

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. Apac	che 2.0.55	
	1.1. Paquetes necesarios 1.2. Instalación	3 3
2. JDK-	+JRE+JEE	
	2.1. Archivos necesarios	4
	2.2. Instalación de Java Runtime Environment (JRE)	4
	2.3. Instalación de Java Development Kit (JDK)	5
	2.4. Instalación de Java Enterprise Editión (JEE) 2.5. Definición de Variables Globales	6 7
3. JB09	SS 4.0.4	
	3.1. Referencia rápida	8
	3.2. ¿Que es mod_jk?	8
	3.3. Archivos necesarios	8
	3.4. Integracion con Apache 2.0.55	8
4. Cuer	ntas de Usuario JBOSS	
	4.1. Creación de una cuenta	1
	4.2. Modificación del archivo jboss-service.xml	1
	4.3. Hota Mundo!, apticación de prueba	10
5. Post	tgreSQL	
	5.1. Librerias necesarias	24
	5.2. Instalación	24
	5.3. Configuración del usuario postgres y base de datos IESI	24
	5.4. Configuracion del servidor de Base de Datos	2
6. Neth	peans 6 + UML	
	6.1 Creacion de Documentacion	2
	6.2 Creacion de Diagramas de Clase con Ingenieria Reversa	28
	6.3 Creacion independiente de Diagramas de Clases	32
	6.4 Creacion de codigo java desde Diag. de Clases en proyectos existentes	3
7. Neth	peans 6 + PostgreSQL	
	7.1 Conexion con el servidor de base de datos	4
	7.2 Crear una tabla	4
	7.3 Ejecutar codigo SQL	44
	7.4 Crear una vista	4
		4
8. Ayud	a con posibles problemas	
	8.1 Ejecucion de comandos	49
	8.2 Lenguaje PLPGSQL	49

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.

_ 3

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 <u>www.java.sun.com</u> 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)



4

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

- 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:



_ 5

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

java EE 5 Tools Bundle Update 3 Beta Installer	
NetBeans IDE 6.0 Beta 1 Installation Choose the installation folder and JDK for the NetBeans IDE.	S <u>NetBeans</u> IDE RO
Install the NetBeans IDE to:	
/usr/local/netbeans-6.0beta1	B <u>r</u> owse
IDK for the NetBeans IDE:	
/usr/lib/jvm/jdk1.6.0_03	▼ Br <u>o</u> wse
	< <u>B</u> ack Next > Cancel

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.

1	java EE 5 Tools Bundle Update 3 B	eta Installer	
Sun Java System A Choose the insta	pplication Server 9.1 Installation allation folder and server properties.		Se <u>NetBeans</u> IDE 6.0
Install Sun Java Sy	stem Application Server 9.1 to:		
/usr/local/SUNWa	opserver		B <u>r</u> owse
ĮDK for application	server:		
/usr/lib/jvm/jdk1.6.	0_03		Browse
Admin <u>U</u> sername:	admin]	
Admin <u>P</u> assword:	жылылыны	(default: adminadmin)	
R <u>e</u> type Password:	*****)	
HTTP Port:	8080		
HTTP <u>S</u> Port:	8181		
<u>A</u> dmin Port:	4848		
			< Back Next > Cancel

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.

_ 7

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.
- Crear la configuración del mod_jk
 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).

#2: Descargar mod_jk 1.2.x (mod_jk-1.2.25-httpd-2.2.4.so) o de la pagina PASO 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 modjk al final del archivo aue use APACHE HOME/conf/httpd.conf agregar las siguientes lineas:

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

_ 8

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.

<u>PASO #5:</u> Configurar workers En la carpeta APACHE HOME/conf, crear el archivo workers.properties y escribir lo siguiente:

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.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.host= localhost
worker.node2.type=ajp13
worker.node2.type=ajp13
worker.node2.type=ajp13

Define list of workers that will be used

9

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 ajpl3 worker
/jmx-console=loadbalancer
/jmx-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/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

1		Opciones	de los usuarios	X
	Nombre	Nombre de usuario	Directorio personal	🕂 🕂 Añadir usuario
	🦲 jboss1	jboss1	/home/jboss1	
	S root	root	/root	<u>Propiedades</u>
				Borrar
				estionar grupos
	<			
	🕜 Ayuda			<u>C</u> errar

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**:

1	5		Cuenta de usuario nueva	X	
	Cuenta	Privilegios del us	suario Avanzado		
	Config	uración básica			
	Nombre de <u>u</u> suario: mfabarca				
Nombre <u>r</u> eal: Miguel Abarca					
	<u>P</u> erf	il:	Usuario del escritorio		
				1	
	Inform	ación de contacto	•		
	Ubio	cación en la o <u>fi</u> cin	ia:		
	Teléfono del trabajo:				
	Teléfono del <u>d</u> omicilio:				
	Contraseña				
	 Establecer la contraseña <u>a</u> mano 				
	с	on <u>t</u> raseña del usu	ario:		
	с	on <u>fi</u> rmación:	••••••		
	0	Generar una cont	traseña aleato <u>r</u> ia		
	с	ontraseña estable	ecida a:		
l					
			🔀 Cancelar 🛛 🛁 Acepta	ar	
				_	

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:

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

0	New Project	×
Steps	Choose Project	
1. Choose Project 2	Categories: Java Web Enterprise SOA NetBeans Modules Samples	Projects: A Enterprise Application Enterprise Application with Existing Sources EJB Module EJB Module with Existing Sources Enterprise Application Client Enterprise Application Client with Existing Sources Packaged Archive
		I
	Description:	
NetBeans IDE 6.0	Creates a new enterprise application in a project and Web application project in the IDE-generated Ant build script to build and	standard project. You can also create an EJB module enterprise application. A standard project uses an d run your projects.
		<pre><<u>Back</u> Next > Einish Cancel Help</pre>

• Enterprise> Enterprise Aplication

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

\bigcirc		Add Server Instance	3
Ste	ps	Server Location	
1. 2. 3.	Choose Server Server Location Instance Properties	Installation Location: Browse	
	NetBeans IDE 6.0		
		⊗ Provide a valid JBoss Application Server 4 or 5 Installation Location	
		< <u>Back</u> Next > <u>Finish</u> Cancel <u>H</u> elp)

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

Û	Add Server Instance	X
~	Choose jBoss Server's Installation Location	\mathbf{X}
Steps 1. Choose Server 2. Server Location 3. Instance Properties	Buscar en: ☐ / ▼ ☐ jboss-4.0.4.6A	
	 ▷ □ bin ▷ □ client ▷ □ docs ▷ □ pruebaCarpeta ▽ □ server ▷ □ all ▷ □ default 	
NetBeanside 6.0	minimal iib Nombre de archivo: /usr/local/jboss-4.0.4.GA Archivos de tipo: /JBoss Server Installations Locations	▼ <u>C</u> hoose <u>C</u> ancelar
	< <u>B</u> ack Next > <u>Finish</u> Can	cel <u>H</u> elp

Seleccionamos y continuamos la creación de nuestra aplicación de ejemplo:

0	A	dd Server Instance
Steps	Instance Prop	erties
1. Choose Server	Select Server	Configuration
3. Instance Properties	<u>D</u> omain:	default
	Domain Path:	/usr/local/jboss-4.0.4.GA/server/default
	<u>H</u> ost:	localhost
	<u>P</u> ort:	8080
NetBeans IDE 6.0		
		< Back Next > Finish Cancel Help

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!

Name and Location				
Proiect Name:	HolaMundo			
Project <u>L</u> ocation:	/home/jboss1/NetBeansProjects Browse			
Project <u>F</u> older:	/home/jboss1/NetBeansProjects/HolaMundo			

A continuación seleccionaremos nuestro servidor de aplicaciones JBOSS

Server:	JBoss Application Server	
Įava EE Version:	J2EE 1.4	

 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:

🗹 Create EJB Module:	mfabarca/HolaMundo-ejb
🗹 Create <u>W</u> eb Application Module:	mfabarca/HolaMundo-war
Create Application Client Module:	(HolaMundo-app-client
Main <u>C</u> lass:	(holamundo.Main

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

Name and Location	
EJB <u>N</u> ame: saludo	Ahora nuestras
Project: mfabarca/HolaMundo-ejb	EJB y
Location: Source Packages	elemento mismo a por lo
Pac <u>k</u> age: saludo	también aplicaci
Session Type:	
Stateless	Ya crea
⊖ State <u>f</u> ul	beans de para el
Create Interface:	derecho
🗹 <u>R</u> emote	creamos

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 nismo 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.

