Tecnicatura Superior en Análisis, Desarrollo y Programación de Aplicaciones

Práctica Profesional

"Desarrollo de Aplicación J2EE utilizando NetBeans 7.1.1, Glassfish 3.1.2 y Firebird Database 2.5"
Versión 1.0 Abril 2012

Lic. Guillermo R. Cherencio

Índice

Introducción

El objetivo de este apunte es orientar al alumno de Práctica Profesional en el desarrollo de una aplicación J2EE de n capas, la cual utilizará tecnologías tales como: JPA 2.0, JSF 2.0, la librería PrimeFaces 3.1.1, Glassfish 3.1.2, NetBeans 7.1.1, Firebird SQL Database 2.5, etc.. Todo el proyecto se desarrolló sobre Linux, no obstante, los productos son multiplataforma y open-source por lo tanto, también podría llevarse a cabo en plataformas windows.

El desarrollo J2EE tiene cierta complejidad a pesar de que las últimas especificaciones han mejorado muchísimo en este sentido, facilitando el desarrollo.

Se pretende que el alumno reutilice conocimientos de la asignatura base de datos y la primera unidad de Práctica Profesional en cuestiones avanzadas sobre la programación de bases de datos, para ello, se ha integrado este desarrollo con Firebird SQL Database 2.5.

El apunte es bien guiado para que el alumno vaya realizando la actividad paso a paso, en forma pausada, razonando las tareas que se realizan para luego automatizarlas en la creación de proyectos de mayor alcance en vistas a la presentación de su trabajo final para esta asignatura.

Abreviaturas

FR: FlameRobin
FB: Firebird SQL Database Server 2.5
GF: servidor Glassfish 3.1.2
NB: IDE NetBeans 7.1.1
PF: librería PrimeFaces 3.1.1

Algunas Convenciones

El contenido de algunos archivos se encuentra indicado en `este tipo de letra y tamaño`. La palabra BACKUP se refiere a realizar una copia del directorio del proyecto, así como también de la carpeta domain dentro de la instalación de GF.

Requisitos Previos

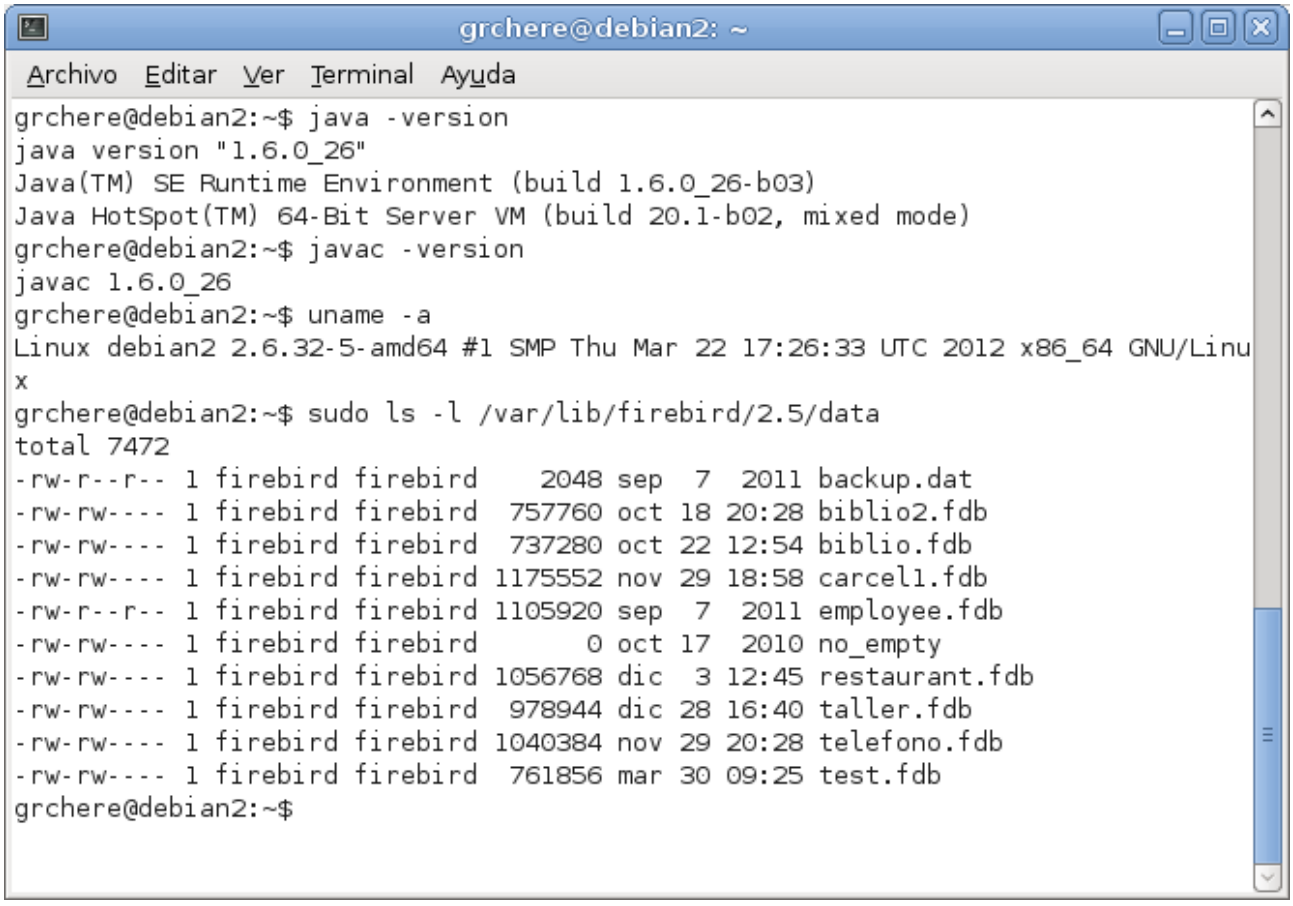
Tener instalado:

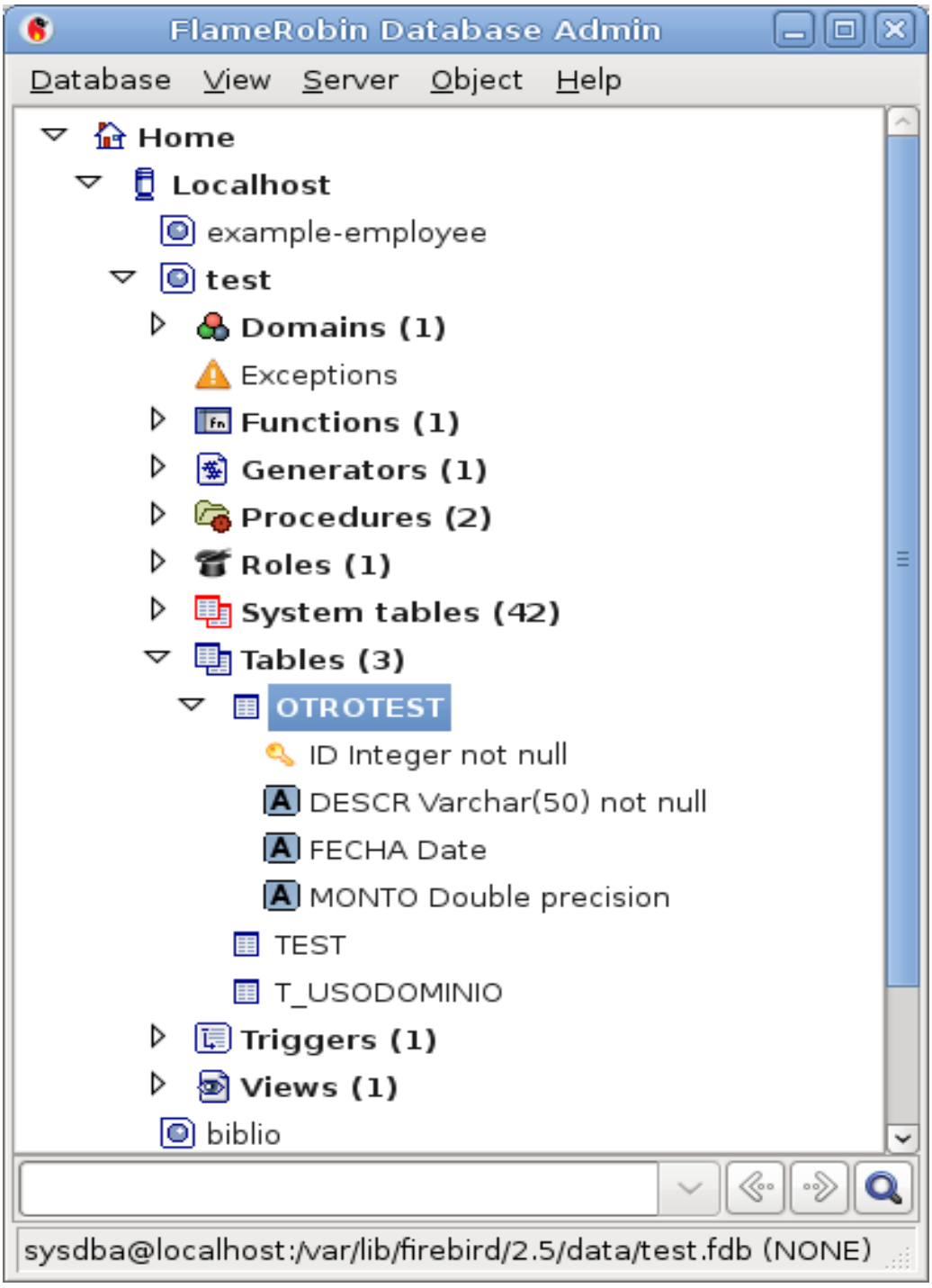
-j2sdk versión 6

-Servidor de Base de Datos Firebird SQL 2.5

-Cliente para Base de Datos Firebird FlameRobin

-Tener creada una base de dato Firebird, sobre la cual trabajar, con -al menos- una tabla con un campo de tipo fecha.




FlameRobin Database Admin
Database View Server Object Help
Home
Localhost
example-employee
test
Domains (1)
Exceptions
Functions (1)
Generators (1)
Procedures (2)
Roles (1)
System tables (42)
Tables (3)
OTROTEST
ID Integer not null
DESCR Varchar(50) not null
FECHA Date
MONTO Double precision
TEST
T_USODOMINIO
Triggers (1)
Views (1)
biblio
sysdba@localhost:/var/lib/firebird/2.5/data/test.fdb (NONE)

