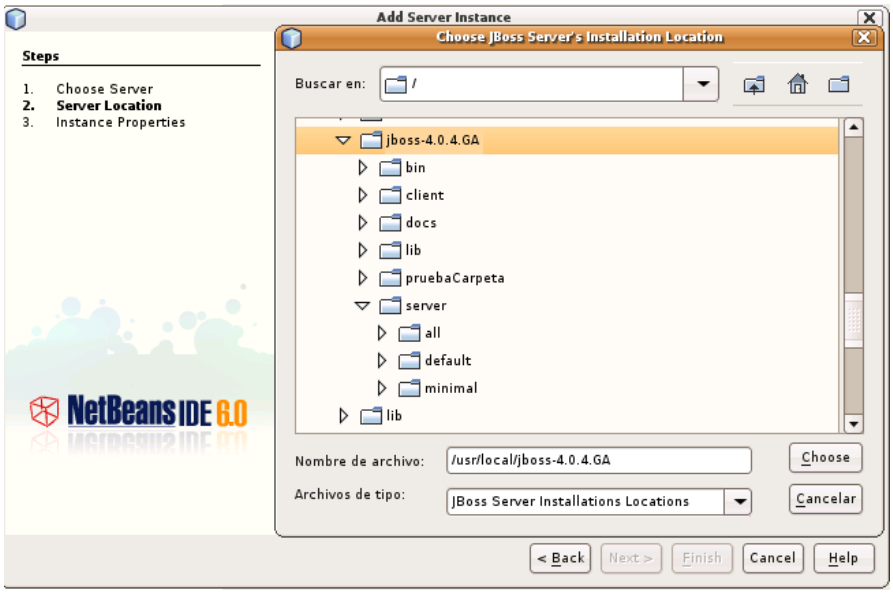


En la siguiente pantalla seleccionamos nuestro servidor JBOSS, sino se encuentra, deberemos añadirlo presionando el botón **ADD:**

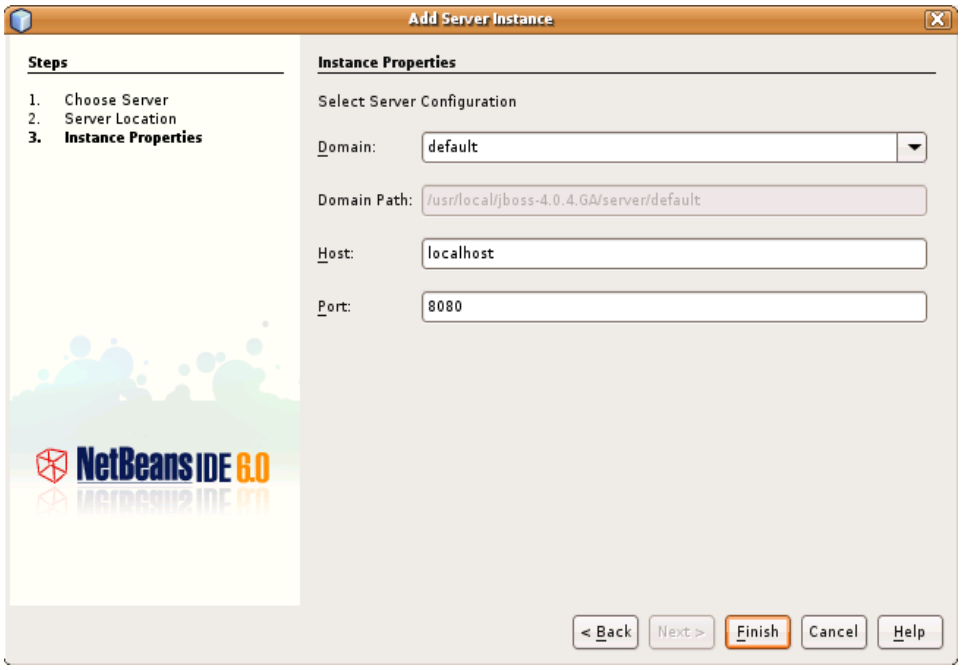


Presionamos **BROWSE**

Y buscamos la carpeta donde instalamos el servidor llamada **jboss-4.0.4.GA**, que en este caso se encuentra en la carpeta **/usr/local/** presionando el botón BROWSE



Seleccionamos y continuamos la creación de nuestra aplicación de ejemplo:

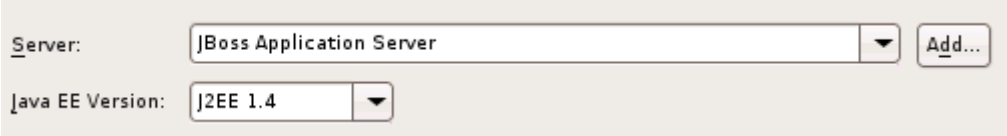


Debido a que JBOSS coloca todas las aplicaciones en un solo lugar, es necesario separar las aplicaciones para cada usuario, para esto, cada vez que creamos un proyecto es necesario indicarle el identificador de cada uno, que en este caso sería el nombre del usuario, para evitar perdidas en las aplicaciones o que se confundan con otras.

Esto se debe realizar de la siguiente forma:

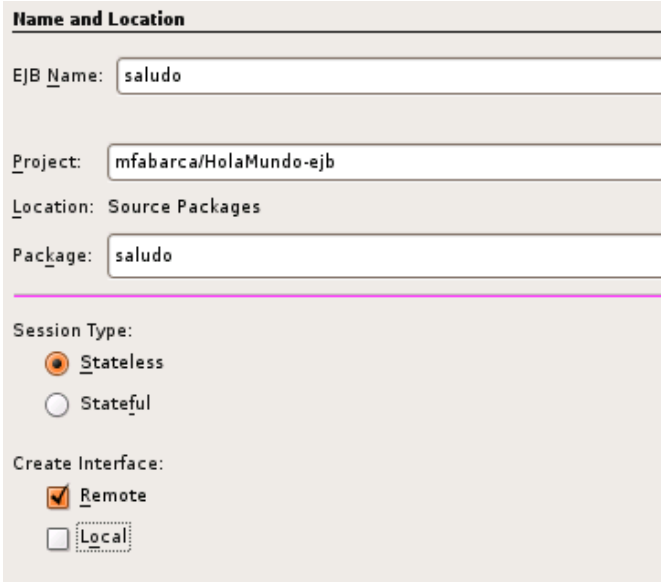
- **Colocamos el nombre de la aplicación normalmente, en este caso el nuestro sera el conocido HOLA MUNDO!**

- **A continuación seleccionaremos nuestro servidor de aplicaciones JBOSS**



- **Y procedemos a indicar que nuestro modulo EJB y aplicación WAR deben quedar en el mismo directorio correspondiente al nombre de usuario, digitandolo de la siguiente forma:**

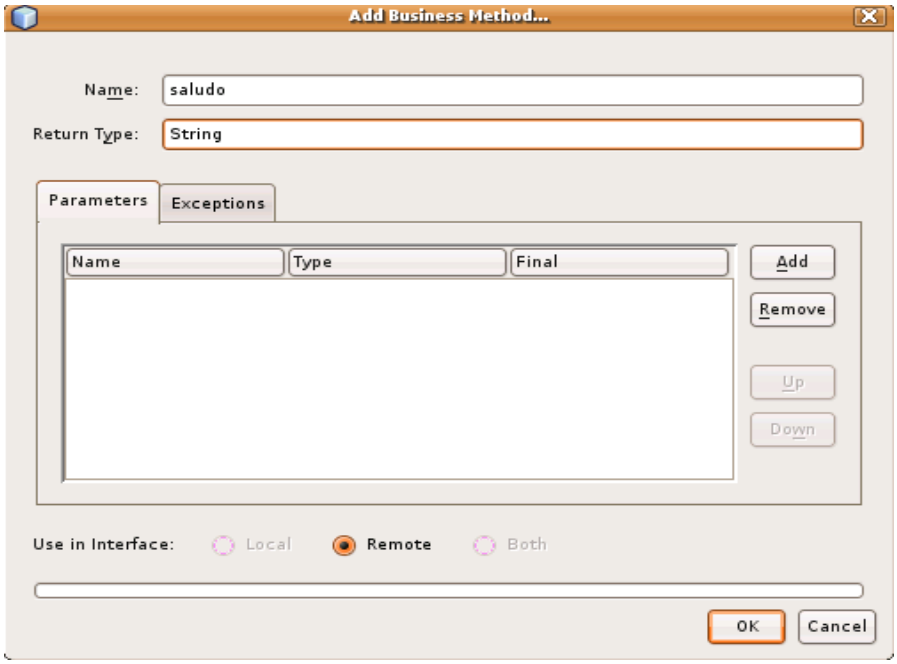
Asi le estamos indicando a nuestro servidor que la aplicación debe ser almacenada y ejecutada dentro de un directorio correspondiente al nombre del usuario.



Ahora es bueno indicar que nuestras aplicaciones EAR estan compuestas de dos partes que son el EJB y el WAR por lo tanto ambos elementos son almacenados en un mismo archivo y compilados juntos, por lo que debemos tener cuidado también con los nombres de las aplicaciones EAR que se van creando.

Ya creado los 2 modulos de la aplicacón EAR, procedemos a crear un beans de sesion en el HolaMundo-ejb, para ello presionamos con el boton derecho sobre **SOURCE PACKAGES** y seleccionamos **NEW SESSION** BEAN y lo creamos como se indica en la figura.

A continuación creamos un nuevo metodo de negocio presionando con el botón derecho sobre el editor de codigo y seleccionamos **EJB METHODS> ADD BUSINESS METHOD** y creamos un metodo básico que retorne un String, para eso lo hacemos de la siguiente forma:

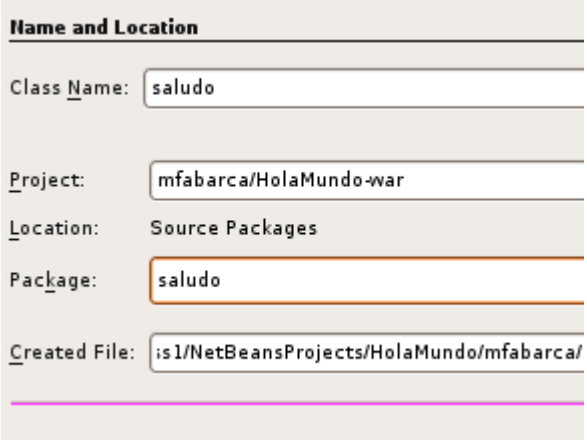


Cuando ya este creado, veremos en el editor el método, lo modificamos y lo dejamos de la siguiente forma:

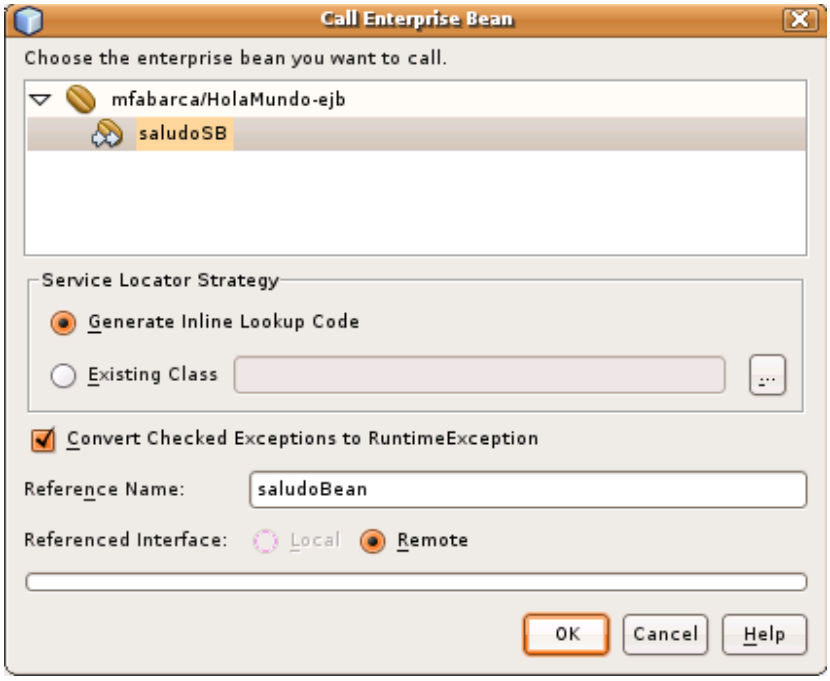
```
public String saludo() {
    return "Hola Mundo!!";
}
```

Ahora procedemos a compilar el modulo EJB y a llamar a nuestro beans de sesión desde el modulo WAR.

Para eso creamos un nuevo servlet, presionando nuevamente sobre **SOURCE PACKAGES** esta vez sobre el modulo WAR y seleccionando **NEW SERVLET**, le damos los datos que tenemos en la figura, y lo creamos.



Una vez creado presionamos con el botón derecho sobre el editor y elegimos **ENTERPRISE RESOURCES> CALL ENTERPRISE BEAN** y veremos nuestro EJB:



Lo seleccionamos y presionamos OK, luego modificamos el metodo processRequest y lo dejamos de la siguiente forma, para que pueda ser ejecutado desde un JSP y mostrar el contenido del metodo creado en el EJB.

```
protected void processRequest(HttpServletRequest request, HttpServletRespons
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    saludoRemote beanSesion = lookupsaludoBean();
    try {
        /* TODO output your page here*/
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet saludo</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>"+beanSesion.saludo()+"</h1>");
        out.println("</body>");
        out.println("</html>");

    } finally {
        out.close();
    }
}
```

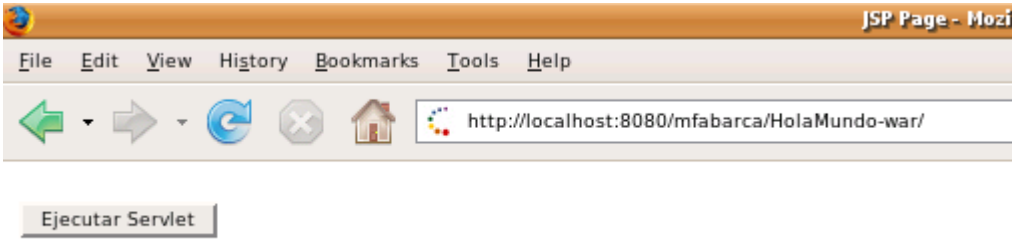
Cuando este modificado de forma correcta, procedemos a escribir nuestro index.jsp que será ejecutado cuando iniciemos nuestra aplicación EAR.