_ 18

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:

\mathbf{O}	Add Business Method	X
Na <u>m</u> e: Return Type:	saludo String	
Parameters	Exceptions	
Name	Type Final Add Remove	
Use in Interfac	e: 🔿 Local 💿 Remote 🔿 Both	
	ОК Саг	cel

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

publ	<pre>ic String saludo() {</pre>	
	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.

Name and Loo	ation
Class <u>N</u> ame:	saludo
<u>P</u> roject:	mfabarca/HolaMundo-war
Location:	Source Packages
Pac <u>k</u> age:	saludo
<u>C</u> reated File:	;s1/NetBeansProjects/HolaMundo/mfabarca/

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

👕 🛛 🛛 Call Enterprise Bean
Choose the enterprise bean you want to call.
▽ 📎 mfabarca/HolaMundo-ejb
🔊 saludoSB
Service Locator Strategy
Generate Inline Lookup Code
○ Existing Class
Convert Checked Exceptions to RuntimeException
Reference Name: saludoBean
Referenced Interface: O Local 💿 Remote
OK Cancel Help

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.

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

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:



Hola Mundo!!

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

							dist - Na	vegador de arc	:hivos
Archivo	<u>E</u> ditar	Ver	<u>l</u> r a	<u>Marcadore</u>	s Ayuda	9			
dtrás	Adel	ante	Ŧ	<mark>॰</mark> Subir	Detener	C Recargar	Carp	ita personal	Equipo
	Lugar:	/hon	ne/mfak	arca/NetBean	sProjects	/HolaMundo	o/dist		
Lugares •	•		8						
🔄 Escrit	orio			Ho	laMundo.	ear			
🛄 Sisten	na de arc	chivos							
🔄 Volum	en 4,0 Gi	ib							

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

\$HOME/public_html/deploy

						d	eploy - Nave	gador de a	rchivos	
Archivo	<u>E</u> ditar	Ver	<u>I</u> r a	<u>M</u> arcador	es Ayuda	9				
dtrás	Adel	ante	-	슅 Subir	X Detener	C Recargar	Carpeta	o personal	Equipo	
	Lugar:	/hon	ne/mfa	barca/publ	ic_html/de	ploy				
Lugares ·	-		8			•				
🔟 Escrit	orio			н	olaMundo.	ear				
🛄 Sister	na de arc	hivos								
🔄 Volum	en 4,0 Gi	ь								

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



Apache/2.0.55 (Unix) mod_jk/1.2.25 Server at 192.168.5.107 Port 80

5.PostgreSQL

5.1. Librerias necesarias

Para esta instalacion debemos descargar primeramente el tar.gz v8.0.1 de la pagina oficial de postgres <u>www.postgresql.org</u>, 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.

_ 24

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:



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 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.
<pre>@param nomParametro DescripcionParametro</pre>	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 Summar	У
<u>BeansDePruebaBean</u>	

Seleccionamos la clase principal en este caso BeansDePruebaBean

Method	Summary
void	<pre>ejbActivate()</pre>
void	<u>ejbCreate</u> ()
void	<pre>ejbPassivate()</pre>
void	<u>ejbRemove</u> ()
void	<u>Metodol</u> (int parametrol, java.lang.String parametro2, int parametro3) descripcion del metodo
java.lang.String	<pre>Metodo2(java.lang.String parametro1, int parametro2)</pre>
void	<pre>setSessionContext(javax.ejb.SessionContext aContext)</pre>

Y en la descripcion detallada de cada uno de ellos:

Metodo1
public void Metodol (int parametrol, 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