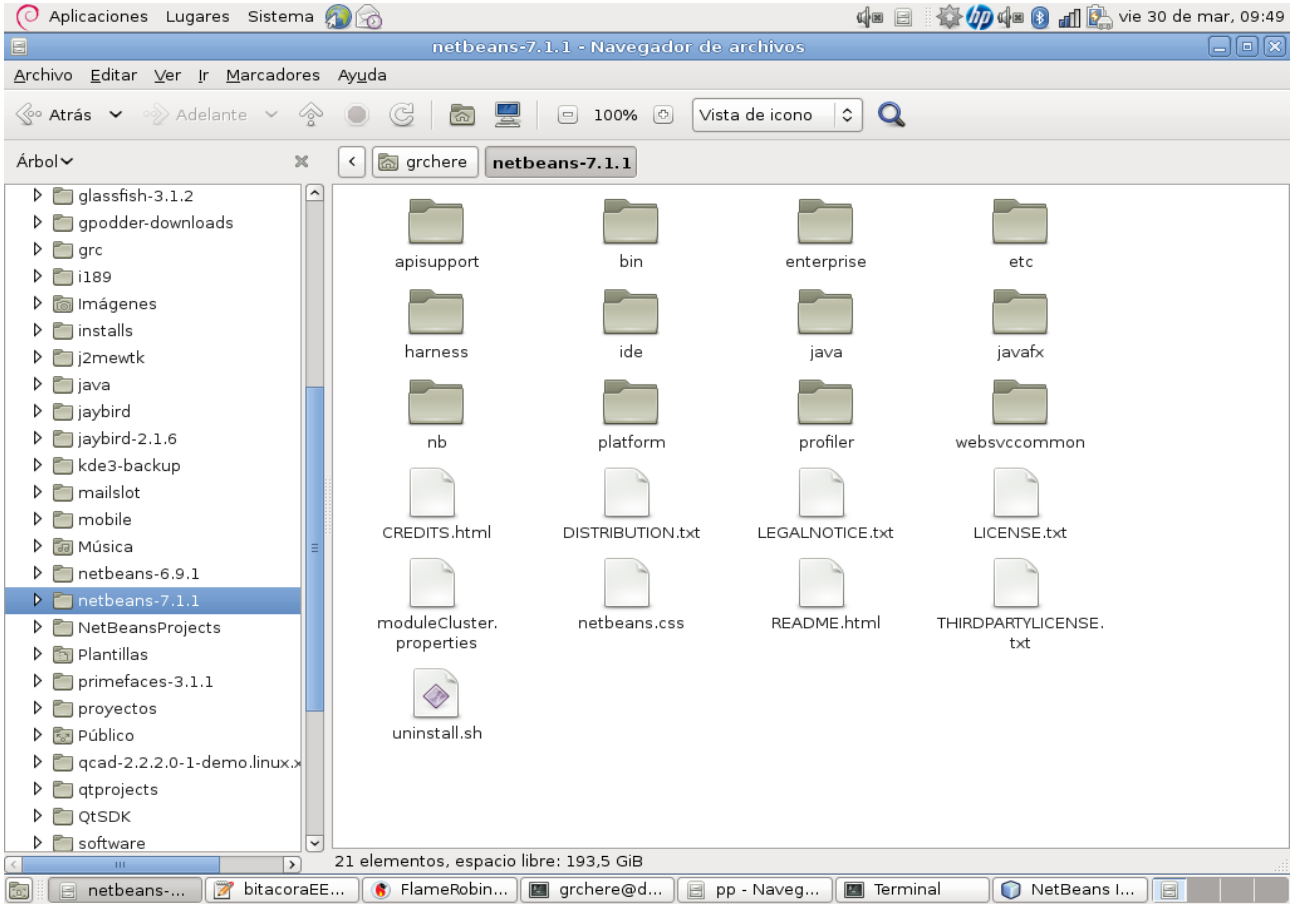
Instalación de NetBeans 7.1.1

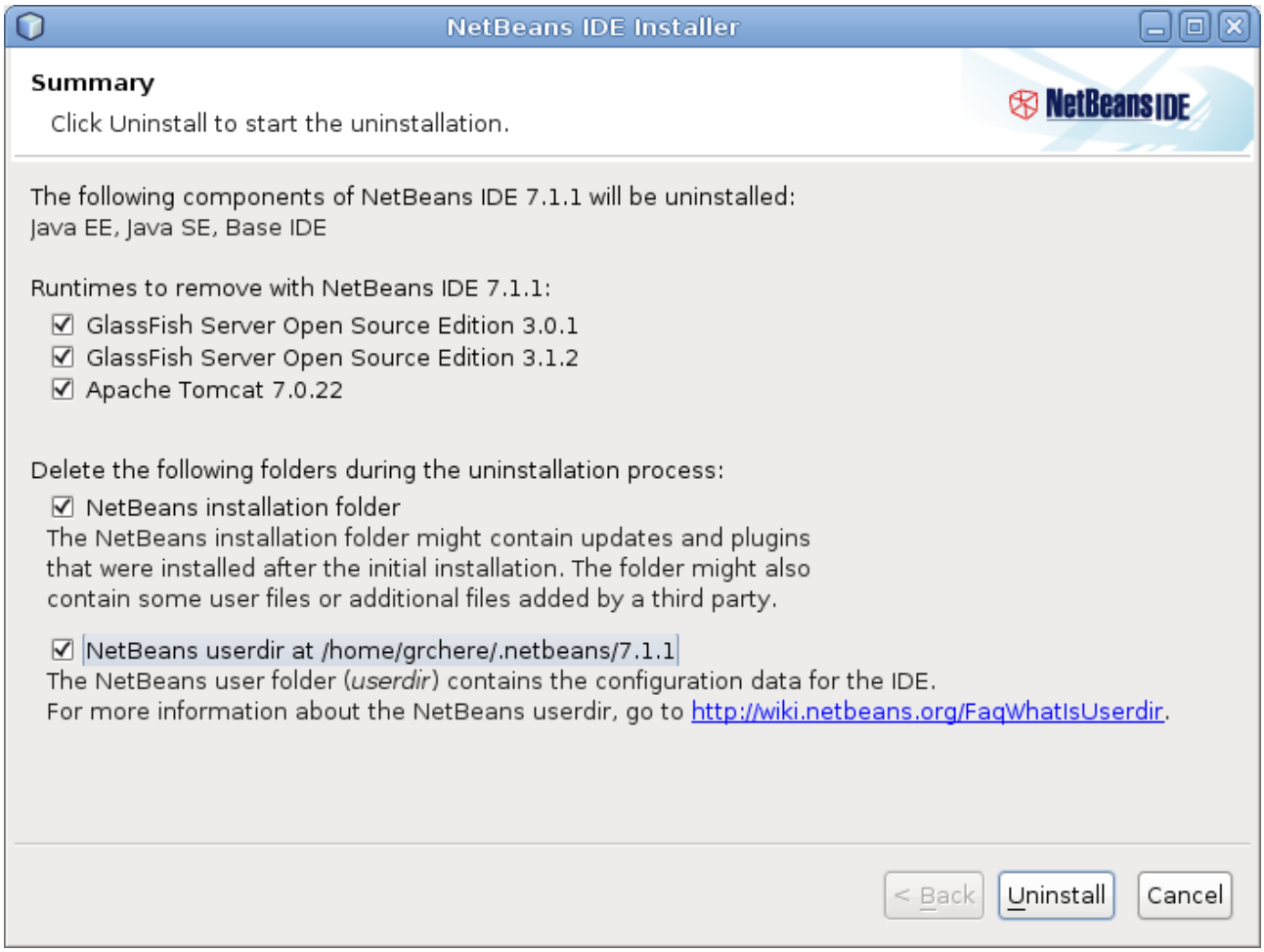
Descargar NetBeans 7.1.1 Java EE (incluye servidores glassfish y tomcat), total 166MB , desde http://netbeans.org/downloads/index.html en carpeta de descargas del navegador se creará el archivo netbeans-7.1.1-ml-javaee-linux.sh el cual es un script linux a ejecutar para comenzar la instalación

Antes de comenzar la instalación, deberá otorgar permiso de ejecución al instalador:

$ chmod +x netbeans-7.1.1-ml-javaee-linux.sh

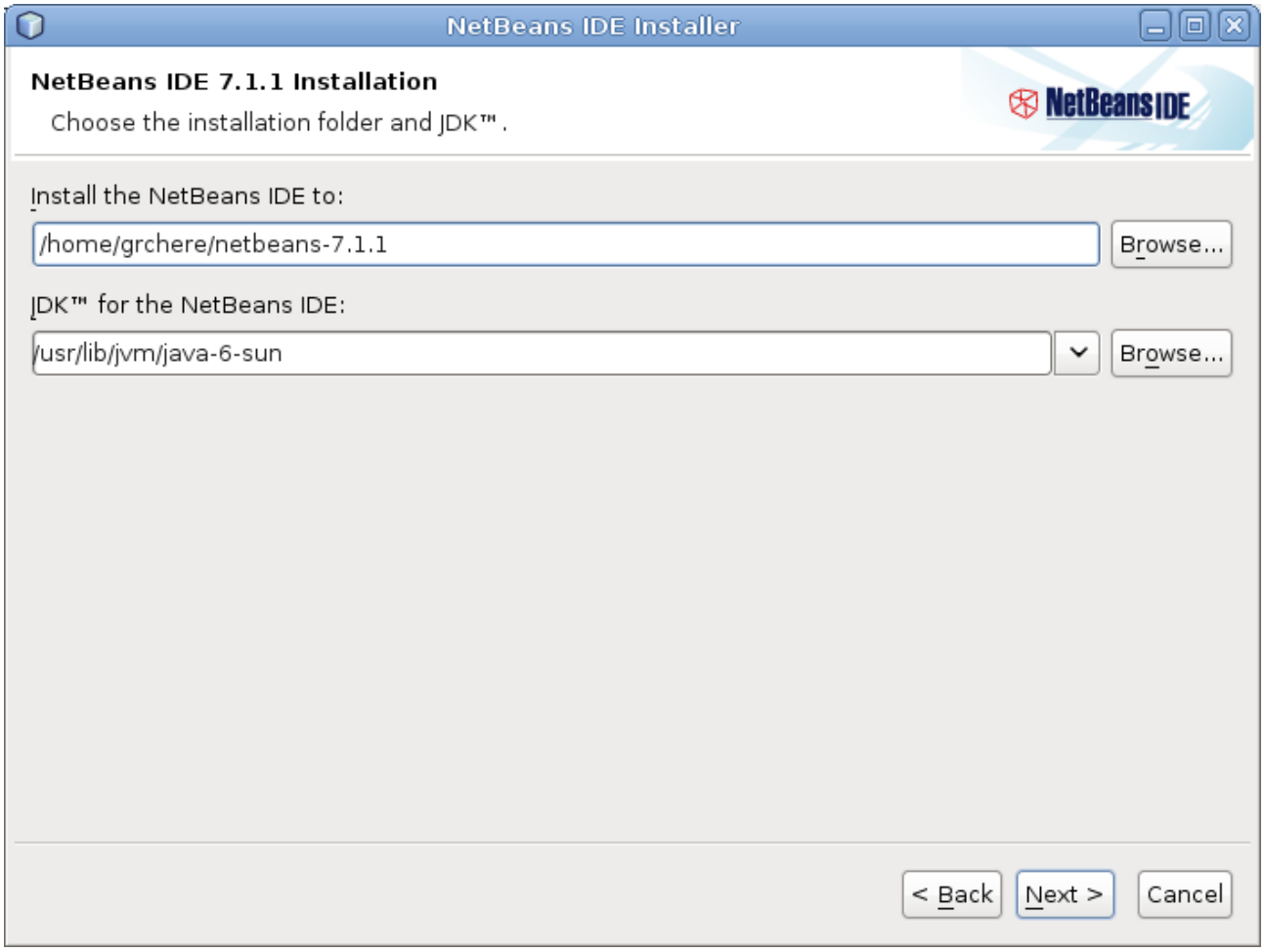
Si ya tiene instalada esta versión de NetBeans en su máquina, puede desinstalarla si lo desea, ejecutando el script uninstall.sh :


Aplicaciones Lugares Sistema
vie 30 de mar, 09:49
netbeans-7.1.1 - Navegador de archivos
Archivo Editar Ver Ir Marcadores Ayuda
Atrás Adelante 100% Vista de icono
Árbol
grchere netbeans-7.1.1
glassfish-3.1.2
gpodder-downloads
grc
i189
Imágenes
installs
j2mewtk
java
jaybird
jaybird-2.1.6
kde3-backup
mailslot
mobile
Música
netbeans-6.9.1
netbeans-7.1.1
NetBeansProjects
Plantillas
primefaces-3.1.1
proyectos
Público
qcad-2.2.2.0-1-demo.linux.x
qtprojects
QtSDK
software
apisupport
bin
enterprise
etc
harness
ide
java
javafx
nb
platform
profiler
websvccommon
CREDITS.html
DISTRIBUTION.txt
LEGALNOTICE.txt
LICENSE.txt
moduleCluster.properties
netbeans.css
README.html
THIRDPARTYLICENSE.txt
uninstall.sh
21 elementos, espacio libre: 193,5 GiB
netbeans-...
bitacoraEE...
FlameRobin...
grchere@d...
pp - Naveg...
Terminal
NetBeans I...