Para ello abrimos el index.jsp que se encuentra en **WEB PAGES** y digitamos el siguiente codigo HTML, que nos permitira ejecutar el servlet que llama a nuestro EJB de saludo

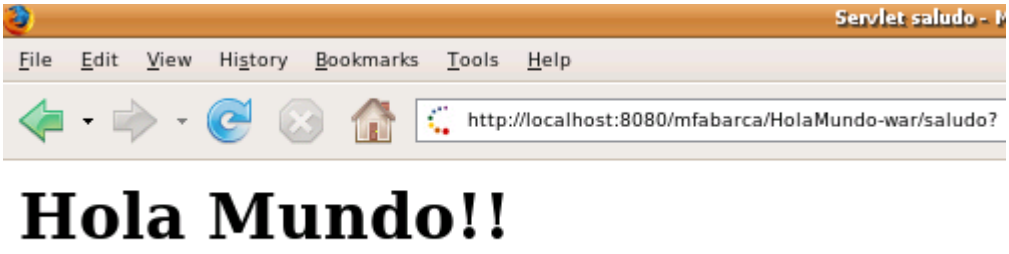
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <form action="saludo">
        <h2><input type="submit" value="Ejecutar Servlet"></h2>
        </form>

    </body>
</html>
```

Si todo ha quedado correctamente procedemos a compilar nuestra aplicación para verla en el navegador de prueba, para ello solamente presionamos sobre nuestra aplicación Enterprise representada por esta figura:

Presionamos el triangulo con el botón derecho y elegimos **Undeploy and Deploy** y luego **Run**, cuando este compilada completamente se abrira nuestro navegador y veremos la aplicación.

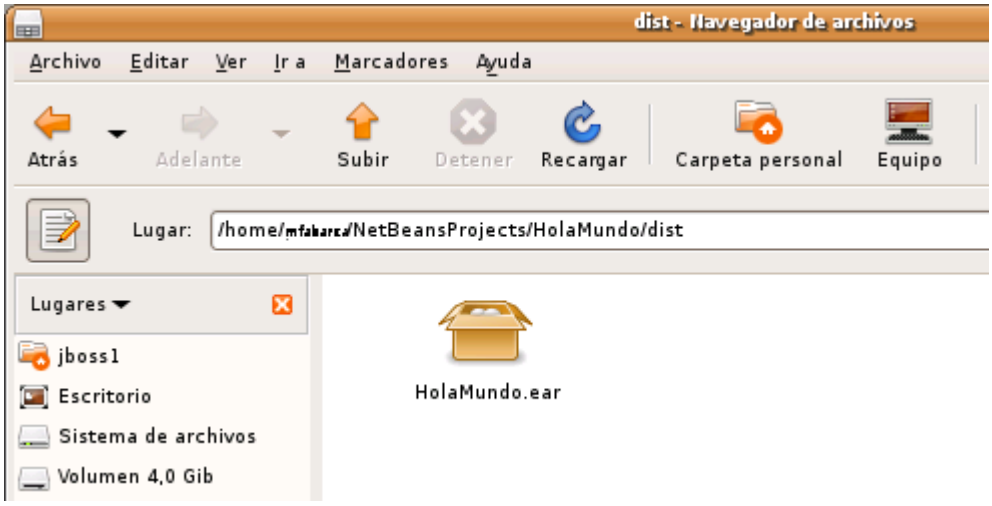


Presionamos sobre el botón y se deberia ejecutar nuestro Servlet:



Ahora, para que nuestra aplicación pueda ser visible de forma local o externa debemos, copiar nuestra aplicación EAR a nuestro correspondiente directorio **$HOME/public_html/deploy** para que pueda ser compilada desde nuestro servidor JBOSS, para ello entramos al directorio donde creamos nuestra aplicación que por defecto es:

- **$HOME/NetBeansProjects/HolaMundo**
- **cd dist**



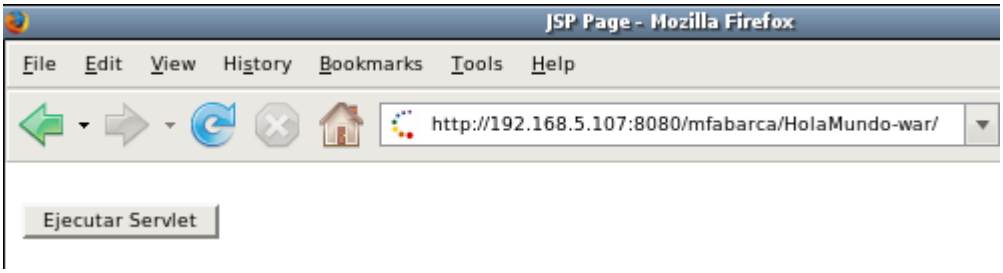
Posteriormente procedemos a copiar la aplicación a nuestro directorio deploy, para ello entramos a:

- **$HOME/public_html/deploy**

Cuando la aplicación es copiada o movida a esta carpeta automaticamente es compilada en el servidor de aplicaciones JBOSS, y puede ser vista ingresando a nuestro navegador escribiendo el numero de la IP o el nombre de la maquina del servidor más nuestro nombre de usuario y nombre de la aplicación WAR.

Asi ingresamos a un navegador externo e ingresamos lo siguiente (192.168.5.107 ip de prueba del servidor)

- **http://192.168.5.107:8080/mfabarca/HolaMundo-war/**



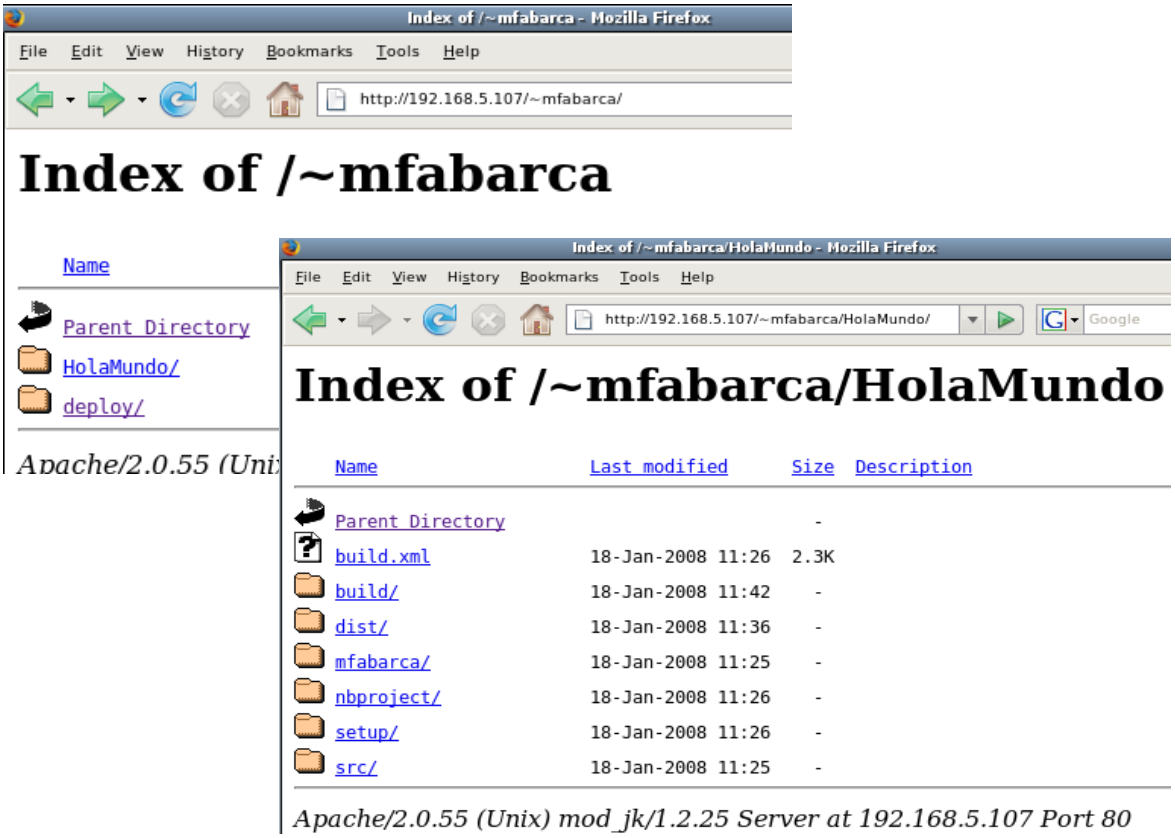
Indicamos el puerto 8080 para decir que queremos ejecutar una aplicación que se encuentra almacenada en el servidor JBOSS.

Ahora si además deseamos ver todos los archivos de nuestra aplicación, cuando creamos la aplicación debemos hacerlo en el correspondiente directorio publico ($HOME/public_html) o debemos moverlo desde el directorio creado por defecto a nuestro $HOME/public_html de la siguiente forma:

- **$ cd $HOME/NetBeansProjects**
- **$ mv HolaMundo $HOME/public_html**

Finalmente deberiamos poder ver los archivos asi, entrando en nuestro navegador:

- **http://192.168.5.107/~mfabarca**



5. PostgreSQL

5.1. Librerias necesarias

Para esta instalacion debemos descargar primeramente el tar.gz v8.0.1 de la pagina oficial de postgres www.postgresql.org, luego deberemos instalar las siguientes librerias, que deben ser descargadas directamente de la pagina de ubuntu, o buscarlas via google.

- **zlib-1.2.3**
- **libncurses5-dev**
- **libreadline5-dbg**
- **libreadline5-dev**
- **tcl8.4**
- **tclreadline**
- **readline-5.2**

La no instalación de estas librerias producira errores en la instalación.

5.2. Instalación

Una vez descargadas e instaladas las librerias necesarias procederemos a instalar postgres, para tener un mayore detalle de la instalación se puede revisar el manual de instalacion en el archivo INSTALL en la carpeta descomprimida con el codigo fuente, sino procedemos a entrar al correspondiente directorio via consola y ejecutar los siguientes comandos.

- **$ sudo -s**
- **# ./configure**
- **# make**
- **# make install**

Si la instalacion se realiza correctamente podemos pasar al siguiente paso, de lo contrario revisar que todos los paquetes nombrados anteriormente hallan sido instalados.

5.3. Configuración del usuario postgres y base de datos TEST

Para la configuración del usuario primeramente debemos comprobar que ha sido creado utilizando el siguiente comando.

- **$ sudo adduser postgres**

Si el resultado de la ejecución es que el usuario ya existe, debemos realizar las siguientes configuraciones para la correcta ejecución del motor de base de datos.

- **$ sudo mkdir /usr/local/pgsql/data**
- **$ sudo chown postgres /usr/local/pgsql/data**
- **$ sudo bash**
- **# su - postgres**
- **$ /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data**
- **$ /usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data >logfile 2>&1 &**

Una vez realizado estos pasos debemos crear la BD de prueba que llamaremos TEST y modificaremos la contraseña del usuario postgres debido a que en nuestra instalación se genero una aleatoriamente y deberemos cambiarla por una conocida.

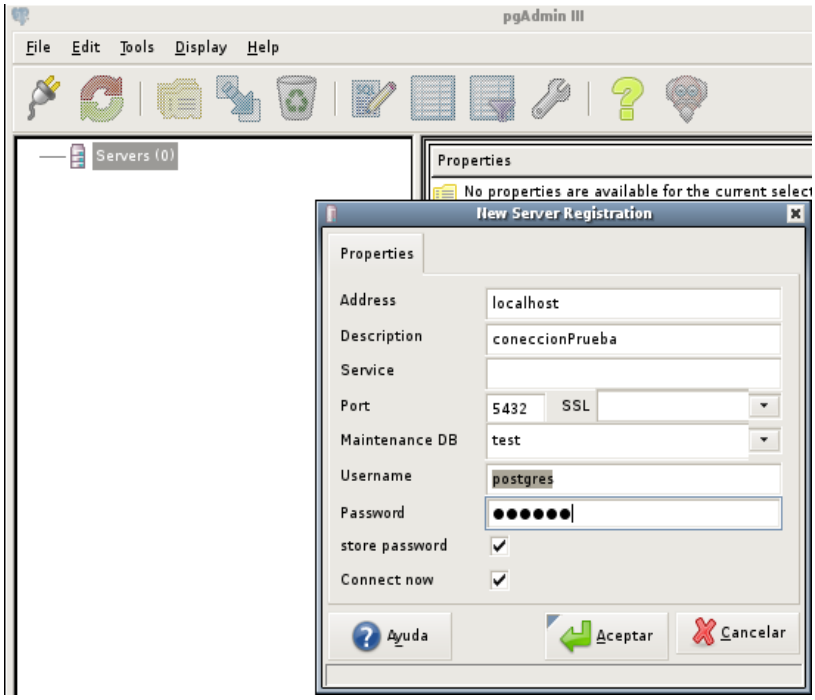
- **$ /usr/local/pgsql/bin/createdb test**
- **$ /usr/local/pgsql/bin/psql test**
- **$ alter user postgres with password [nueva contrasena]**
- **\q**

Con estos pasos realizados, procedemos a realizar nuestra coneccion de prueba, para esto con Synaptic instalamos el pg-admin que nos permitira realizar las conecciones a nuestro motor de base de datos.

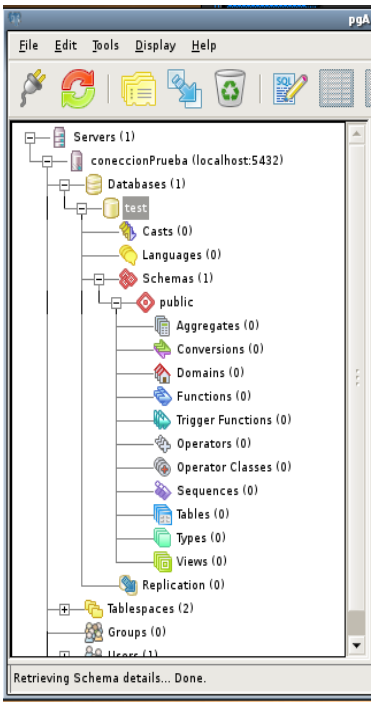
5.4. Conexión de Prueba

Para nuestra conexión de prueba ya debemos tener instalado el pg-admin para ello lo abrimos desde el menu Aplicaciones>Herramientas del Sistema>pgAdmin III, y creamos una nueva conexión con los siguientes datos:

- **Address : localhost**
- **Description : conexionPrueba**
- **Port : 5432**
- **maintance BD : test**
- **Username : postgres**
- **Password : ******



Si la conexión fue realizada exitosamente veremos la conexión con nuestra BD en la barra lateral izquierda



5.5. Configuracion del servidor de Base de Datos

Para poder dejar nuestro motor de base de datos como servidor, y que permita conecciones locales y externas, lo primero es encontrar la ubucacion de los siguientes archivos para poder modificarlos:

- **postgresql.conf**
- **pg_hba.conf**

Si no sabes como ubicarlos solamente digita el siguiente comando:

- **$ sudo find / -name [n_archivo]**

Una vez encontrados procederemos a modificar primeramente el archivo postgresql.conf, para eso entramos a su correspondiente directorio y ejecutamos lo siguiente:

- **$ sudo gedit postgresql.conf**

Y buscamos las siguientes lineas:

- **#listen_addressess = 'localhost'**
- **#port=5432**

Que deben ser modificadas por estas:

- **listen_addressess = '*'**
- **port=5432**

Ahora modificamos el pg_hba.conf, en el buscamos la siguiente linea:

- **#IPv4 local connections:**
- **host all all 127.0.0.1/32 trust**

Y la modificamos por estas:

- **#IPv4 local connections:**
- **host all all 0.0.0.0/0 trust**

Una vez realizados estos cambios reiniciamos el pc y hacemos correr nuestro servidor de BD con los siguientes comandos:

- **$ sudo bash**
- **$ su - postgres**
- **$ cd /usr/local/pgsql/bin**
- **$ postmaster -D /usr/local/pgsql/data**

Para probar las conecciones externas y locales, solamente colocamos en el pg-admin de una maquina local que tenga instalado el pgadmin la direccion IP de nuestro servidor, si no conoces la IP del servidor de BD digita lo siguiente en un terminarl:

- **ifconfig**

Con esto puedes saber la IP del pc en el que esta instalado tu servidor de base de datos.

6. Netbeans 6 + UML

En esta parte básicamente nos centraremos en dos partes importantes que son

- Crear documentacion
- Crear Diagramas de Clases con ingenieria reversa
- Crear Diagramas de Clases independientes
- Crear codigo java a partir de Diagramas de Clases en proyectos existentes

6.1 Creacion de Documentacion

Para la creacion de estos, solamente debemos comentar cada uno de los metodos de nuestras clases con las siguientes etiquetas:

Etiqueta	**Descripcion**
@author *nombre*	Sirve para declarar quien es el autor de la clase.
@version *numero_version*	Sirve para indicar la version de nuestra clase.
@param *nomParametro DescripcionParametro*	Agrega el nombre y la descripcion de uno de los parametros que ingresa a la funcion o metodo.
@return *Descripcion*	Descripcion del valor de salida de la funcion o metodo.
@throws *nombreExc Descripcion* **@exception** *nombre Exc Descripcion*	Agrega el nombre y descripcion de una excepcion que puede ser lanzada en algun momento.

Entonces por ejemplo supongamos que tenemos un metodo con algunos parametros de entrada, deberiamos escribir de la siguiente manera:

```
/**
 * descripcion del metodo
 * @param parametro1 descripcion del parametro1 del metodo1
 * @param parametro2 descripcion del parametro2 del metodo1
 * @param parametro2 descripcion del parametro3 del metodo1
 */
public void Metodo1(int parametro1, String parametro2, int parametro3) {
}
```

Una vez que ya tenemos comentados nuestros metodos procedemos a generar la documentacion de nuestra clase o programa, para esto nos vamos al menu **BUILD> Generate Javadoc for...**

Cuando la documentacion sea generada, podremos verla ingresando al directorio donde seleccionamos, abrimos el archivo index.html y veremos nuestra clase con su documentacion respectiva, para el ejemplo anterior veriamos en el resumen de metodos de la clase:

En un principio veremos todas las clases de nuestro paquete de clases, para ello seleccionamos la que deseamos y veremos su detalle:

Package BeansDePrueba

Interface Summary	
BeansDePruebaRemote	
BeansDePruebaRemoteHome	

Class Summary	
BeansDePruebaBean	

Seleccionamos la clase principal en este caso **BeansDePruebaBean**

Method Summary

void	ejbActivate()
void	ejbCreate()
void	ejbPassivate()
void	ejbRemove()
void	Metodo1(int parametro1, java.lang.String parametro2, int parametro3) descripcion del metodo
java.lang.String	Metodo2(java.lang.String parametro1, int parametro2)
void	setSessionContext(javax.ejb.SessionContext aContext)

Y en la descripcion detallada de cada uno de ellos:

Metodo1