Relead C		instance	Jett		Sourch
<u>Reload</u> C	acalog			_	<u>search</u>
Install	Name 🔻	Category	Source	_	LIMI
	RESTful Web Services	Web & Java EE	ß	•	OML
	Ruby and Rails	Ruby			NotPeans Castified Plugin
	Ruby Extra Hints	Ruby			
	Ruby RSpec Support	Ruby			Version: 1.2.2
	Spring Framework Support	Web & Java EE	ß		Date: 18-02-08 Source: NetBeans
	Sprint Mobility Extensions	Sprint Mobility	କିଳି		Homepage: http://uml.netbeans.org/
	Sun Grid Access	Sun Grid	ß		
	Sun Grid Java Support	Sun Grid	ß		Plugin Description
	Sun Java System Portal Server	PortalPack	ß		For the list of included fixed bugs please check
	Sun Java System Web Server 7.0	Web & Java EE			http://wiki.netbeans.org/wiki/view/NetBeans6.0PatchesInfo.
	Sun Java Wireless Toolkit 2.5.2 f	Sun Microsyste		-	The UML plugin enables model-driven analysis, design and implementation
	Task List Editor Support	Bridges			through the use of the Unified Modeling Language (UML). It supports all major
	UML	UML			reporting to user interaction and process modeling, on through to code
	User Tasks	Task List	Ũ		generation and deployment of applications.
Ιŏ	Visual Mobility Designer	Mobility	Õ		The UML plugin currently does not support Macintosh(tm) platforms due to known
I õ	Visual Web JSF Backwards Com	Web & Java EE	Ĩ		issues related with the delay in the Apple(tm) JDK development. Note that
L Ö	Visual Web JSF Portlet Support	PortalPack	ß		macintosn users can still use the UML plugin, but you may encounter the known issues. Please refer to http://wiki.netbeans.org/wiki/view/UMLMacIssues for details
	Visual Web JSF Post Release Sa	Web & Java EE	ß		on these issues.
	XTest Module	Testing Tools	ß	•	
Install	1 plugin selected, 15MB				

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**

Projects	€ ×	Files	Services
🗢 🔺 V	olicacionPru	iepa	· · · · · ·
▶ 🗟	Configurat	tion Files	
Þ 🖬	Server Res	sources	
Þ 🗖	Java EE Mo	odules	333
🗢 🚫 Ap	licacionPru	ueba-ejb	
- ⊂ 🖏	Enterprise	Beans	
▶ \$	👌 BeansD	ePruebaSB	
▶ 🗟	Configurat	tion Files	
▶ 🖬	Server Res	sources	
- ▽ 📬	Source Pa	ckages	
	BeansD	ePrueba	•

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**

			Reverse Engineer	×			
5	elected No	des: Ap	licacionPrueba-ejb				
J	ava Project	<u>S</u> ource Pa	ckage Folders				
	Reverse	Packa	Package Folder Label				
	✓	java	Source Packages				
	✓	test	Test Packages				
	🔵 <u>U</u> se Exis	ting UML F	Project				
	Target Pro	ject:					
	_						
(i <u>C</u> reate N	lew UML P	roject				
	Project <u>N</u> a	ame:	AplicacionPrueba-ejb-Model				
	Dunia at La		(harma (anniah ao (h) an Dao isana (hali a a isan Daniah a	Deserver			
	Project Lo	cation:	//nome/pruebas/NetBeansProjects/ApricacionPrueba	Browse			
	Project Fo	ol <u>d</u> er:	/home/pruebas/NetBeansProjects/AplicacionPrueba/Ap				
			OK Can	cel <u>H</u> elp			

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

	Create New D	iagram 🕱
Steps	Create New Diagra	m
1. Create New Diagram	Diagram <u>T</u> ype:	Activity Diagram Class Diagram Collaboration Diagram Component Diagram Deployment Diagram Sequence Diagram State Diagram Use Case Diagram
	<u>D</u> iagram Name: <u>N</u> amespace:	BeansDePrueba AplicacionPrueba-ejb-Model
		< Back Next > Finish Cancel Help

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

5	itart Pag	e x	В	eans	sDePr	ueba *	×				▶	-
) 🖑	Q	٩			2	20			5	1	L	÷
			Be	ansD {} Beans	< <in PePrue From Be As Op DePruel</in 	terface> ebaRen ans DePri mbutes entions haRemote	> ooteH eba}	e()				
				Bean	< <in ISDeP From Be An</in 	terface > ruebaß ans DePri	e mot eba j	te				
					- Op	ender son a						
	public voi public Str.	d Metac ing Met	loI(int µ odo2(St	iarami ring pa	etro1, S arametr	tring para oI, int pa	metro rameti	2, int pa ro2)	rametro	3)		
	public voi public Str.	d Metoc ing Met	loI(int µ odo2(St	iarame ring pa	etrol, S trametr	ePrueb	metra rameti aBear	2, int pa ro2)	rametro	3)		
	public voi public Str.	d Metac ing Met	loI(int µ odo2(St	Be	ansDe From B	ePrueb eansDePr	aBear	2, int pa ro2)	rametro	3)		
	public voi public Stra	d Metac ing Met essionC	lo2(int µ odo2(St	Be (In g parameters) (I (In g parameters) (In g	ansD(From B	ePrueb eans DePr	aBear ueba)	2, int pa ro2)	rametro	3)		
	public voi public Str. private S- public voi public voi	d Metoc ing Met essionC id setSe id setSe	ontext c ssionCc tivate(Be (1 ontext)	ansDo From Bo Sessio	ePrueba ePrueba eans DePrueba eans DePrueba eans DePrueba estitutes	aConte	2, int pa ra2)	rametro	3)		
	public voi public Str	d Metoc ing Met essionC id setSe id ejbAc id ejbRe id ejbRe	ontext c ssionCc tivate(ssivate(eate()	Be (1 ontext))	ansD From B A Sessio	ePrueb: eans DePr mibutes	aConte	2, int pa ro2)	rametro	3)		