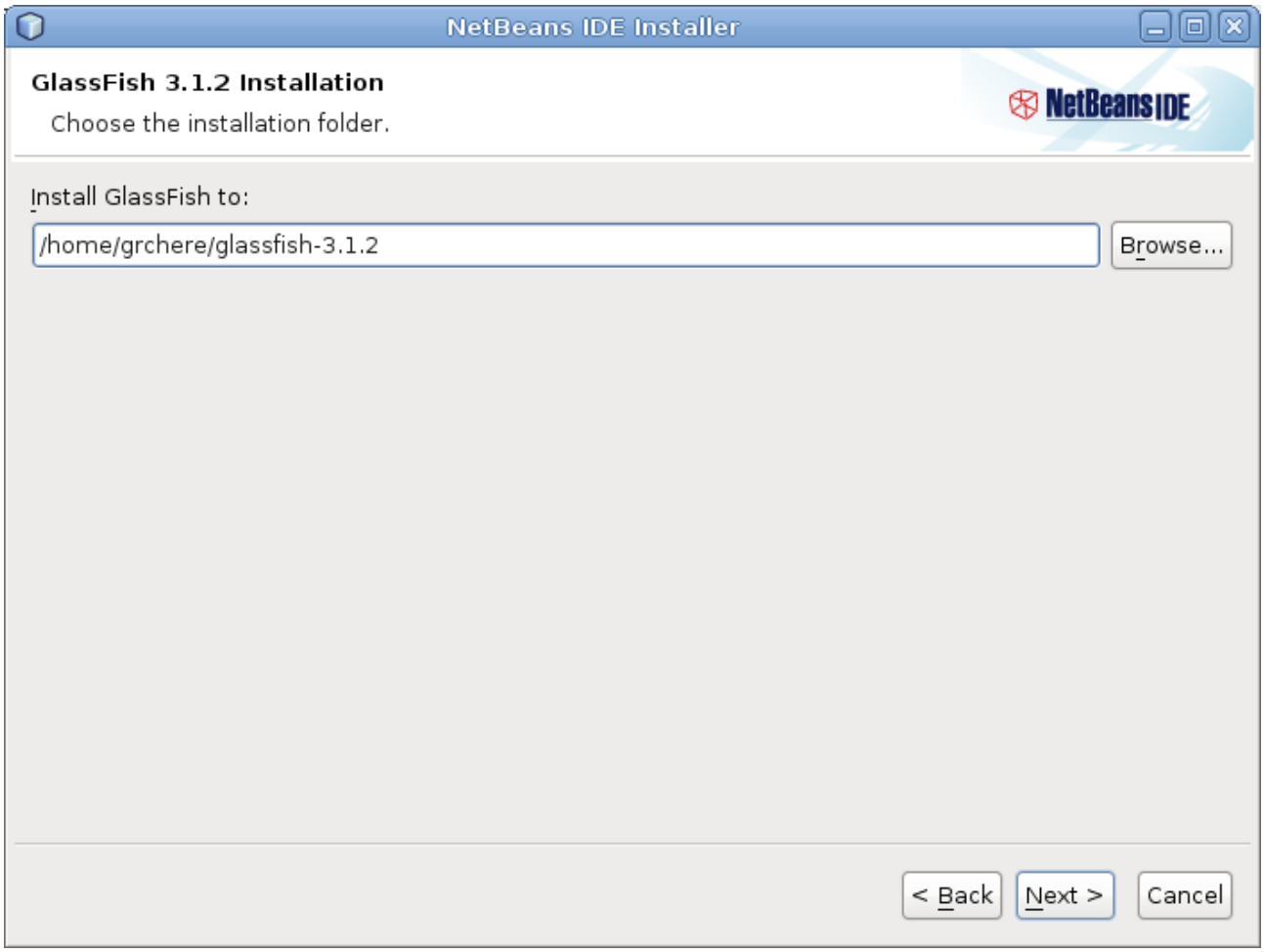


Luego puede proceder a instalar NetBeans 7.1.1 ejecutando el script netbeans-7.1.1-ml-javaee-linux.sh

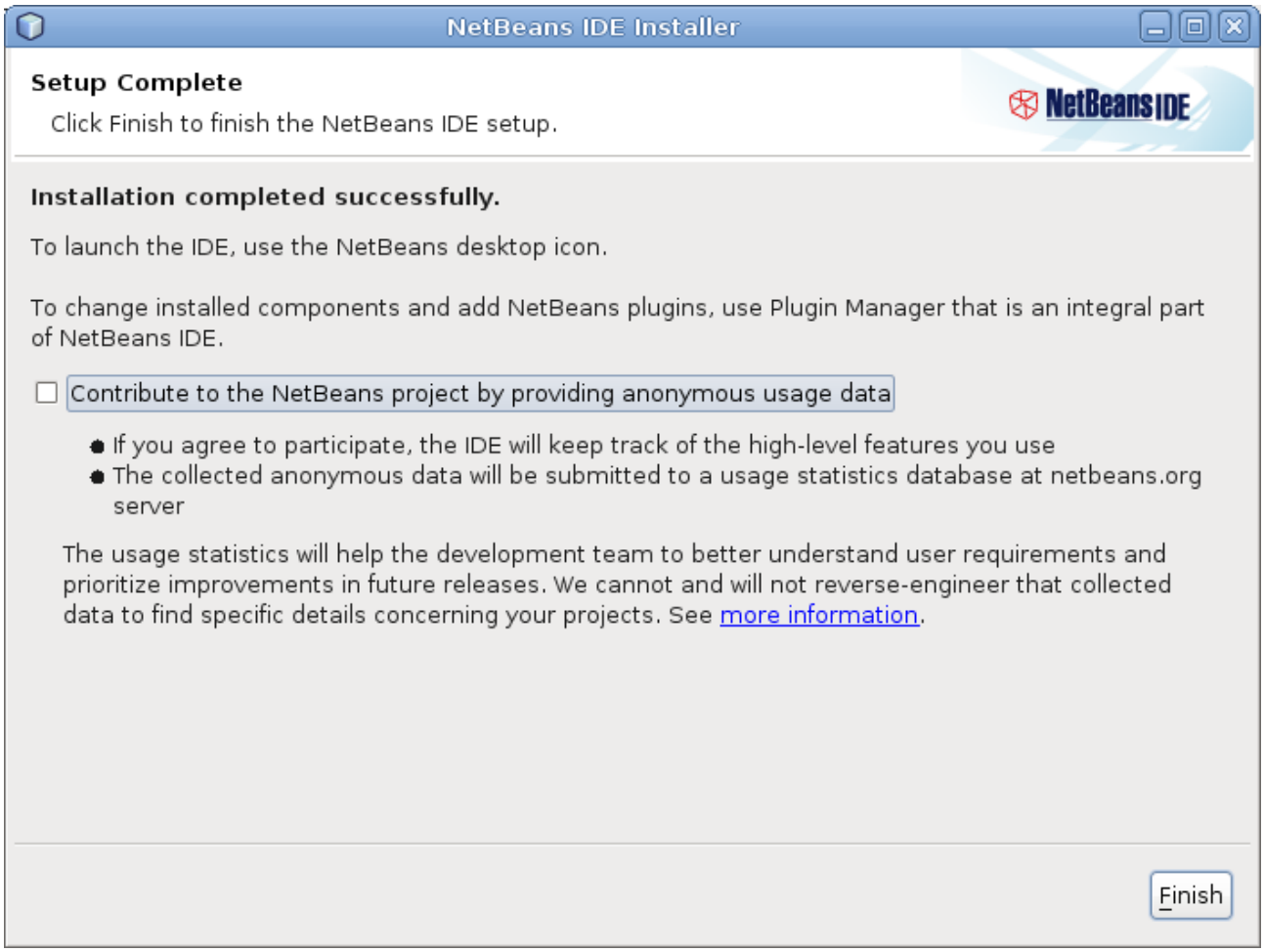
Si lo desea puede instalar ambos servidores, en nuestro caso, solo vamos a utilizar Glassfish 3.1.2.