```
public void Metodo1(int parametro1,
                    java.lang.String parametro2,
                    int parametro3)
```

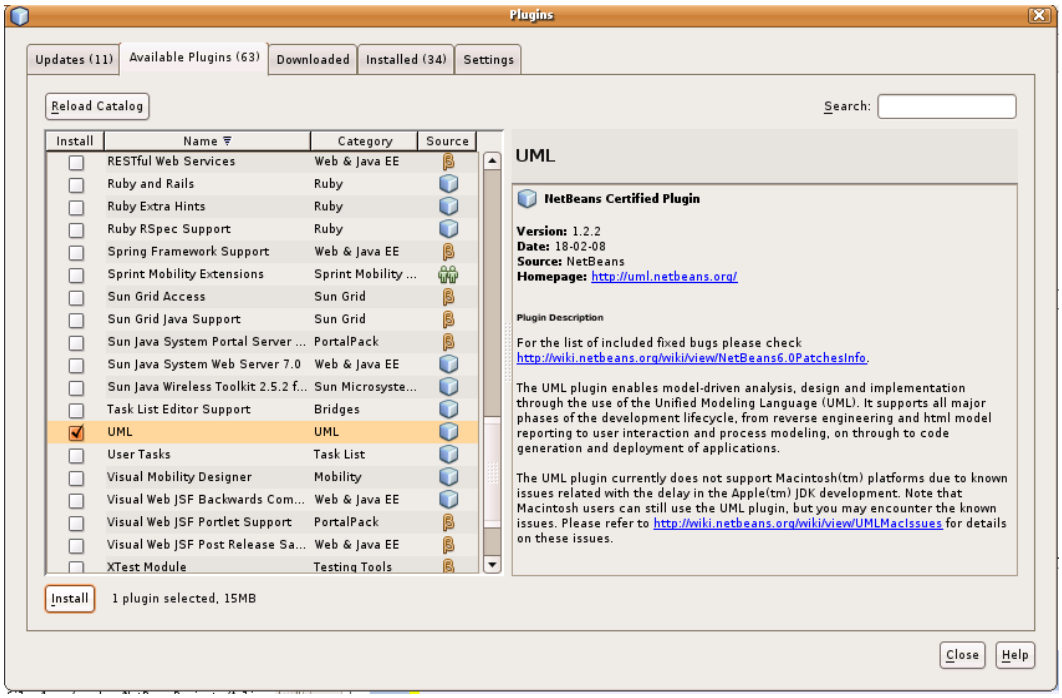
descripcion del metodo

Parameters:

parametro1 - descripcion del parametro1 del metodo1
parametro2 - descripcion del parametro2 del metodo1
parametro2 - descripcion del parametro3 del metodo1

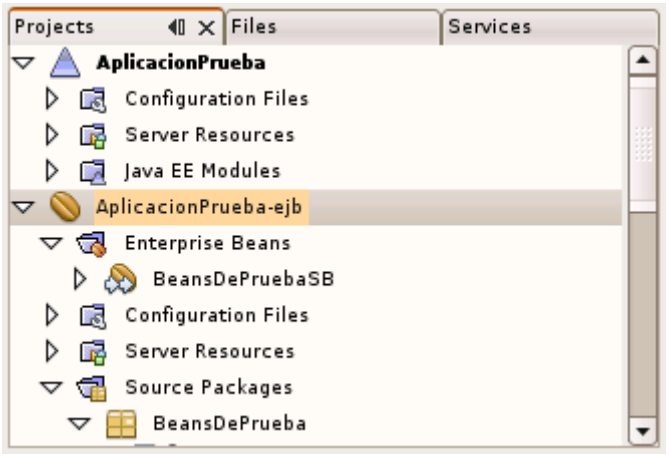
6.2 Creacion de Diagramas de Clase con Ingenieria Reversa

Para poder generarlos, es necesario instalar un paquete adicional al netbeans para aquello nos vamos al menu **TOOL>PLUGIN** y pinchamos la ficha **AVAILABLES PLUGINS**, buscamos **UML** seleccionamos e instalamos

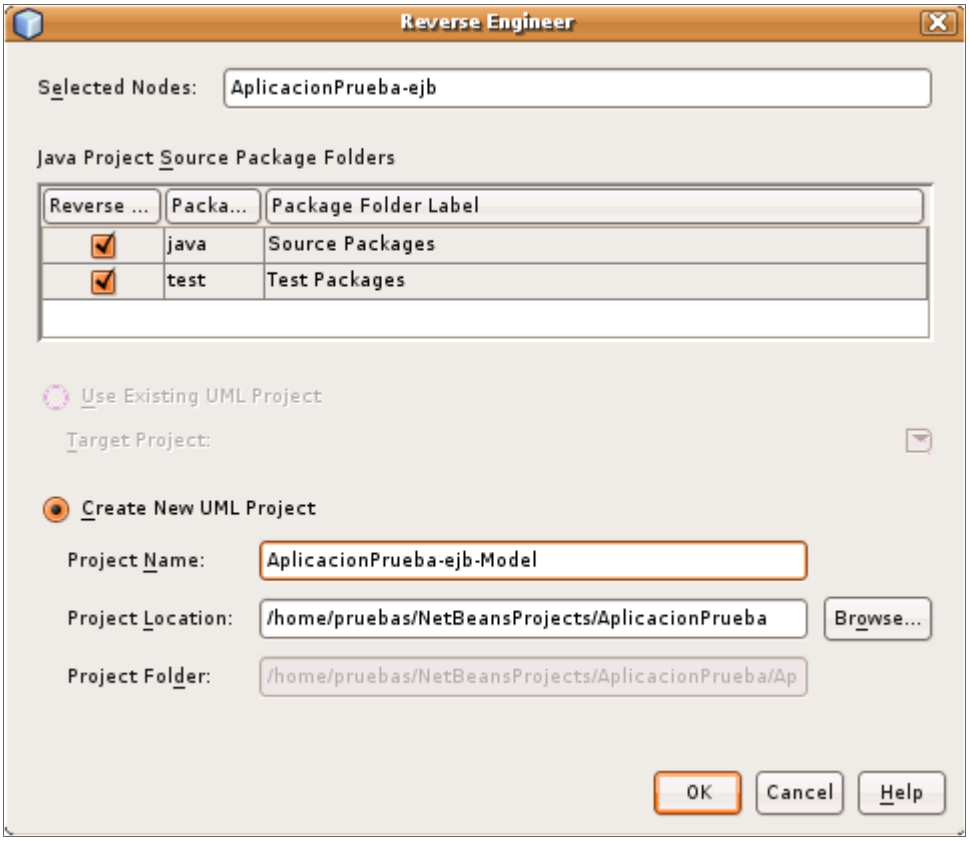


Una vez seleccionado el modulo UML, presionamos el boton **INSTALL** y listo.

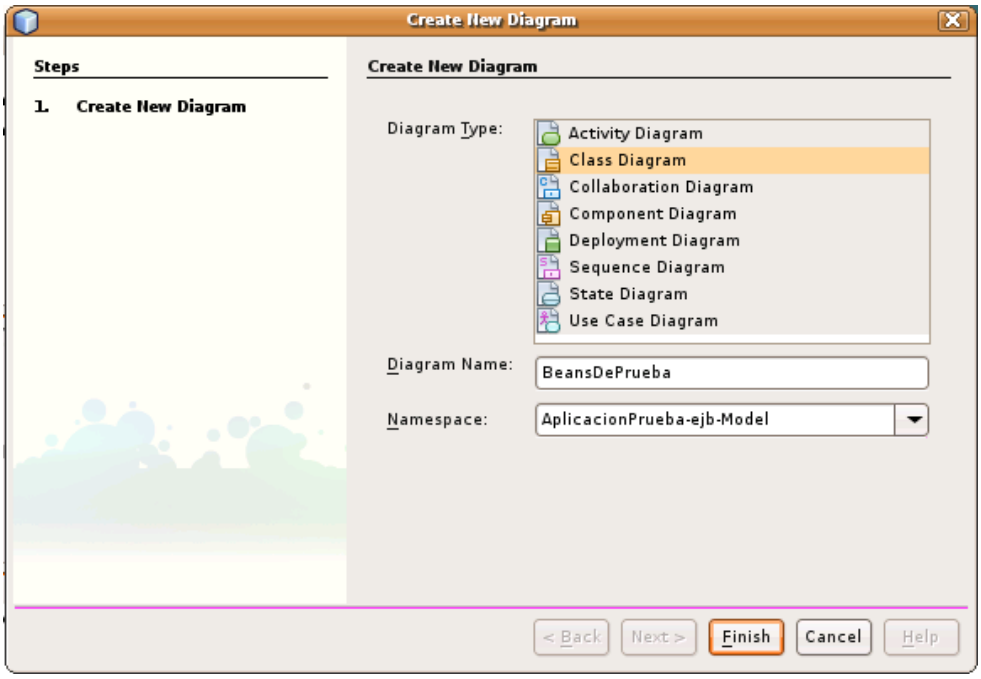
Como ya esta instalado, comenzamos a probar la creacion de los diagramas, para ello presionamos con el boton derecho en el icono principal del EJB, que en esta ocacion se llama **AplicacionPrueba-ejb** y seleccionamos la opcion **REVERSE ENGINEER**



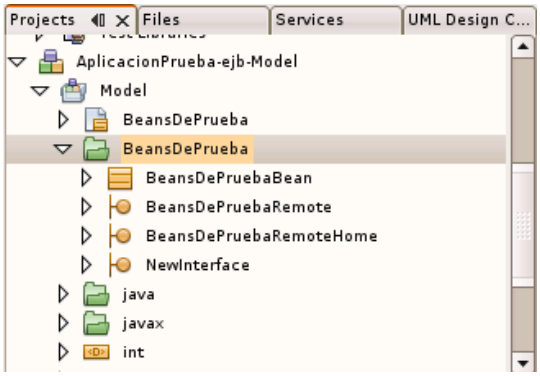
Luego aparecera una ventana en donde debemos seleccionar los elementos para los que queremos hacer ingenieria reversa y le colocamos un nombre al modelo en la opcion **CREATE NEW UML PROJECT**

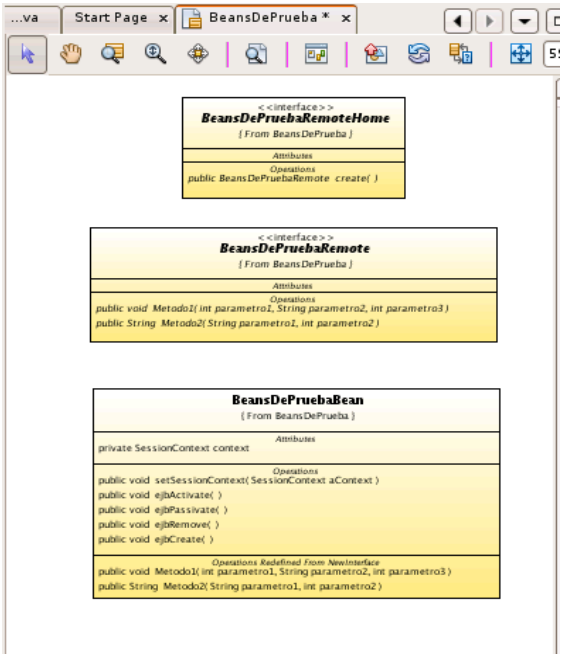


Cuando le demos el nombre, deberemos seleccionar el tipo de modelo que deseamos desarrollar, Colocamos el nombre y presionamos **FINISH**



Cuando este creado el modulo, veremos en la barra lateral izquierda que se han creado algunos elementos pertenecientes a la clase Prueba

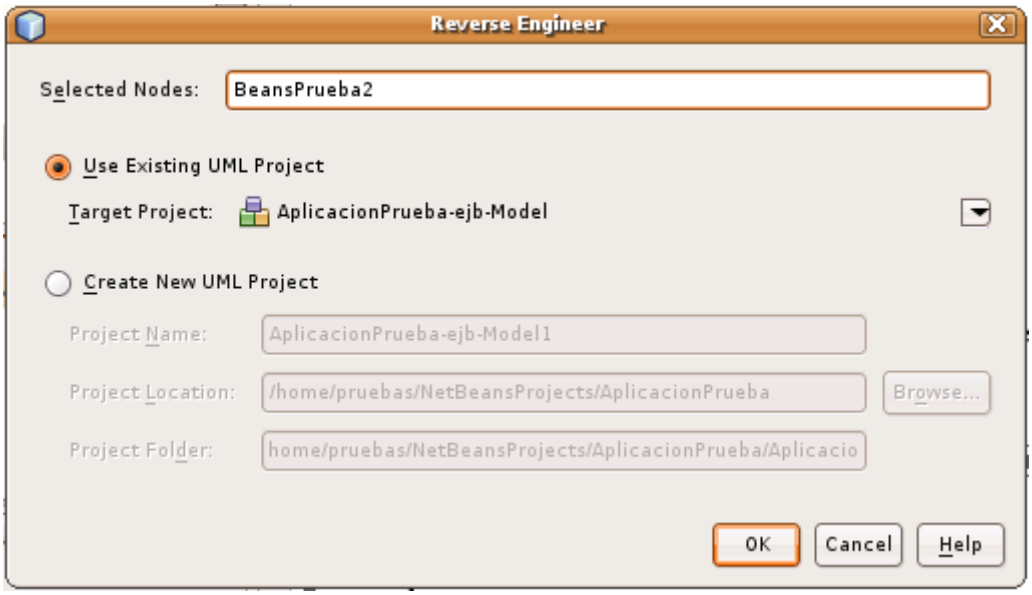




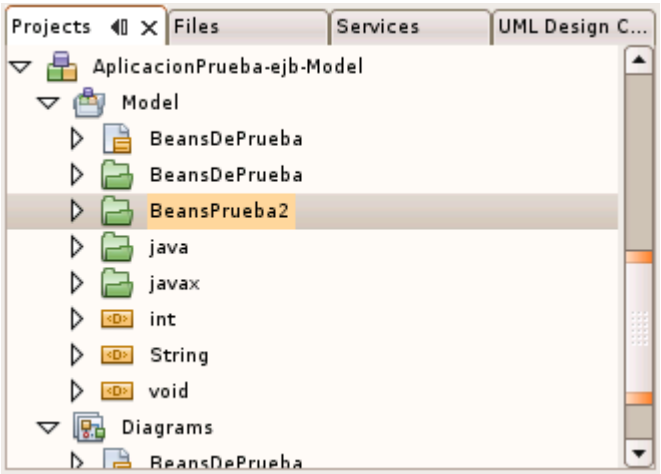
Y en la parte central el diagrama de clase de nuestra respectiva clase o en este caso un beans de sesion.

Ahora, ¿que sucede si vamos agregando mas clases y deseamos agregarlas a nuestro modulo de diagramas?

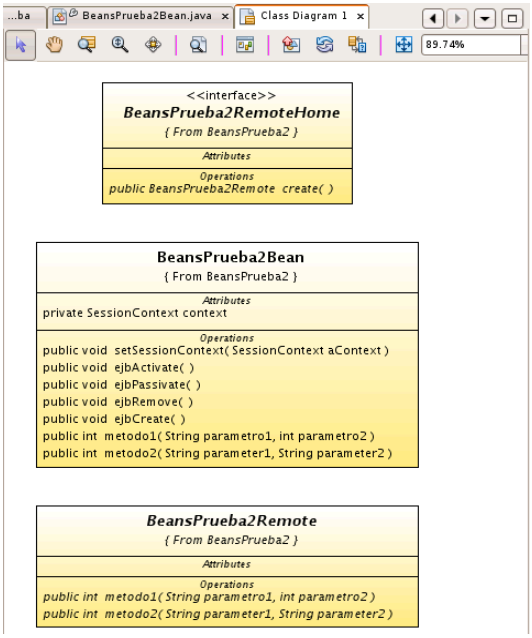
Realizamos los mismos paso, creamos nuestro beans en este caso, posterior mente presionamos con el boton derecho sobre el y seleccionamos **REVERSE ENGINEER**, pero ahora en vez de crear un nuevo modelo solamente agregamos nuestro nuevo beans a la clase seleccionando la opcion **USE EXISTING UML PROJECT**



Cuando ya este agregado presionamos con el boton derecho sobre la carpeta BeansPrueba2 y seleccionamos la opcion **CREATE DIAGRAM FROM SELECTED ELEMENTS** y este se creara en la perte central del area de trabajo de la misma forma que el primer diagrama



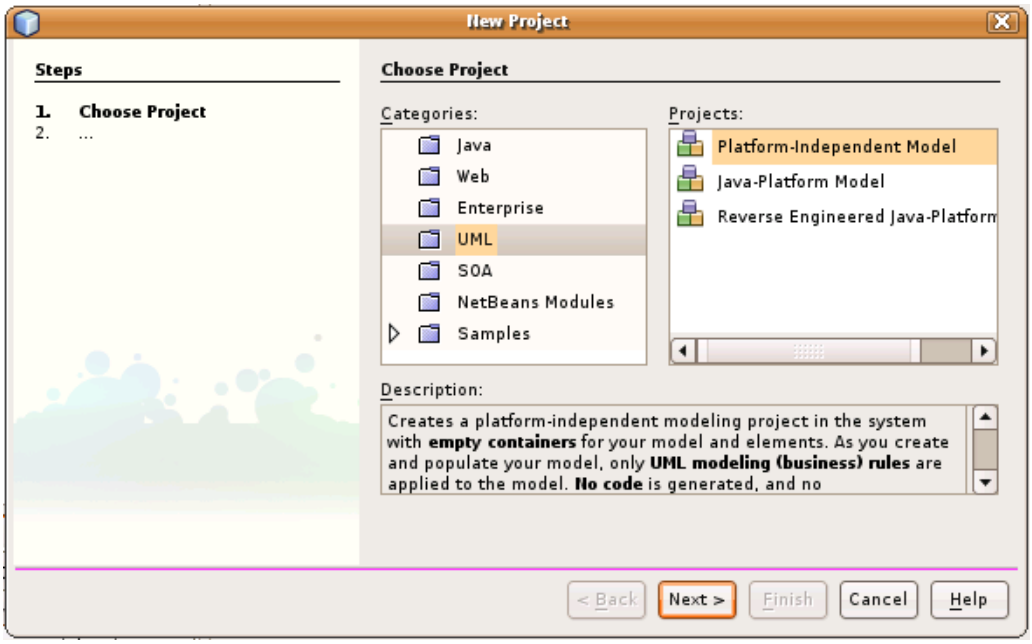
El diagrama, en este caso un beans, aparecera con sus respectivas clases.



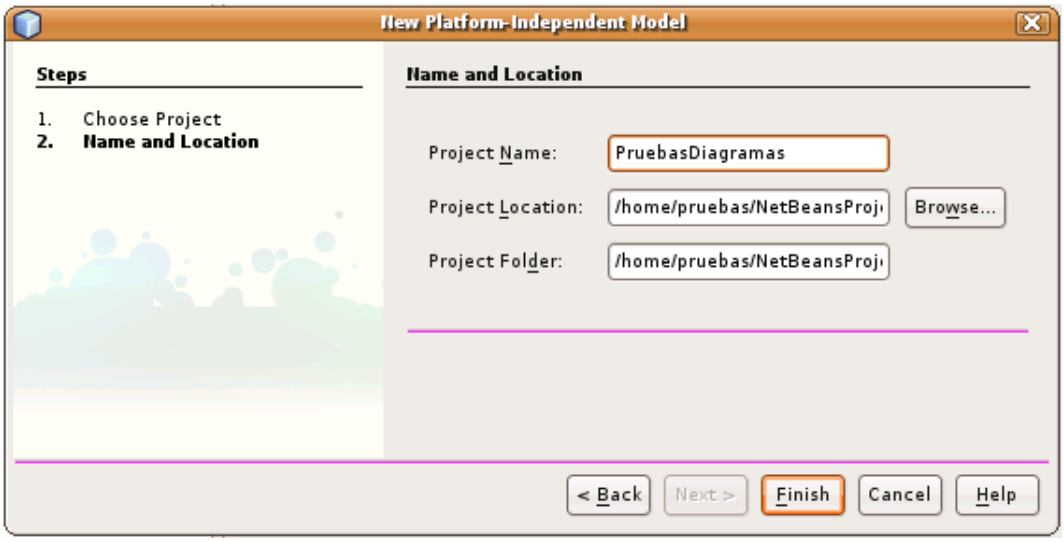
6.3 Creacion independiente de Diagramas de Clases

Para esta parte necesitaremos crear un nuevo proyecto UML como JEE, para comenzar creamos el modulo UML, para ello presionamos el boton de crear proyecto en el menu de barra o nos vamos al menu **FILE** y selecciionamos **NEW PROJECT**

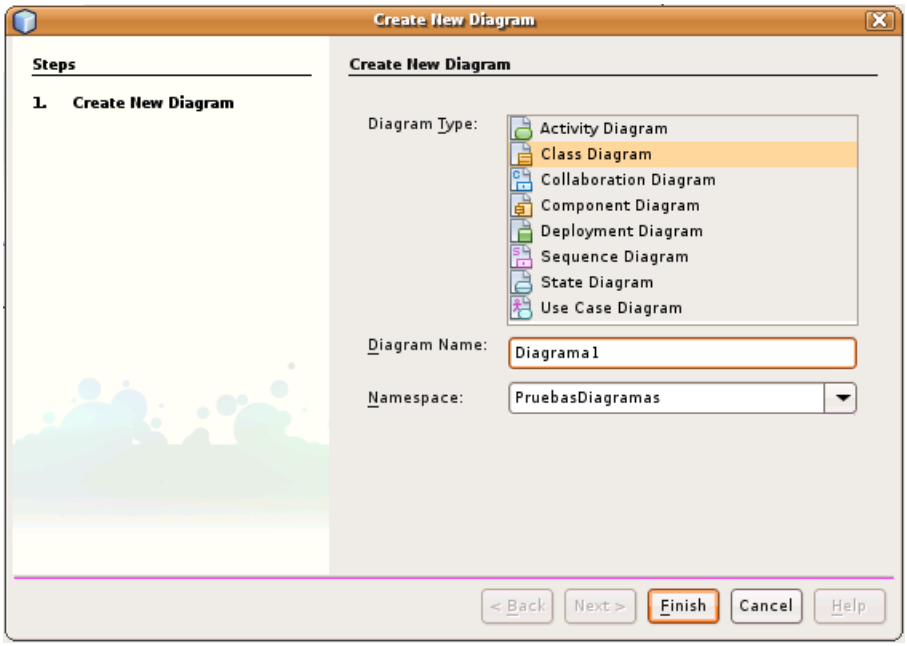
Seleccionamos el **PLATFORM-INDEPENDENT MODEL**, y presionamos **NEXT**



Ahora le damos un nombre que en este caso sera **PruebasDiagramas**



Presionamos **Finish** y seleccionamos el tipo de modelo que deseamos hacer en este caso nos centramos solo en **DIAGRAMAS DE CLASES**, lo seleccionamos y le damos un nombre **DIAGRAMA1**