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

Ahora, $\dot{z} que$ 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**

\bigcirc	Reverse Engineer 🛛 🗙
S <u>e</u> lected Nodes:	BeansPrueba2
<u>●</u> Use Existing UM	L Project
Target Project:	🖶 AplicacionPrueba-ejb-Model 💽 💌
Create New UML	Project
Project <u>N</u> ame:	AplicacionPrueba-ejb-Model 1
Project <u>L</u> ocation:	/home/pruebas/NetBeansProjects/AplicacionPrueba
Project Fol <u>d</u> er:	home/pruebas/NetBeansProjects/AplicacionPrueba/Aplicacio
	OK Cancel <u>H</u> elp

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

Projects 🜗	× Files	Services	UML Design C			
マ 🖶 AplicacionPrueba-ejb-Model ▲						
🗢 🎒 М	odel					
▶ 📄	BeansDePrueba					
D 🔁	BeansDePrueba					
Þ 🔒	BeansPrueba2					
D 🔁	java					
D 🔁	java×					
۵ (int					
Image:	String					
Image:	void					
🗢 💽 Di	agrams					
N 📑	BeansDePrueba		•			

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**

	New Project	×
Steps	Choose Project	
1. Choose Project	<u>C</u> ategories:	Projects:
2	📑 Java	Platform-Independent Model
	🛅 Web	🖶 Java-Platform Model
	🛅 Enterprise	Reverse Engineered Java-Platform
	🗂 UML	
	SOA	
	🛅 NetBeans Modules	
	👂 🛅 Samples	
	Description:	
	with empty containers for your	nt modeling project in the system
	and populate your model, only	/ UML modeling (business) rules are
	applied to the model. No code	is generated, and no
	< <u>B</u> ack	Next > Einish Cancel Help

Ahora le damos un nombre que en este caso sera PruebasDiagramas

\bigcirc	New Platform-Independent Model 🛛 🗙
Steps	Name and Location
1. Choose Project 2. Name and Location	Project <u>N</u> ame: PruebasDiagramas
	Project Location: /home/pruebas/NetBeansProj. Browse Project Folder: /home/pruebas/NetBeansProj.
	< Back Next > Finish Cancel Help

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**

	Create New Dia	gram 🔀
Steps	Create New Diagra	m
1. Create New Diagram	Diagram Type:	A saluita : Dia mana
	5.114.111 Jpc.	Class Diagram
		Collaboration Diagram
		Deployment Diagram
		Sequence Diagram
		🔁 Use Case Diagram
	<u>D</u> iagram Name:	Diagrama 1
	<u>N</u> amespace:	PruebasDiagramas 💌
	<	Back Next > Finish Cancel Help

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

 ${\bf a})$ 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

Palette	⊪×
Basic	•
Class	
😔 Interface	
🗁 Package	
Collaboration Lifeline	
Enumeration	
i Node	

 ${\rm Y}$ 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 Clasel



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**

	New Enterprise Application
Steps	Name and Location
1. Choose Project 2. Name and Location	Project Name: CodigosDiagramas Project Location: /home/pruebas/NetBeansProjects Browse
	Project <u>F</u> older: //home/pruebas/NetBeansProjects/CodigosDiagrama
	Server: [JBoss Application Server 🔍 Add]
	Java EE Version: J2EE 1.4
	Create EJB Module: CodigosDiagramas-ejb
	Create Web Application Module: CodigosDiagramas-war
	Create Application Client Module: CodigosDiagramas-app-client
	Main <u>C</u> lass: codigosdiagramas.Main
	Recommendation: Source Level 1.4 should be used in J2EE 1.4 projects.
	< <u>B</u> ack Next > Finish Cancel Help

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**

🗢 💾 PruebasDiagramas					
▽ 📥	1odel				
	Diagramal				
	Clase 1				
	Clase2				
Þ 💷	int				
Þ 💷	String				
Þ 💷	Stringint				
Þ 💷	void				

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.