Indicamos los directorios de instalación.



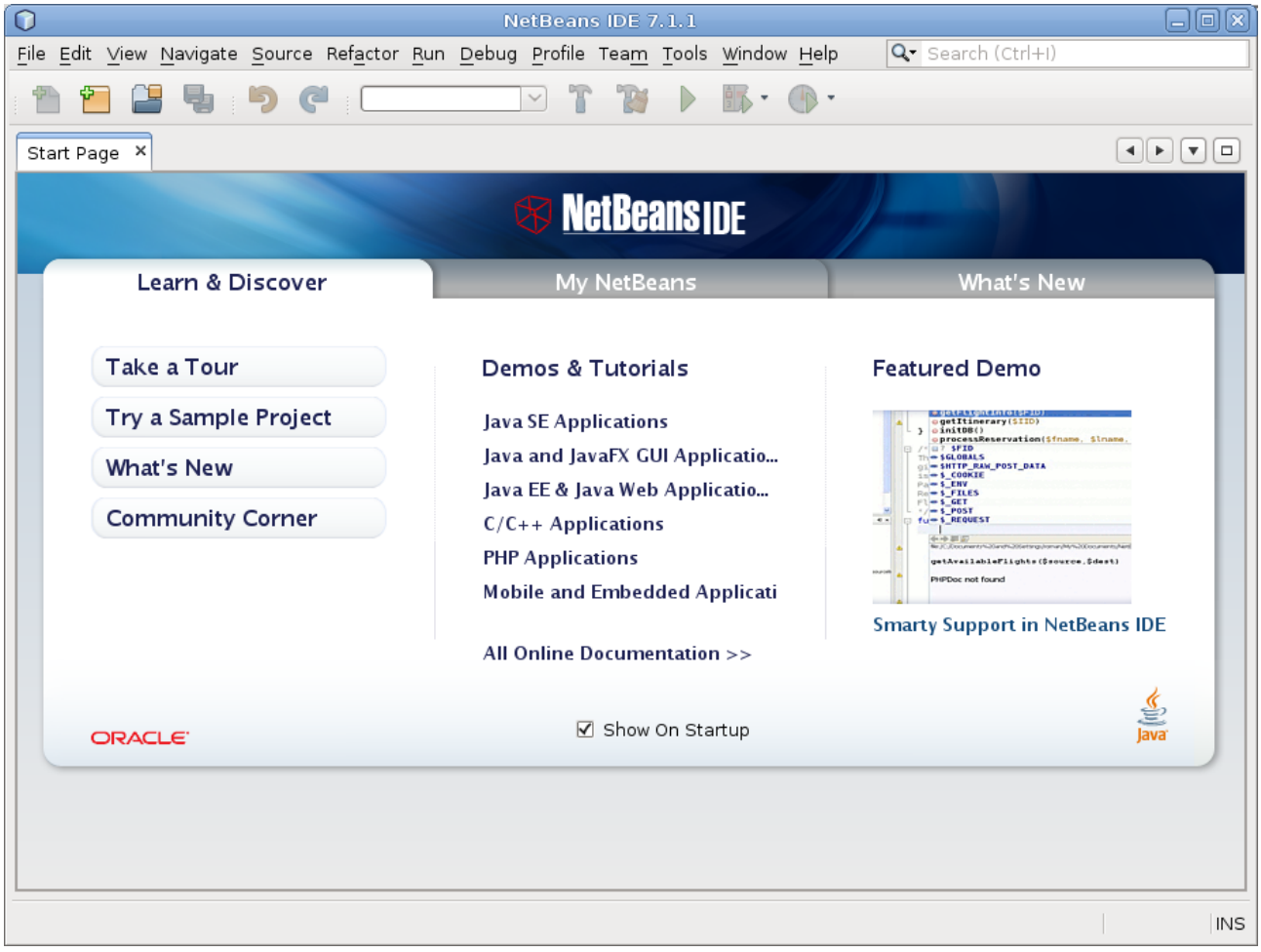
El directorio para el servidor Glassfish 3.1.2.