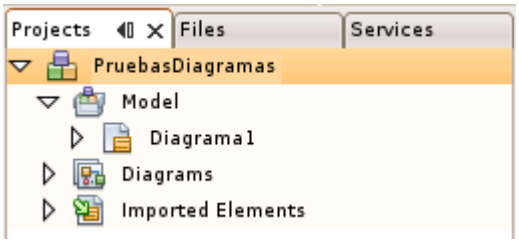
Ahora que nuestro modulo ha sido creado es necesario que sepamos algunas cosas.

a) El netbeans crea un modulo UML que puede ser agregado a cualquier proyecto que tengamos abierto en ese momento.

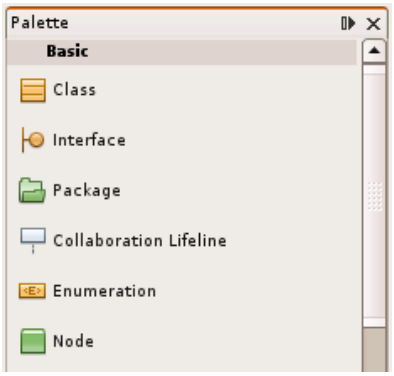
b) Cada diagrama que se genere dentro del modulo UML es independiente y cuando se generan los codigos de las clases se hacen de manera separada, por lo cual es necesario que siempre se creen con mucho cuidado para evitar errores de dependencias.

c) Al crear modulos diagramas separados es posible crear diagramas de dependencias de cada clase para ver que clases deben ser tambien traducidas a codigo.

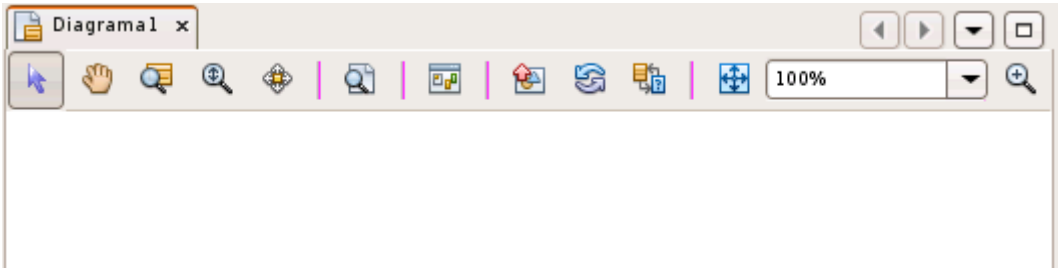
Ahora observemos que el modulo fue creado en la barra lateral izquierda



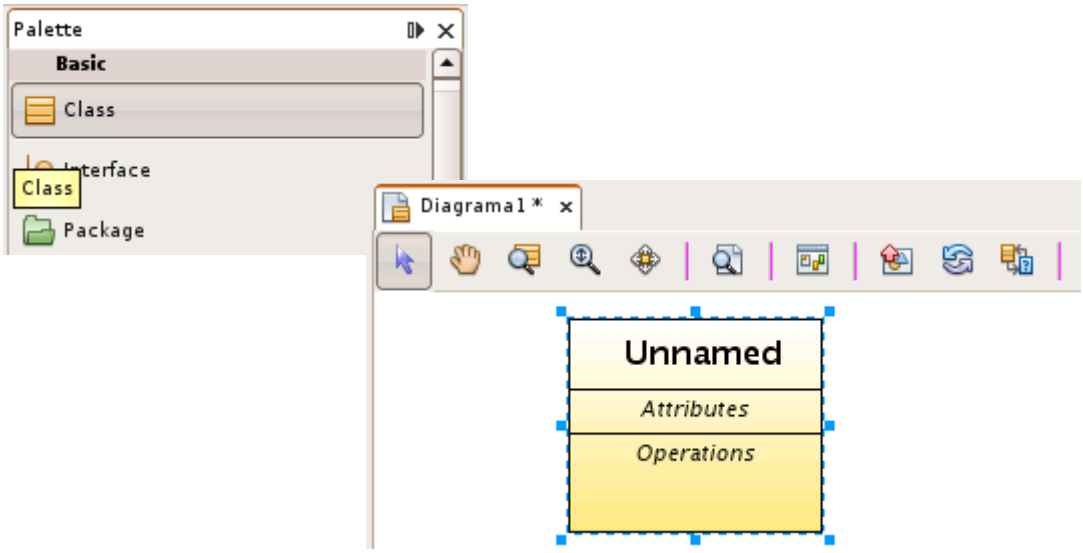
Y en la barra lateral derecha se encuentra la barra o paleta de herramientas de diseño de el diagrama, en este caso solo utilizaremos dos de ellas la de Clases y Generalizacion



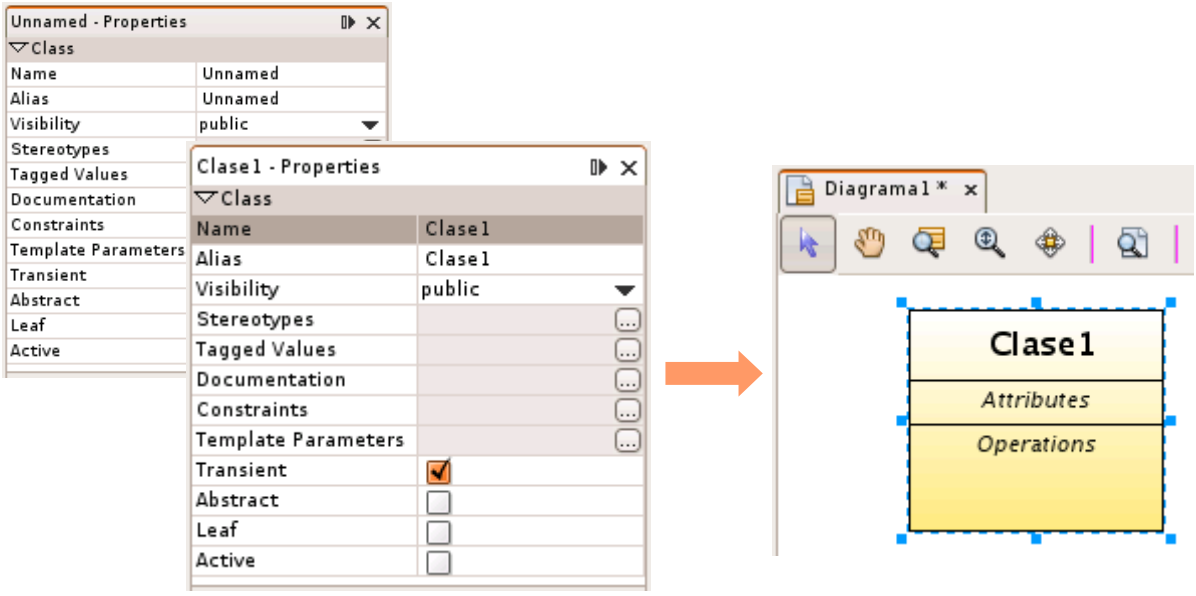
Y en la parte central tenemos el espacio en blanco o area de trabajo donde realizaremos el diseño.



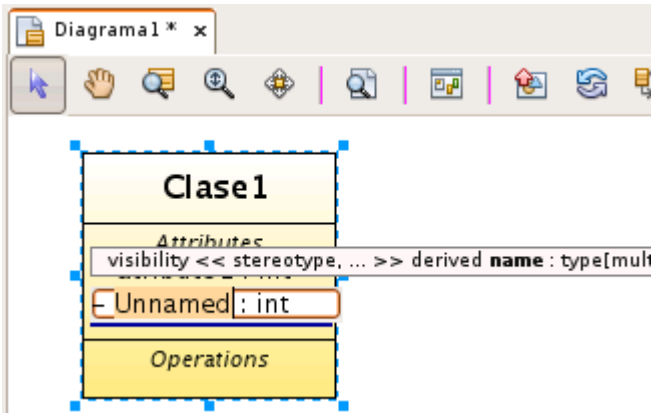
Para ello seleccionamos la herramienta **CLASS** y luego pinchamos sobre el area de trabajo para dibujarla

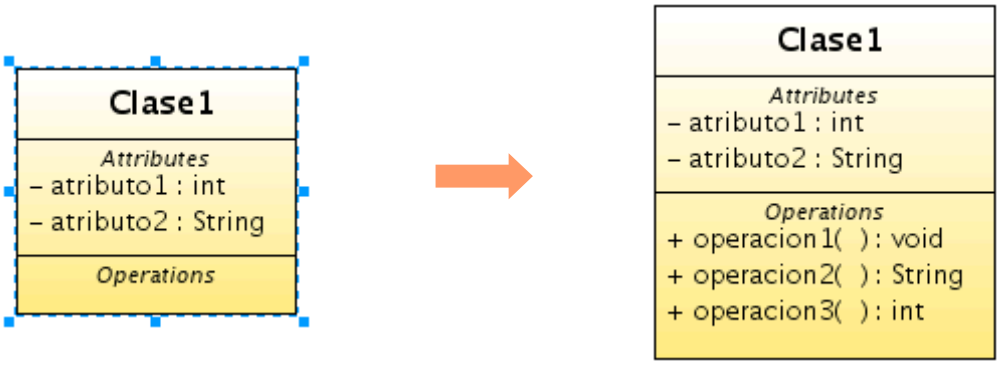


En la parte inferior a la paleta de herramientas tenemos la ventana de propiedades de la clase y en ella podemos modificar todos sus valores por defecto, seleccionar el tipo de clase y agregar comentarios de documentacion. Ahora en la clase creada modificamos el nombre de esta con la propiedad NAME, o simplemente hacemos doble click sobre el nombre de la clase en el area de trabajo, le daremos el nombre Clase1

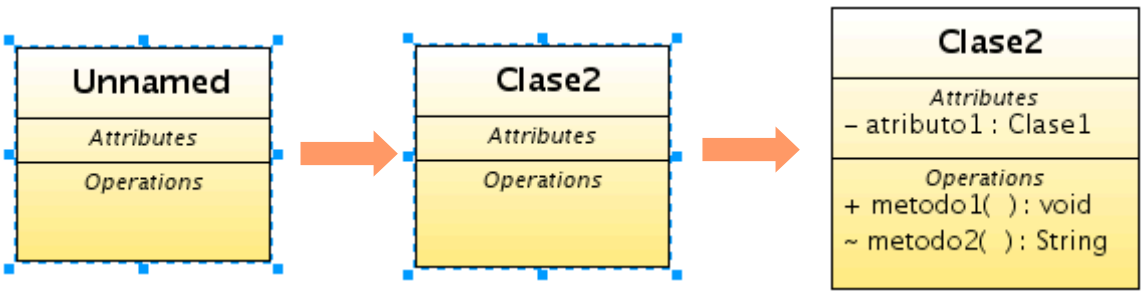


Luego comenzamos a agregarle metodos y atributos, para realizar la prueba agregamos dos atributos a los que le daremos los siguientes valores: atributo1:int, atributo2:String, para ello presionamos con el boton derecho sobre la palabra ***Attributes*** y seleccionamos **INSERT ATTRIBUTE** hacemos lo mismo para los Metodos pinchando sobre ***Operations*** y seleccionando **INSERT OPERATION**, al que agregaremos los siguientes metodos operacion1():void, operacion2():String, operacion3():int.

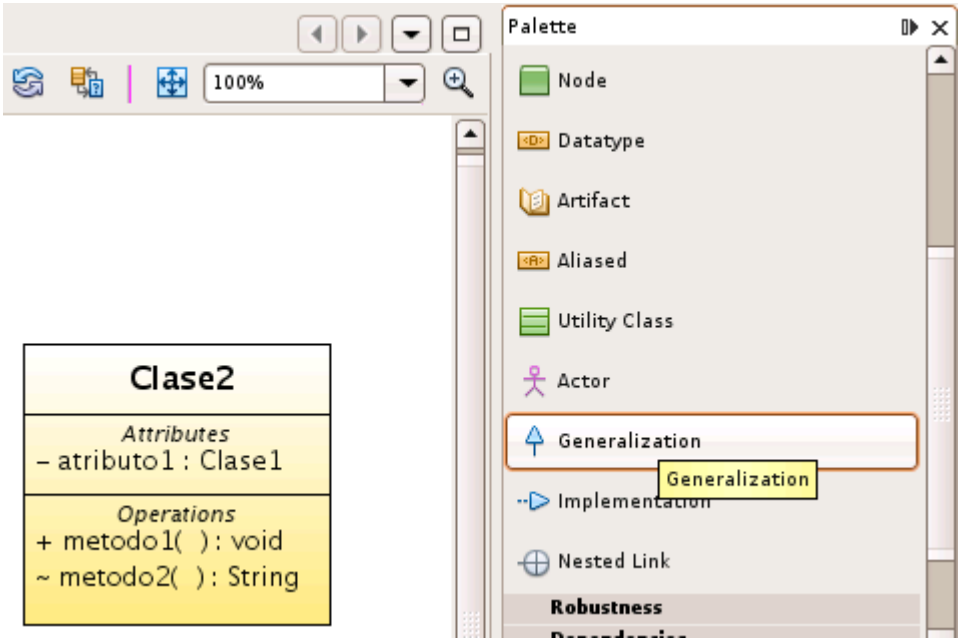




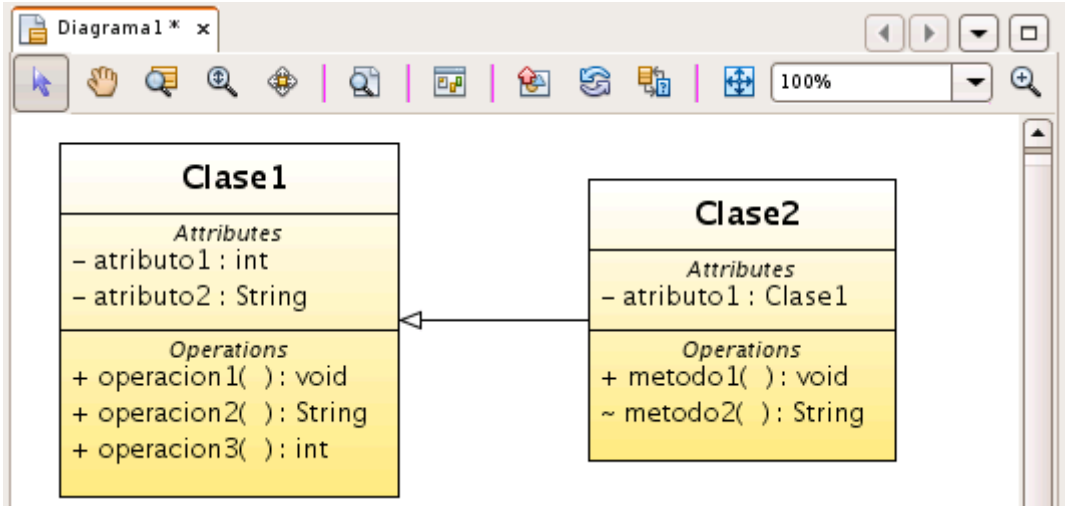
Una vez que este lista, procedemos a crear una nueva clase con los siguientes atributos: atributo1:Clase1, metodos: metodo1():void, metodo2():String.



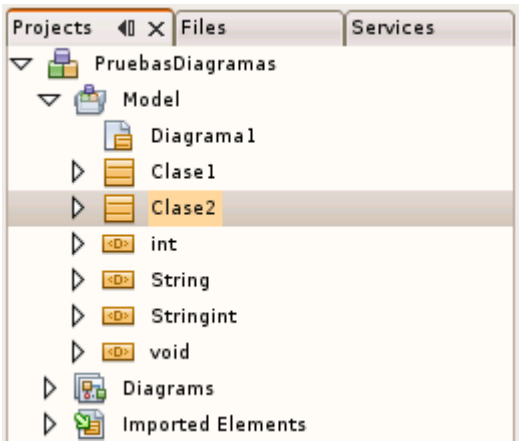
Ahora agregamos una Generalizacion para indicar que la Clase2 depende de la Clase1, o simplemente generamos un diagrama de dependencias, para esto pinchamos la herramienta en la paleta



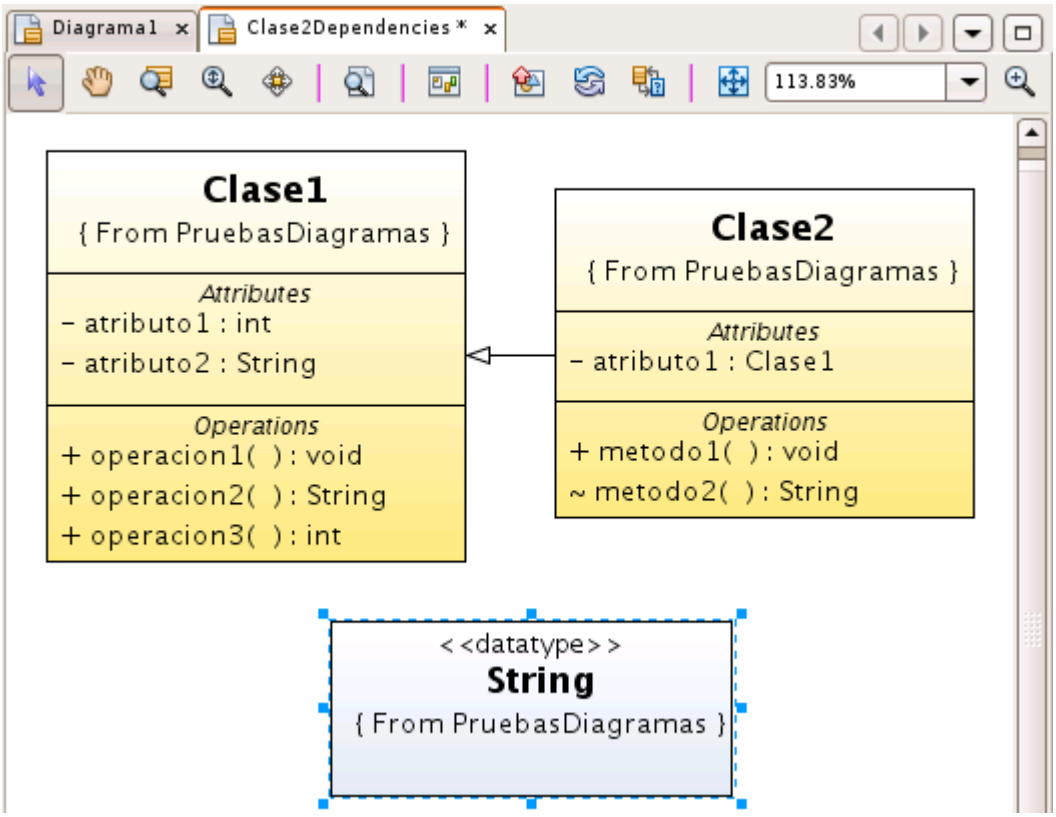
Y pinchamos desde la Clase2 a la Clase1, ya que esta depende de ella



Para generar el diagrama de dependencias pinchamos en la clase dos en este caso y seleccionamos con el boton derecho sobre el icono de la clase en la ventana de proyectos y seleccionamos **GENERATE DEPENDENCY DIAGRAM.**



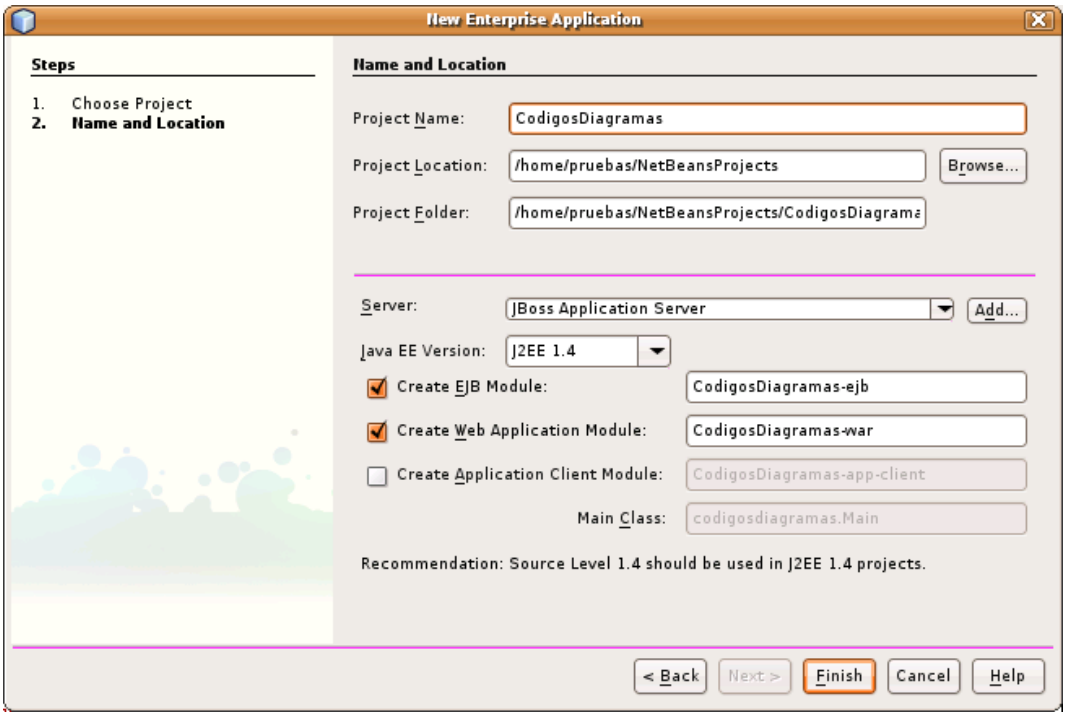
Y veremos el mismo resultado que al agregar la generalizacion, solo que en este caso estara un poco mas detallada, ya que ademas de aparecer las clases relacionadas se encuentra descrito el modulo al que pertenecen y los tipos de datos asociados a las clases.