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 <u>http://www.postgresql.org/download/</u> o directamente entramos a <u>http://jdbc.postgresql.org/</u>



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

0										Download	l - Mozi	lla Firefox
Archivo	<u>E</u> ditar	Ver	<u>H</u> istorial	Marcado	res	Herram	ien <u>t</u> as	Ayuda				
•	> •	C	\bigotimes	1	http)://jdbc.p	ostgreso	ql.org/do	wnload.ht	ml		
Supported Versions												
				Ve	rsio	n		IDBC	2	ID	BC 2	FF
				8.3 Bu	ld 6(03	8.3-60	3 JDB	C 2	8.3-603	DBC	2 <u>EE</u>
				8.2 Bu	ld 5(08	8.2-50	8 JDB	C 2	8.2-508	DBC	2EE
				8.1 Bu	ld 43	12	8.1-41	2 JDB	C 2	8.1-412	DBC	2 <u>EE</u>
				8.0 Bu	ld 32	22	8.0-32	2 JDB	<u>C 2</u>	8.0-322	DBC	<u>2EE</u>

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**

Pro	jects		Files		Services	40	×		
⊳		Databases							
⊳	Q	Web Services							
⊳	۹	Enterprise	Enterprise Beans (2.x)						
⊳		Servers							

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

\bigcirc	New JDBC D	niver			
<u>D</u> river File(s):			<u>A</u> dd		
	\bigcirc	Select I	iver 🔀		
Driver <u>C</u> lass:	Buscaren: 📑 Desk	top			
<u>N</u> ame:	Clases				
	jboss-4.0.4.GA				
	📑 pantallazos				
	📑 postgrepantallazo	5			
	postgresql-8.0-32		New JDBC Driver		(X)
	Nombre de archivo: Archivos de tipo:	<u>D</u> river File(s):	/home/pruebas/Desktop/postgresql-8.0-322.jdbc2ee.jar		<u>A</u> dd
				B	emove
		Driver <u>C</u> lass:	org.postgresql.Driver	•	Find
		<u>N</u> ame:	PostgreSQL		
				ОК	Cancel

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 N**EW 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

\bigcirc	New Database Connection
Basic setting	Advanced
<u>N</u> ame:	PostgreSQL
<u>D</u> river:	org.postgresql.Driver
D <u>a</u> tabase URL:	jdbc:postgresql://localhost:5432/test
<u>U</u> ser Name:	postgres
<u>P</u> assword:	(XXXX
	Remember password (see help for information on security risks)
Connection es	tablished.
	OK Cancel <u>H</u> elp

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**

\bigcirc	Hew Database Connection	X
Basic setting Ad	vanced	
Select a databas	e schema to use.	
<u>S</u> elect schema:	public	
Connection esta	blished.	
		OK Cancel <u>H</u> elp

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.

Cre	ate Table	X
Table name: Untitled	<u>O</u> wner: public 💌	
Key Index Null Unique Column name	Data type Size	A <u>d</u> d column
	NUMERIC 0	Remove
	Image: A start of the start	
		K Cancel

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

C	Create Table								
Table name: tabla_prueba							public 💌		
	Key	Index	Null	Unique	Column name	Data type	Size	A <u>d</u> d column	
	•			1	id_persona	INT4	0	Remove	
					nombre_persona	VARCHAR	30		
					apellidos	VARCHAR	45		
			•		correo	VARCHAR	50		
			1		edad	INT2	0		
	•						Þ		
								OK Cancel	

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

Create Table								
Table n	ame: dat	tos_tabla	a_prueba		<u>0</u> wner:	public 💌		
Key	Index	Null	Unique	Column name	Data type	Size	A <u>d</u> d column	
1				iddato	INT4	0	Remove	
				idpersona	INT4	0	<u></u>	
				direccion	VARCHAR	60		
				nombre_papa	VARCHAR	60		
				nombre_mama	VARCHAR	60		
							0K Cancel	

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

▽ 🖳 jdbc:postgresql://localhost:5432/test [postgres on p					
▽ 📋 Tables					
datos_tabla_prueba					
tabla_prueba					