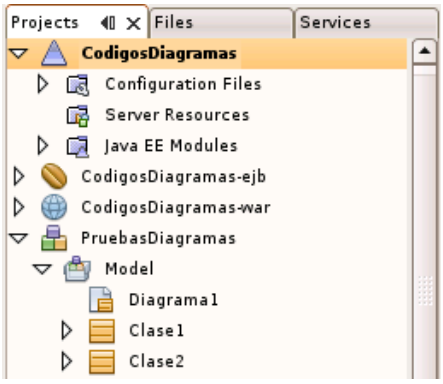
Como ya sabemos crear los diagramas de clase, procederemos a generar los codigos java para las clases, a partir de los modelos creados.

6.4 Creacion de codigo java a partir de Diagramas de Clases en proyectos existentes

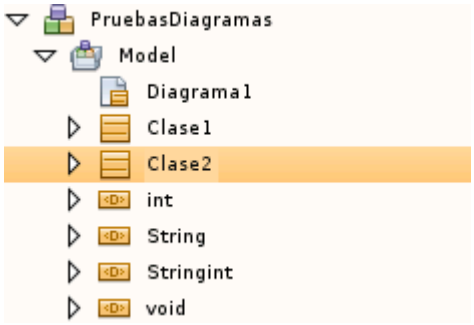
Para comenzar debemos crear una nueva aplicación JEE en la cual generaremos nuestro codigo, en este caso se generara el codigo en el modulo EJB del proyecto JEE, para esto creamos un nuevo proyecto al que le daremos el nombre de **CodigosDiagramas**



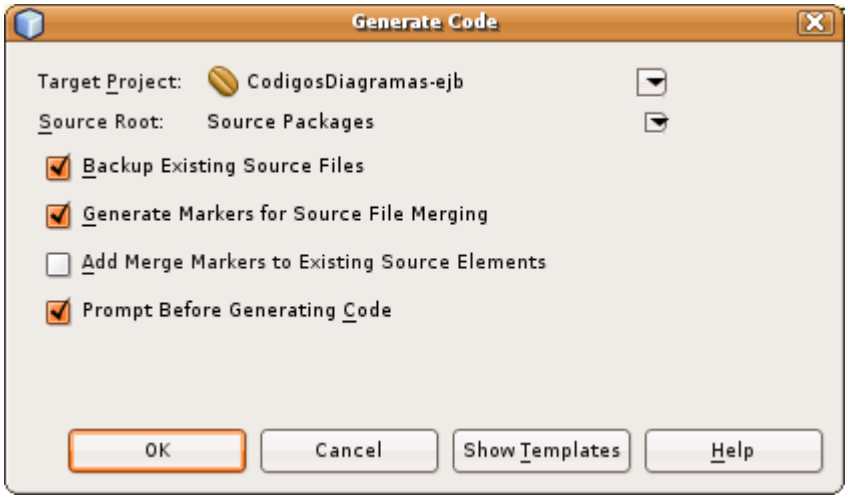
Una vez creado se generaran sus respectivos modulos en los que se pueden almacenar diferentes paquetes de aplicaciones



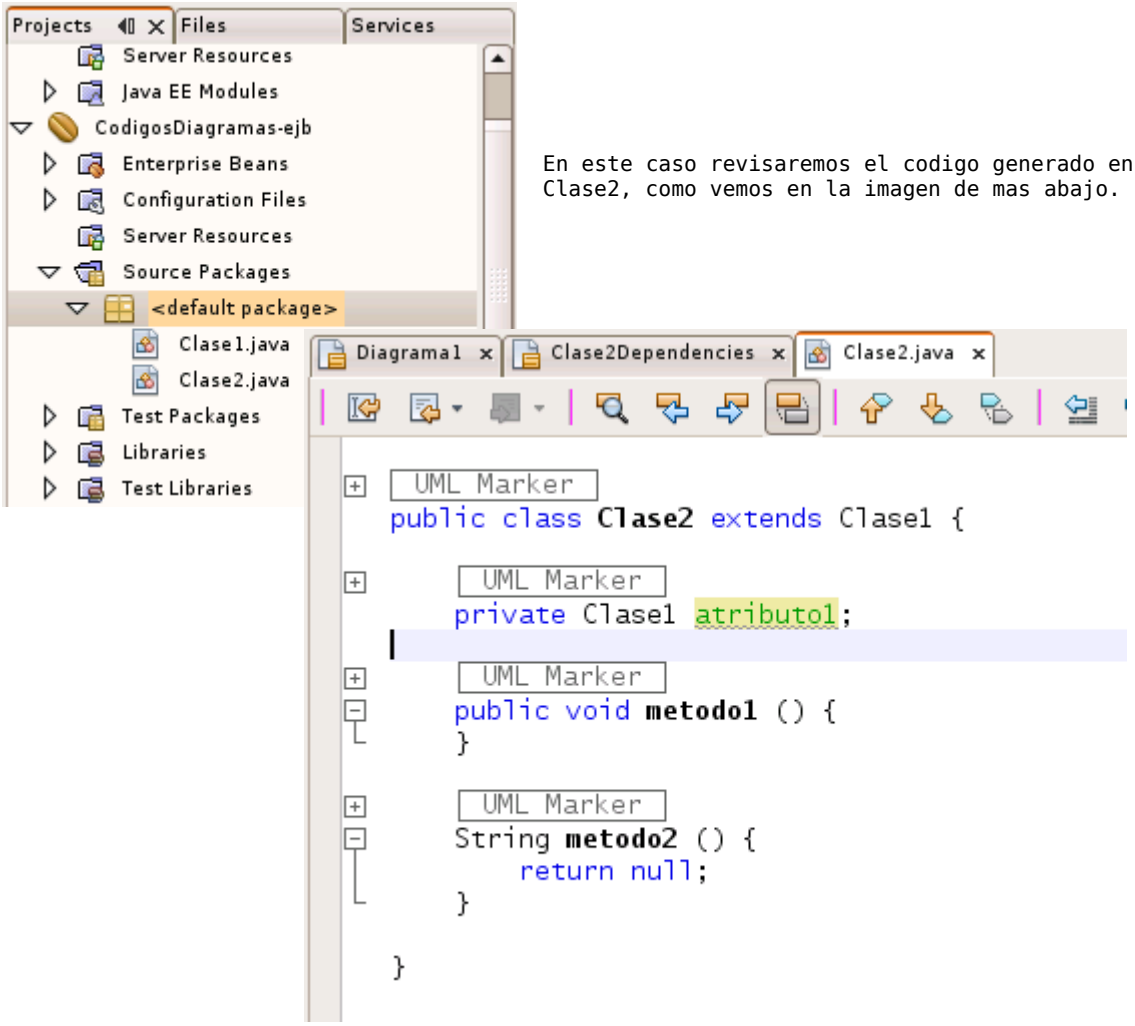
Ahora presionamos con el boton derecho en la clase que deseamos generar en este caso la Clase2 y seleccionamos **GENERATE CODE**



Cuando selecionemos la opcion deberemos seleccionar el proyecto al que deseamos enviar el codigo de nuestro modelo, para el ejemplo seleccionamos el modulo EJB del proyecto JEE que creamos



Para evitar errores en el paquete de aplicaciones del modulo EJB, debemos generar el codigo para la Clase1 ya que la Clase2 depende de ella. Cuando esten generados ambos codigos nos vamos al Item donde deben aparecer los paquetes creados para revisarlos.



En este caso revisaremos el codigo generado en la Clase2, como vemos en la imagen de mas abajo.

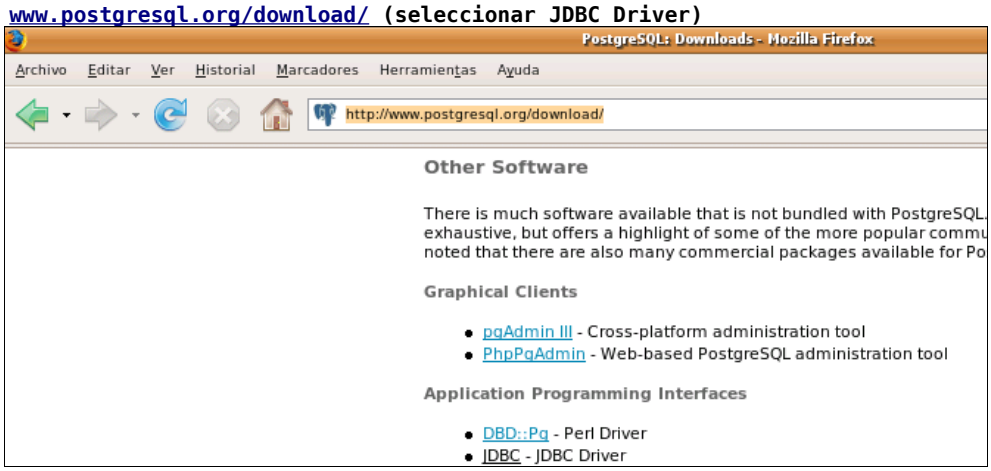
7. Netbeans 6 + PostgreSQL

A contiuacion nos centraremos en poder ocupar la herramienta de base de datos que trae netbeans con ellas podemos

- Crear tablas
- Crear vistas
- Crear Consultas
- Ejecutar codigo SQL para, insertar, modificar y eliminar registros

para ello antes deberemos descargar el driver de PostgreSQL de la pagina oficial http://www.postgresql.org/download/ o directamente entramos a http://jdbc.postgresql.org/

www.postgresql.org/download/ (seleccionar JDBC Driver)

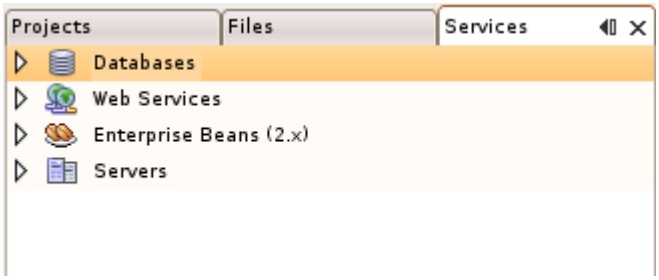


jdbc.postgresql.org (seleccionar el driver JDBC 2 EE para la version 8.0)

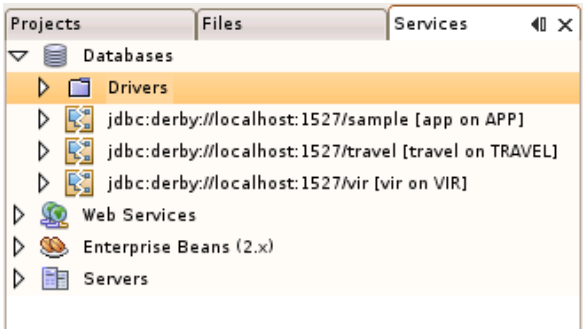
No podremos crar procedimientos almacenados o funciones ya que no trae soporte para ello, asi que basicamente nos centraremos en realizar las operaciones de mayor uso que son las antes descritas.

7.1 Conexion con el servidor de base de datos

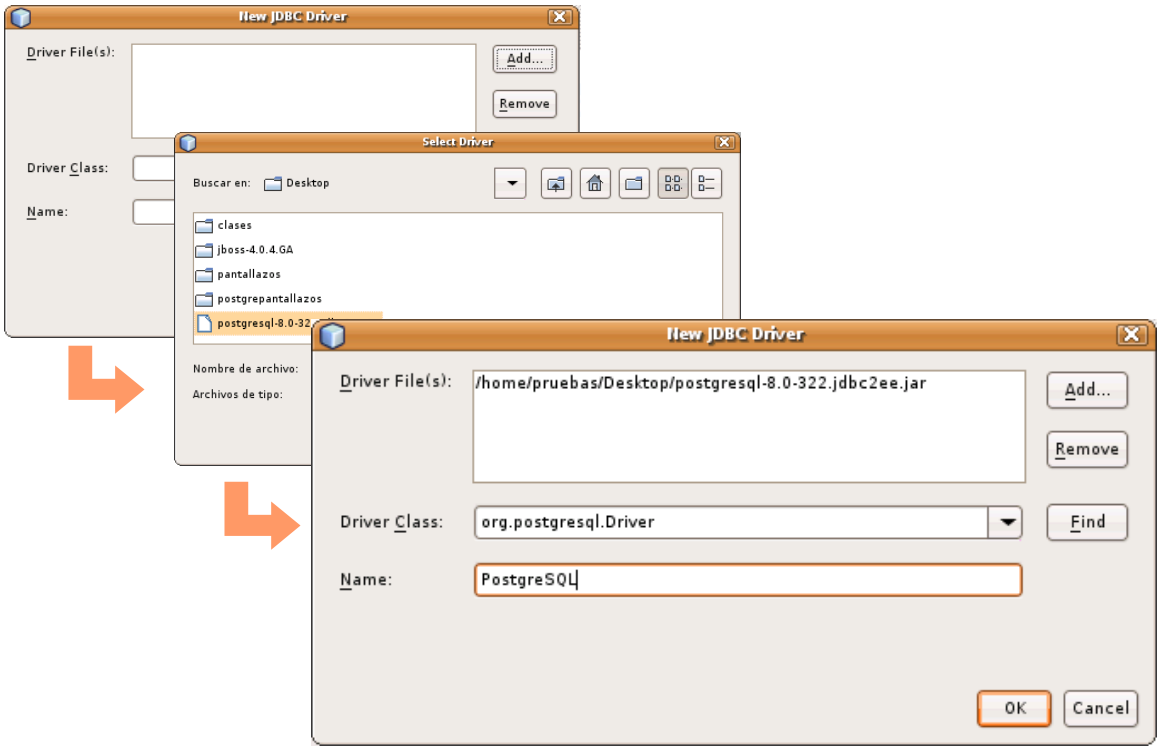
Cuando ya tengamos el driver descargado, nos vamos a la ficha Services de la barra lateral izquierda, y expandimos el icono **DATABASES**



En el tendremos las siguientes opciones

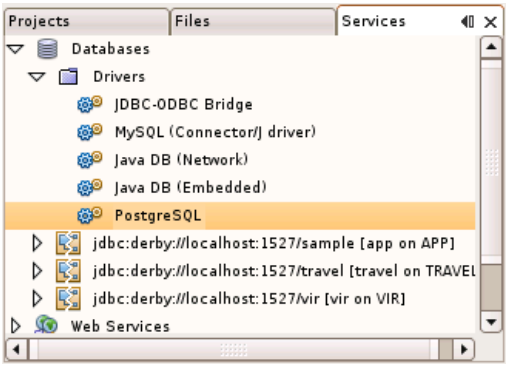


Pinchamos con el boton derecho y seleccionamos **NEW DRIVER**, ahora deberemos buscarlo y agregarlo para ello pinchamos el boton ADD y lo buscamos en el directorio donde fue descargado



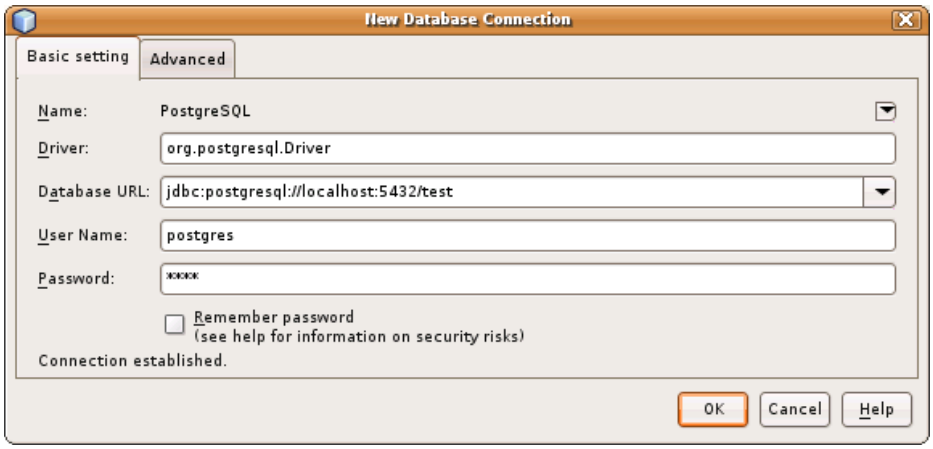
Cuando ya este seleccionado presionamos OK

Y el driver deberia haber sido agregado, podemos verificarlo expandiendo el nodo Drivers del servicio **DATABASES**

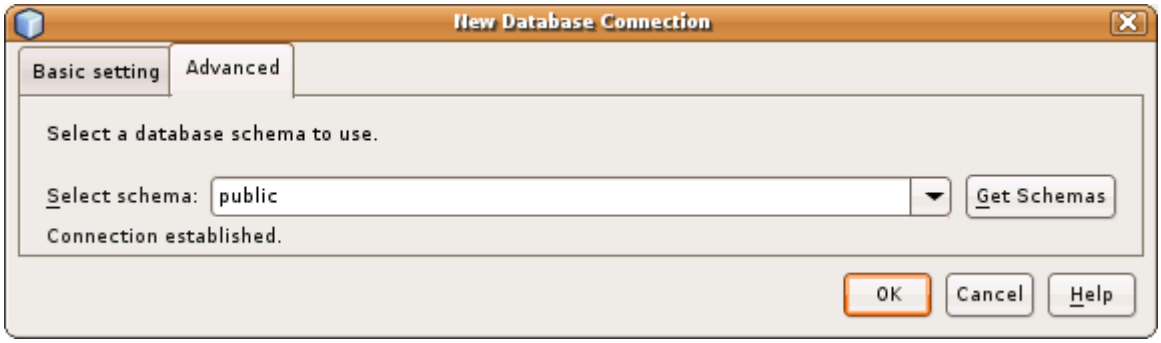


Como ya tenemos el driver agregado deberemos crear una conexión a nuestra base de datos, que se encuentra en el servidor que instalamos en el capitulo 5.

Para ello pinchamos con el boton derecho en el icono **DATABASES** y seleccionamos **NEW CONNECTION**, luego deberemos seleccionar el nombre del Driver que debera ser el que agregamos, darle la direccion de la base de datos, ingresar nuestro nombre de usuario y contraseña



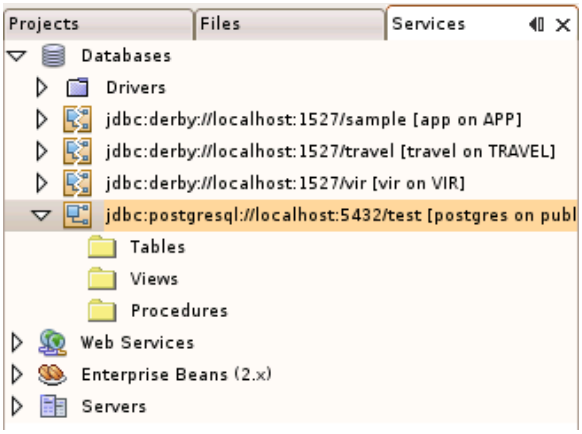
Seleccionamos la ficha **Advanced** para seleccionar el esquema con el que deseamos trabajar, que en este caso es **public**, para ello pinchamos el boton GET SCHEMAS y seleccionamos en el combo de al lado **public**



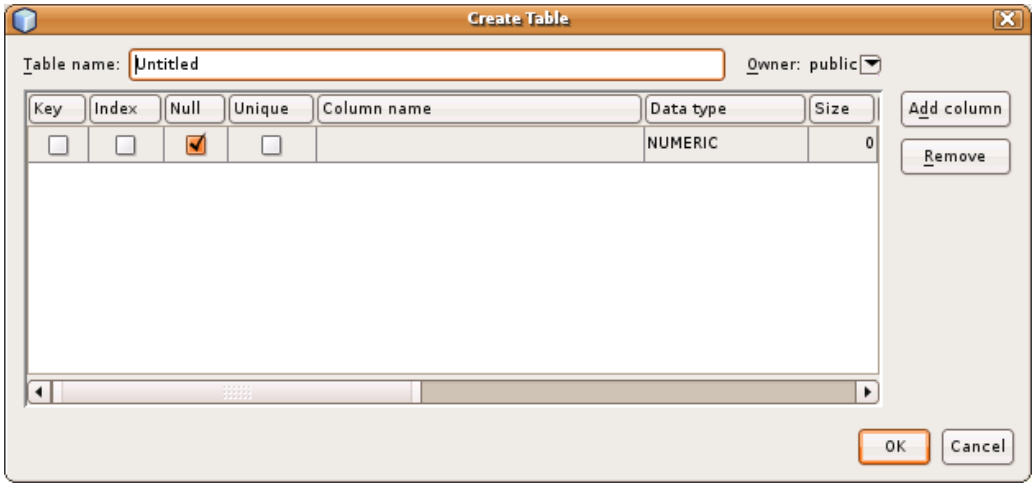
Ahora ya estamos listos para comenzar a a realizar operaciones con la base de datos.

7.2 Crear una tabla

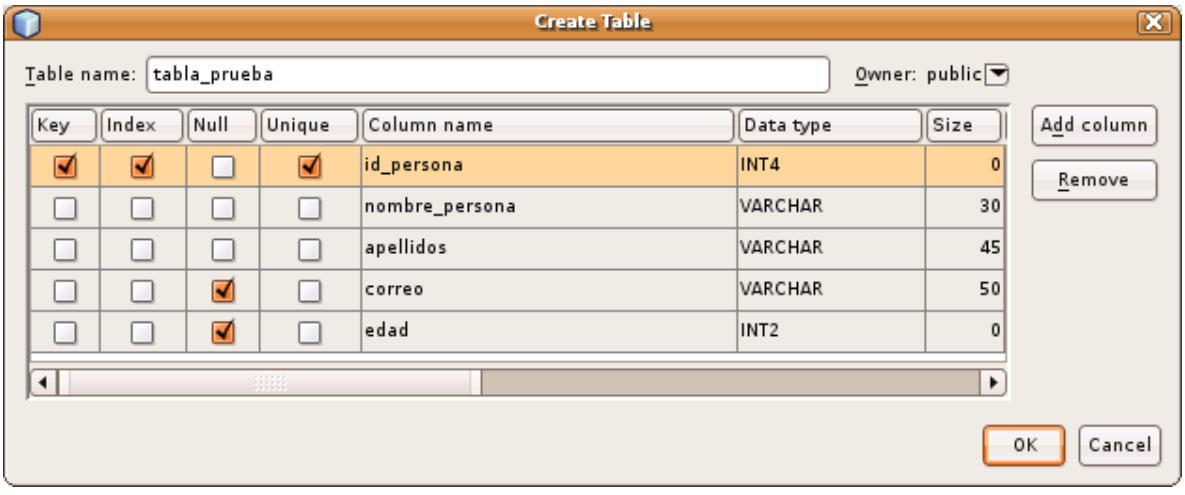
Ahora que hemos creado la conexión a la base de datos, podremos ver tres carpetas cada una representa los tres elementos a los que tenemos acceso, sin embargo no podemos crear Procedimientos (carpeta Procedure) pero si tablas y vistas



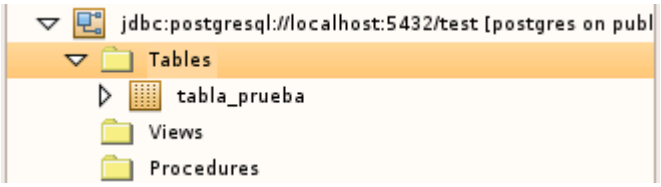
Para agregar una tabla debemos pinchar sobre la carpeta **TABLES** y seleccionar la opcion **CREATE TABLE**, nos aparecera esta ventana donde debemos agregar cada uno de los campos de nuestra tabla, seleccionar tipo de dato tipo de campo y tamaño en el caso de los tipo varchar.



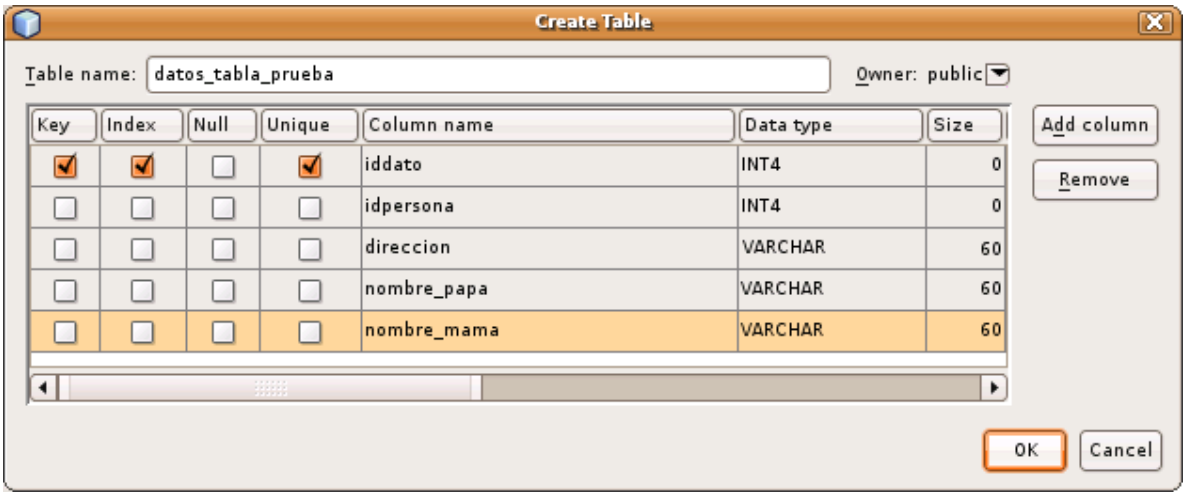
Para esta prueba crearemos una tabla llamada tabla_prueba, con los campos que aparecen aca abajo



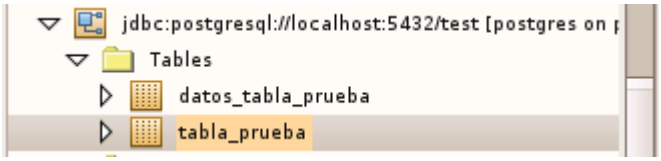
Cuando esten todos los cambios agregados presionamos el boton OK y estara lista, cuando se genere se agregara el icono de esta a la carpeta **TABLES**



Ahora crearemos otra tabla para relacionarla con la **tabla_prueba** con los siguientes campos



Cuando se genereren las dos tablas, las veremos en nuestra ficha de conexiones



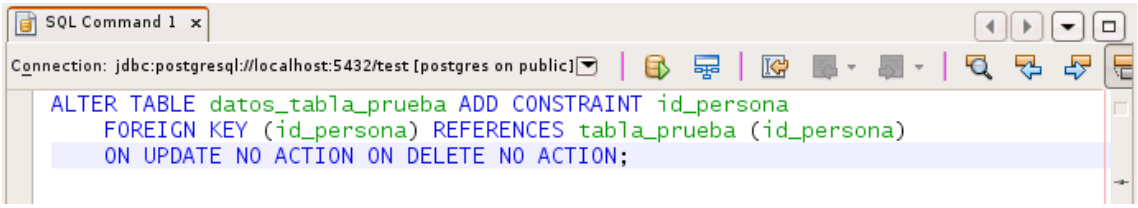
Ahora deberemos crear la clave foranea de la tabla **datos_tabla_prueba** acia **tabla_pruebas** con codigo SQL.

7.3 Ejecutar codigo SQL

Para ejecutar codigo SQL debemos pinchar con el boton derecho sobre la carpeta **TABLES** con el boton derecho y seleccionar **EXECUTE COMMAND**, con esto se nos abrira un area de trabajo para ejecutar instrucciones SQL, solo es posible ejecutar una o varias instrucciones seguidas no por parte o por selección ya que al escribir varias las ejecuta de forma secuencial, lo que hay que tener especial cuidado. Entonces tenemos el area de trabajo y la ventana OUTPUT donde veremos los resultados de cualquier tipo de consulta que realizemos

Asi ahora ejecutaremos el codigo para crear la clave foranea de la tabla **datos_tabla_pruebas** y para ingresar un dato a cada una para posteriormente crear una vista.

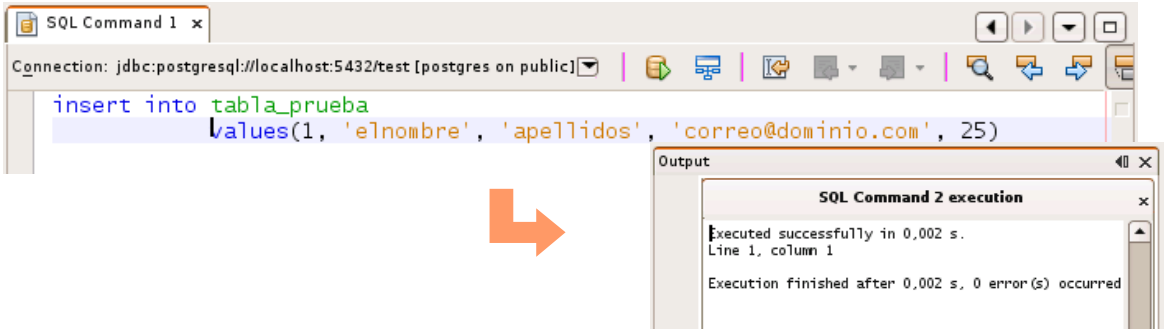
El icono  ejecuta el codigo.



Cuando sea ejecutado en la ventana OUTPUT (barra lateral izquierda) podemos ver la clave agregada a la table expandiendo el icono FOREIGN KEYS de la tabla **datos_tabla_pruebas**

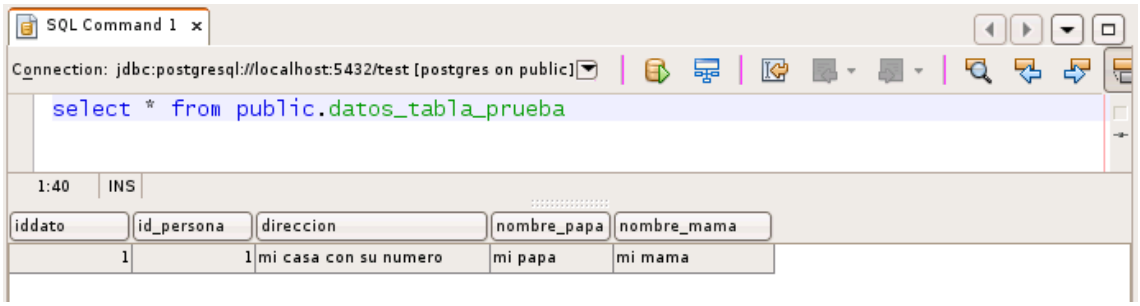
Ahora borramos el codigo ke ejecutamos para que no sea ejecutado nuevamente y creamos los codigos para ingresar un dato a cada uno de las tablas

Ingresa dato a la **tabla_prueba**



Ingresa dato a **datos_tabla_pruebas**

Cuando esten ingresados realizamos un select a cada una de las tablas para verificar que esten ingresadas.