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

SQL Command 1 ×			
Connection: jdbc:postgresql://localhost:5432/test [postgres on public] 🗨 🛛 🚯	🐺 🛛 🚱 🕶	- Q 💀 🖓 层	
		Output	

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	Ξ	<pre>ejecuta</pre>	el	codigo.
----------	---	--------------------	----	---------

	SQL Command 1 x						
Cg	nnection: jdbc:postgresql://localhost:5432/test [postgres on public] 🕤 🚯 🖙 🚱 🖉 🚛 - 🚚 - 🕴 🏹 😓 🧦	· [=					
ALTER TABLE datos_tabla_prueba ADD CONSTRAINT id_persona FOREIGN KEY (id_persona) REFERENCES tabla_prueba (id_persona)							
		-8-					

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**

▽ 🚞 Tables	
🗢 🏢 datos_tabla_prueba	
iddato iddato	
idpersona	
ireccion direccion	
nombre_papa	
📑 nombre_mama	
Indexes	
▽ 🚞 Foreign keys	
🕨 📴 id_persona	
tabla_prueba	

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



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

SQL Command 2 x								
Connection: jdbc:postgresql://localhost:5432/test [postgres on public] 🕤 🚯 🐺 🔯 🌉 - 🚚 - 🍳 🔧 🖓								
select * from public.tabla_prueba								
1:1 INS								
id_persona nombre_persona ape	llidos	reo	edad					
1 elnombre apel	llidos corre	eo@dominio.com	25					

SQL Command 1 ×						••			
Connection: jdbc:postgresql://localhost:5432/test [postgres	on public] 💌	🚯 🐺	🚱 🔜 -		🔍 🖓	- 7 (=			
select * from public.datos_tabla_prueba									
1:40 INS									
iddato id_persona direccion	nombre_papa	nombre_mama)						
1 1 mi casa con su numero	mi papa	mi mama							

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.

Contraction View	vs cedures
	Create View View name: vista_prueba SQL expression to create view: select nombre_persona, apellidos, correo from tabla_prueba OK Cancel
	Views Vista_prueba

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.

🥠 jdbc:postgresql://loca	lhost:5432/test	postgres on public] 🗙			•			
datos_ta datos_ta dato dipersa direcci direcci	abla_prueba nna on e_papa	[Table]						
Column	Alias	Table	Output	Sort Type	Sort Order	Criteri		
iddato		public.datos_tabla_pru	eba 🛛	1				
idpersona		public.datos tabla pru	eba 🚺	1				
direccion		public.datos_tabla_pru	eba 🛛	1				
nombre_papa		public.datos_tabla_pru	eba 💽	1				
nombre_mama		public.datos_tabla_pru	eba 💽	1				
id_persona		public.tabla_prueba		2				
nombre_persona		public.tabla_prueba		1				
•	1			-				
SELECT ALL publi	c datos ta	abla prueba idd	ato			1		
public.datos tabla prueba.idpersona.								
public.datos tabla prueba.direccion.								
public.datos_tabla_prueba.nombre_papa,								
public.datos_tabla_prueba.nombre_mama,								
public.tabla_prueba.id_persona,								
public table pruche persona								

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.

🔋 SQL Comn	and 1 ×								
Connection: jdbc:postgresql://localhost:5432/test [postgres on public] 🕤 🚯 🐺 🞯 📰 - 🜉 - 🍳 🔧 🖓									
SELECT	SELECT ALL public.datos_tabla_prueba.id_persona, public.datos_tabla_prueba.direccion, public.datos_tabla_prueba.nombre_papa, public.datos_tabla_prueba.nombre_mama, public.tabla_prueba.nombre_persona, public.tabla_prueba.apellidos, public.tabla_prueba.id_persona FROM public.datos_tabla_prueba INNER JOIN public_tabla_prueba ON public_datos_tabla_prueba_id_persona								
9:114 INS									
[id_persona	[id_persona	direccion	nombre_papa	nombre_mama	nombre_persona	apellidos			
1	1	mi casa con su numero	mipapa	mi mama	elnombre	apellidos			
			•				,		

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();

Funcion 2:

-- Function: plpgsql_validator(oid)

-- DROP FUNCTION plpgsql_validator(oid);

Funcion 3:

- -- Language: plpgsql
- -- DROP LANGUAGE plpgsql;

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