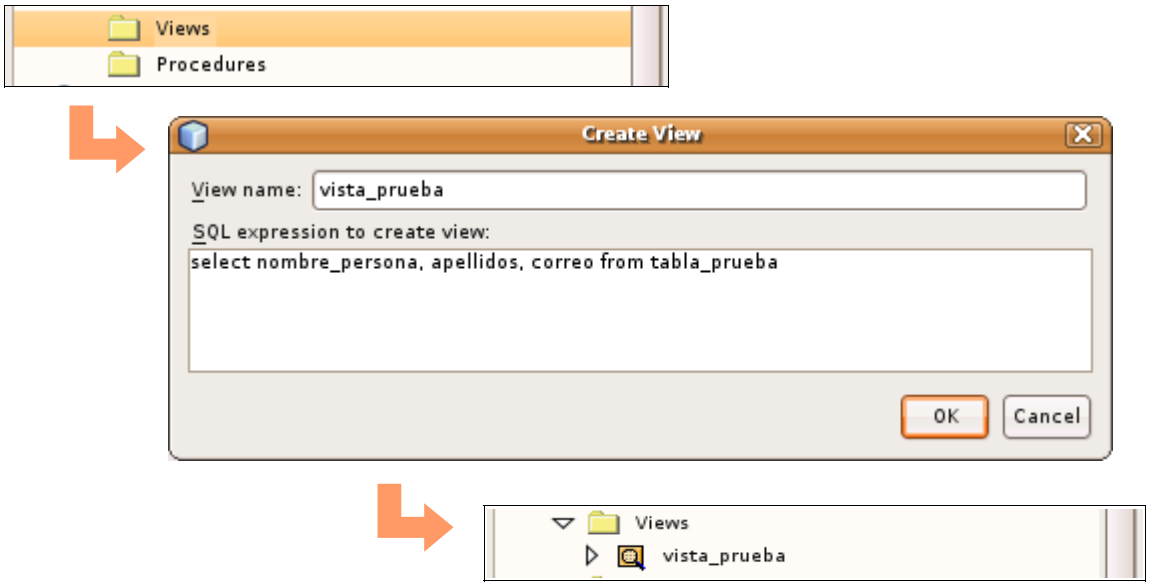


Como ya tenemos por lo menos un dato ingresado a cada uno, deberemos hacer una vista para ver los datos de ambas tablas que estan relacionados.

7.4 Crear una vista

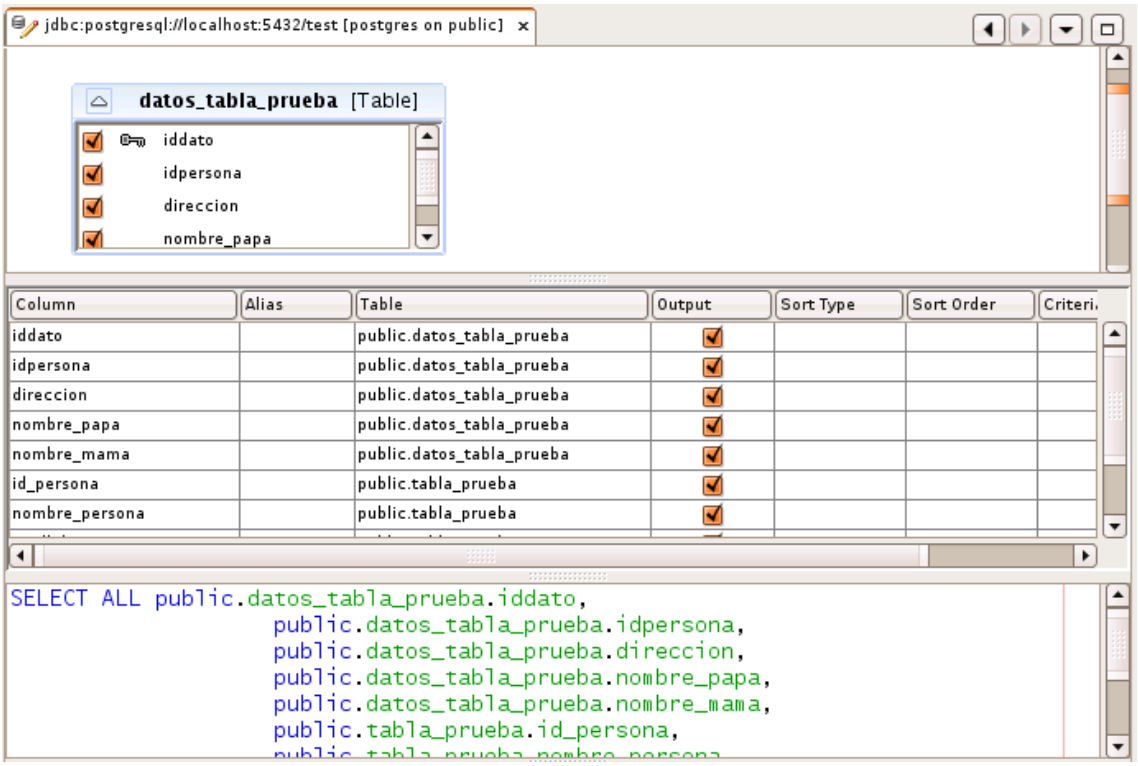
Para ello tenemos dos formas de hacerla, la primera es con codigo, y la segunda es con el editor de consultas.

Para la primera debemos solamente hacer click con el boton derecho sobre la carpeta VIEWS y seleccionar **CREATE VIEW** e ingresar el nombre de la vista y su respectivo codigo sql.

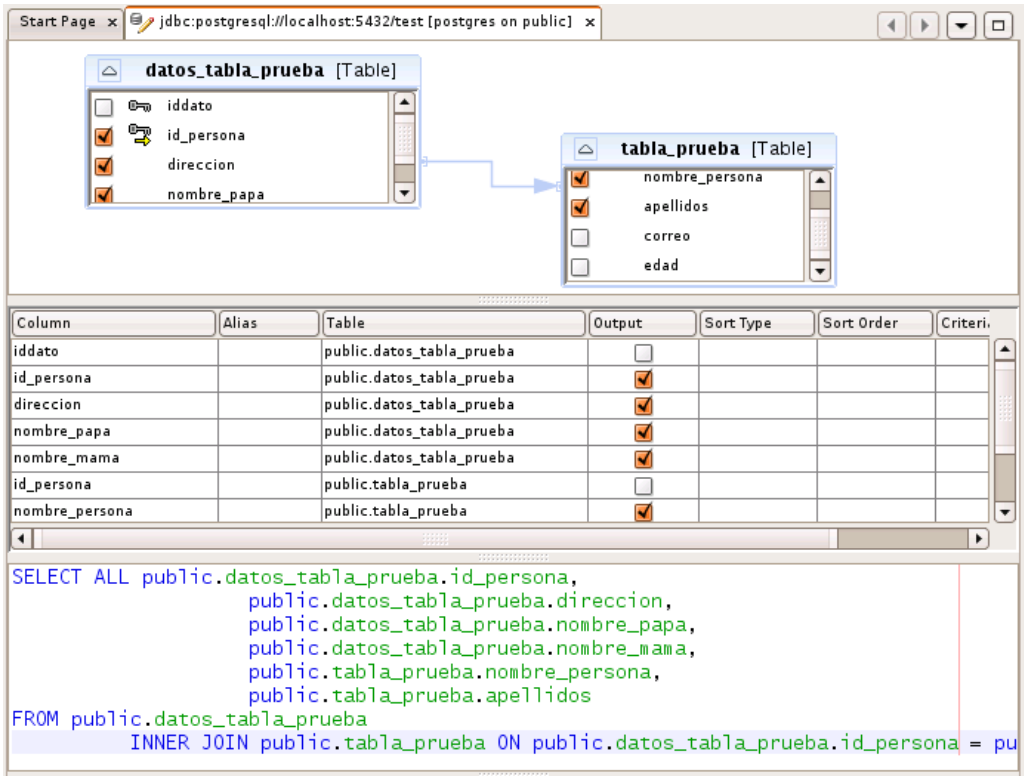


7.5 Crear una consulta con el editor

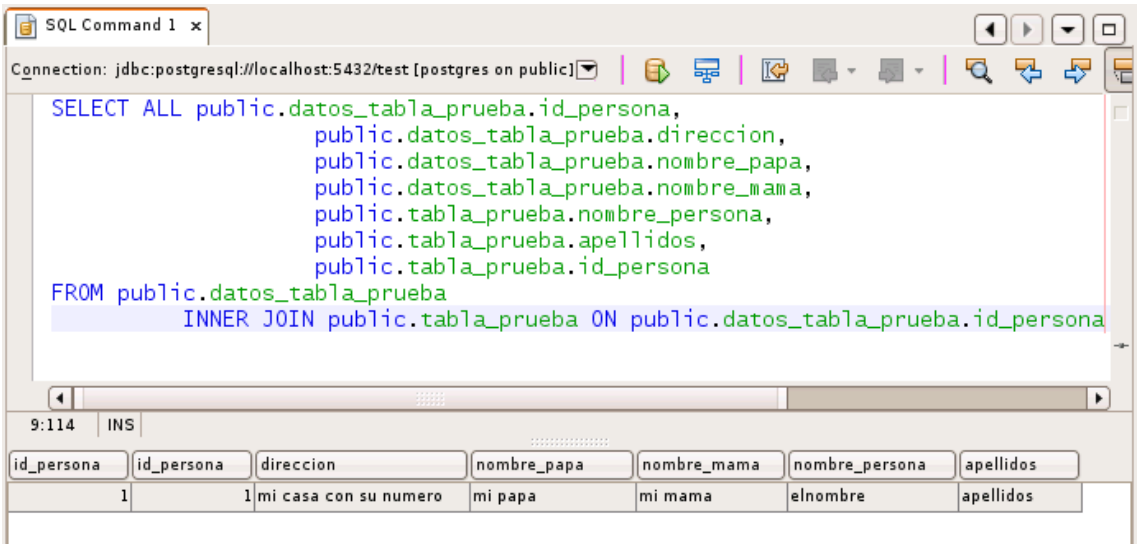
Para realizar una consulta con el editor basta con presionar con el boton derecho sobre cualquiera de las tablas, y seleccionar **DESIGN QUERY**, a continuacion nos abrira una ventana donde aparecera la tabla seleccionada solamente debemos seleccionar los campos que deseamos mostrar e ir editando sus parametros como orden, condiciones, etc. En este caso pinchamos sobre **datos_tabla_prueba** y deleccionamos la opcion antes nombrada.



Si deseamos agregar otra tabla, como **tabla_pruebas**, pinchamos con el boton derecho en el area blanca al lado del cuadro de la tabla **datos_tabla_pruebas** y seleccionamos **ADD TABLE** donde agregamos la **tabla_pruebas**, cuando se agregue aparecera una linea que muestra la relacion entre ambas tablas.



Cuando hallamos seleccionado los campos de seamos debemos ejecutar la consulta presionando con el boton derecho en el area blanca y seleccionar **RUN QUERY**, en el caso de enviar algun error bastara solamente con copiar el codigo que ha sido creado y pegarlo en una nueva area de trabajo, o editor de codigo SQL, para probarla, y luego si deseamos, convertirla en vista.



8. Ayuda con posibles problemas

8.1 Ejecucion de comandos

En algunos casos al copiar el texto desde este manual al terminarl de gnome, algunos caracteres son copiados de manera incorrecta es el caso de los guiones (-) como cuando deseamos agregar el atributo PREFIX que en vez de copiarse como doble guion se copia como uno simple

- **./configure -prefix=.... MALO**
- **./configure –prefix=.... Correcto**

De igual forma sucede con el carácter comilla (“) en vez de ser copiado de igual forma al final y al inicio del atributo que estamos describiendo se copia de manera diferente.

Por eso es recomendado revisar antes de que forma se han copiado y si es necesario volver a escribirlos.

8.2 Lenguaje PLPGSQL

Cuando este instalado el motor de base de datos, en algunos casos el lenguaje plpgsql no es instalado por lo que se deberan crear las siguientes funciones para poder agregar el lenguaje.

Funcion 1:

```
-- Function: plpgsql_call_handler()

-- DROP FUNCTION plpgsql_call_handler();

CREATE OR REPLACE FUNCTION plpgsql_call_handler()
RETURNS language_handler AS
       '$libdir/plpgsql', 'plpgsql_call_handler'
LANGUAGE 'c' VOLATILE;
```

Funcion 2:

```
-- Function: plpgsql_validator(oid)

-- DROP FUNCTION plpgsql_validator(oid);

CREATE OR REPLACE FUNCTION plpgsql_validator(oid)
RETURNS void AS
       '$libdir/plpgsql', 'plpgsql_validator'
LANGUAGE 'c' VOLATILE;
```

Funcion 3:

```
-- Language: plpgsql

-- DROP LANGUAGE plpgsql;

CREATE TRUSTED PROCEDURAL LANGUAGE 'plpgsql'
HANDLER plpgsql_call_handler
VALIDATOR plpgsql_validator;
```