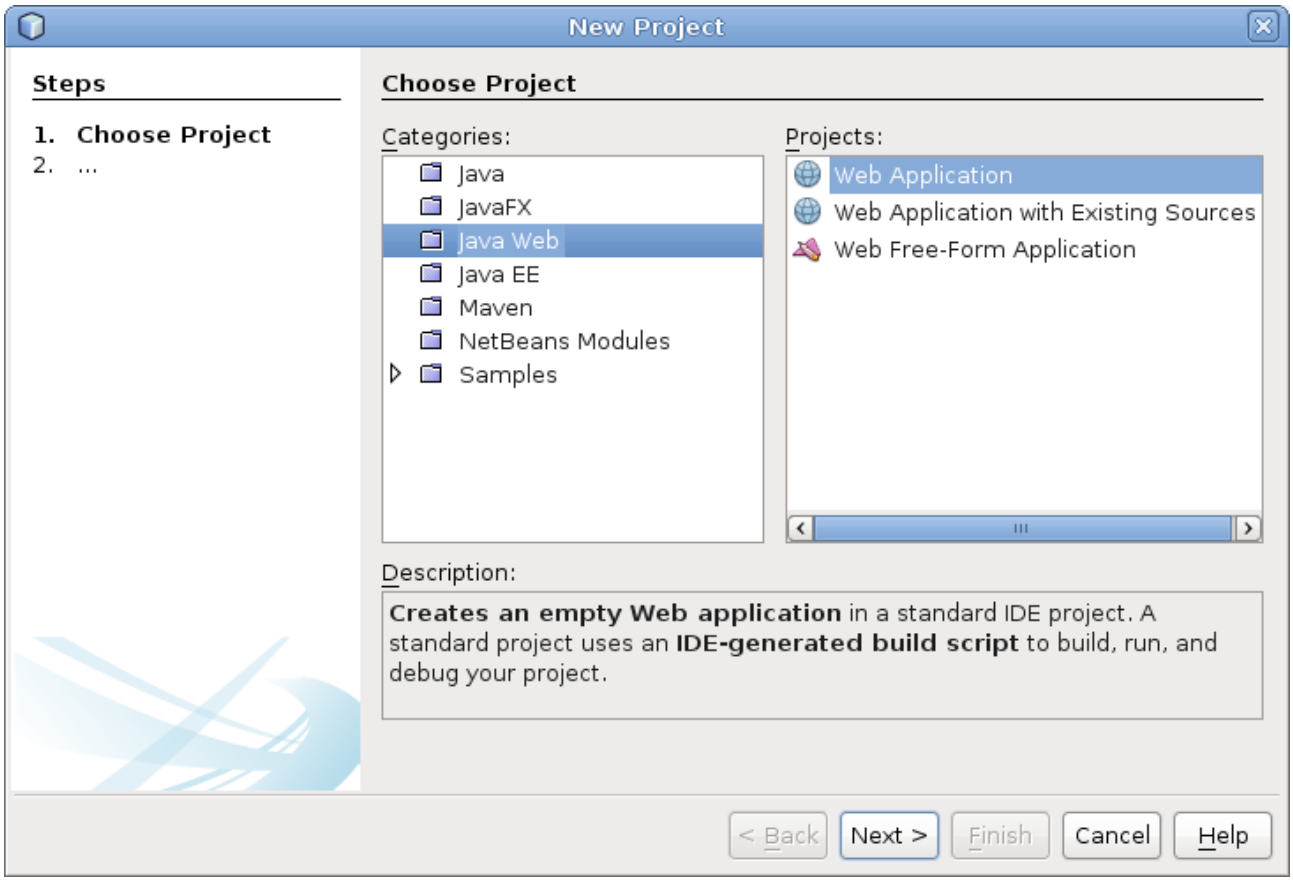
Y finalizamos la instalación.

Creación de Proyecto

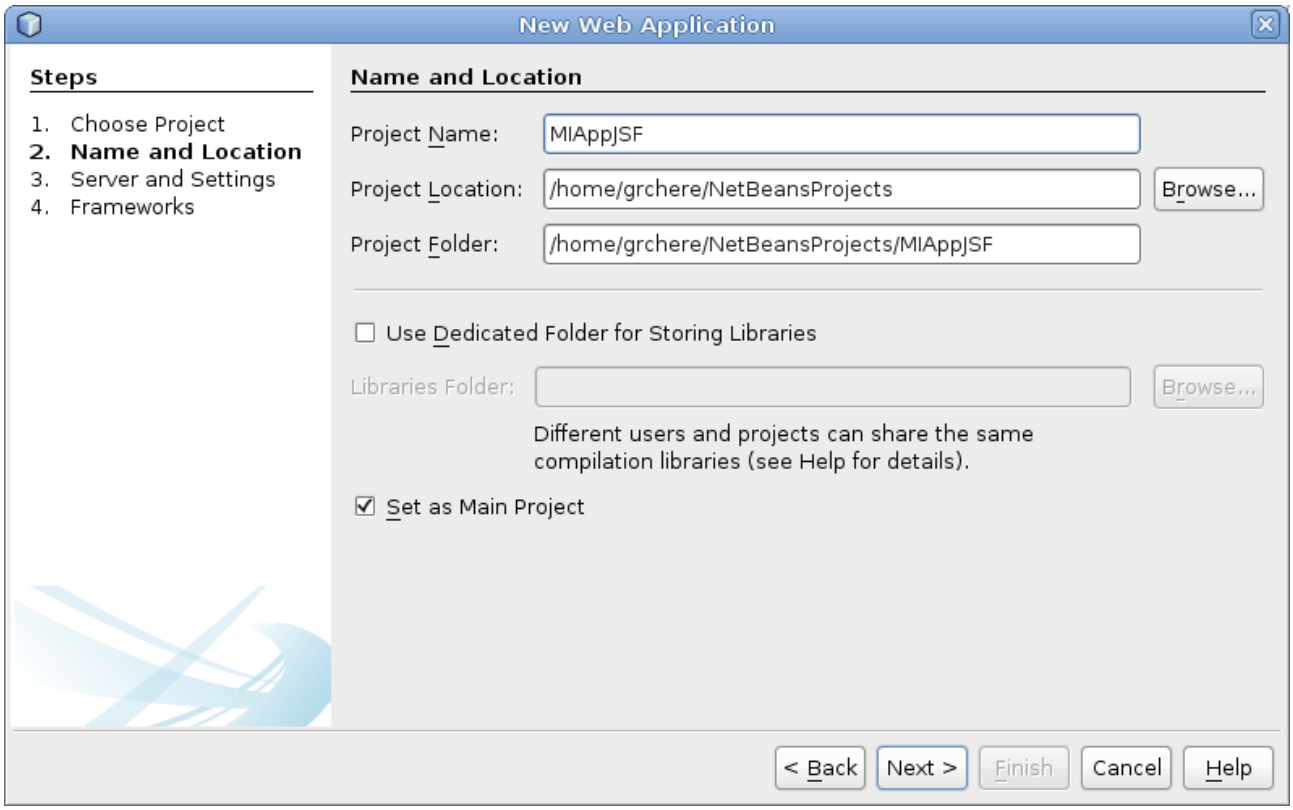
Ejecutamos NetBeans 7.1.1:



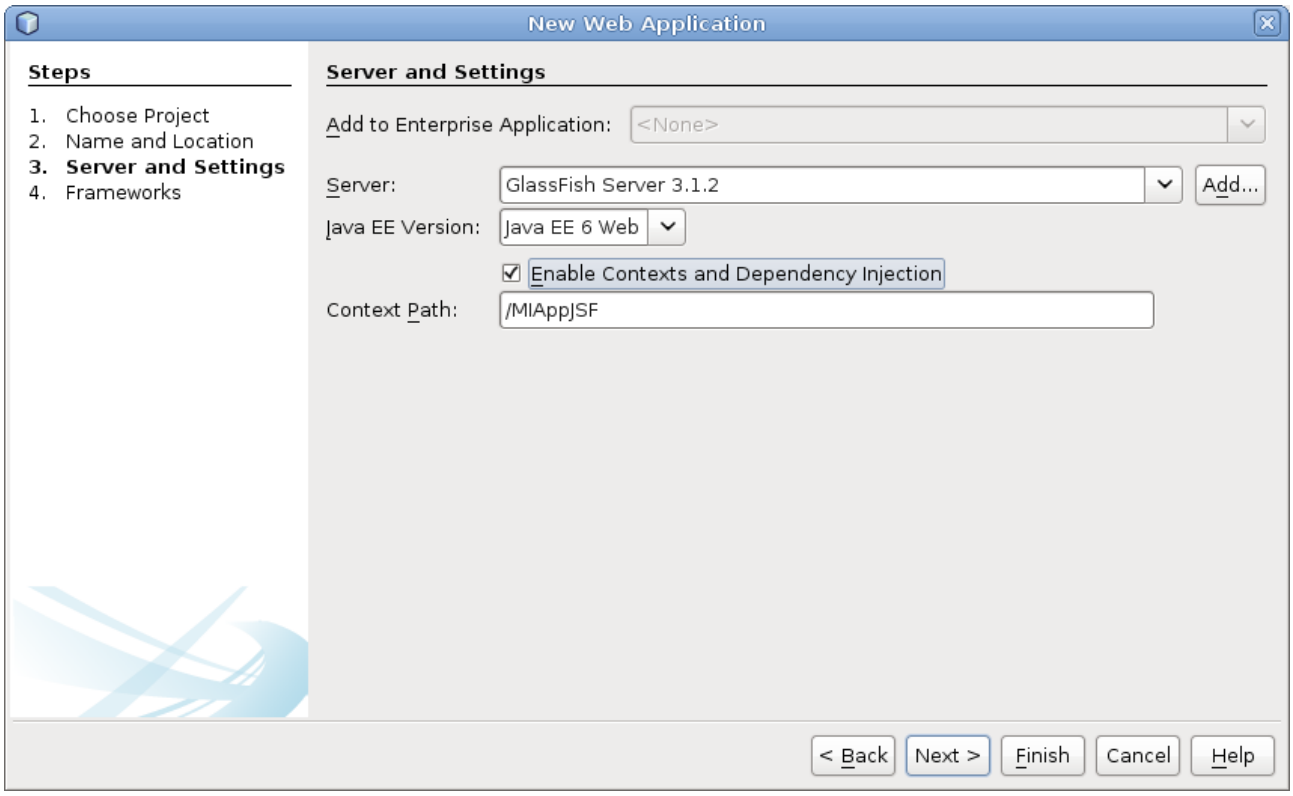
Creamos proyecto nuevo (presionando el segundo botón -de izquierda a derecha- de la toolbar NB) de tipo Web:



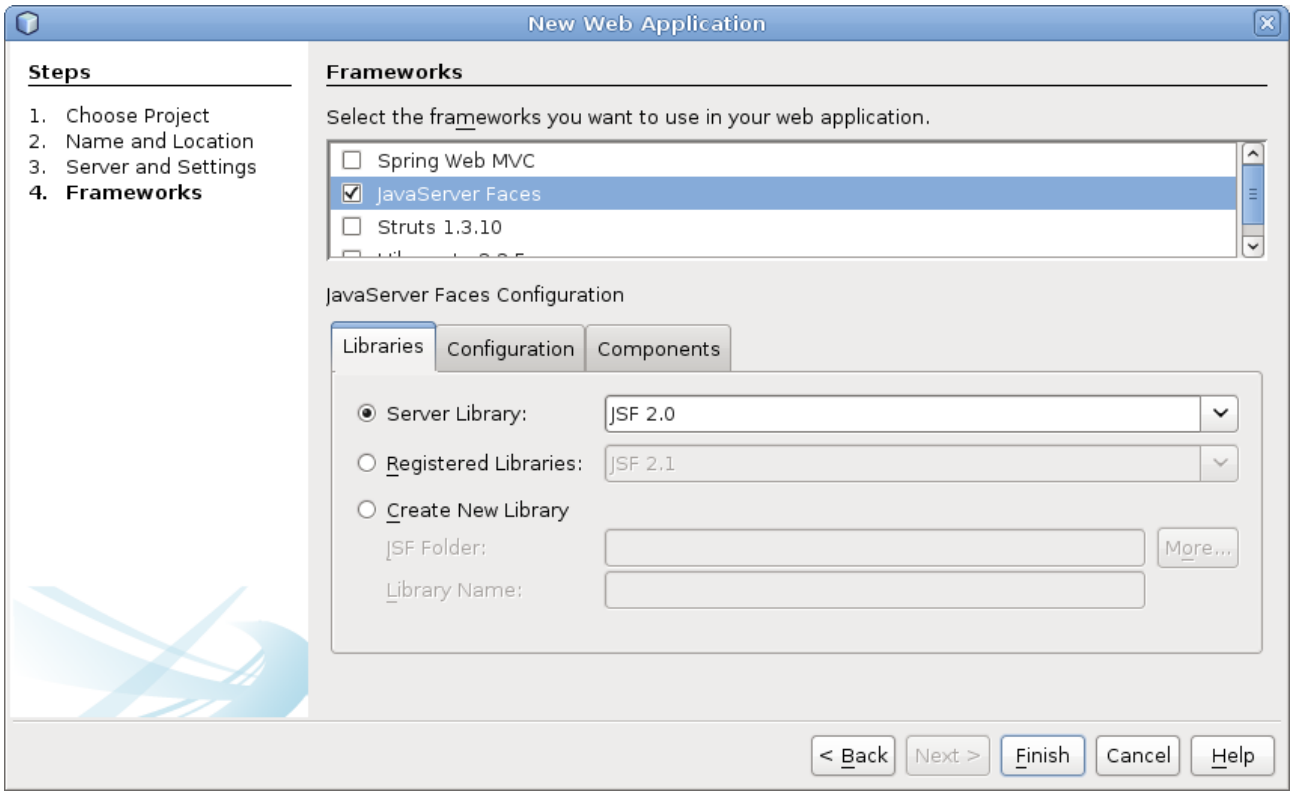
Indicamos el nombre del proyecto (MiAppJSF) y su ubicación:



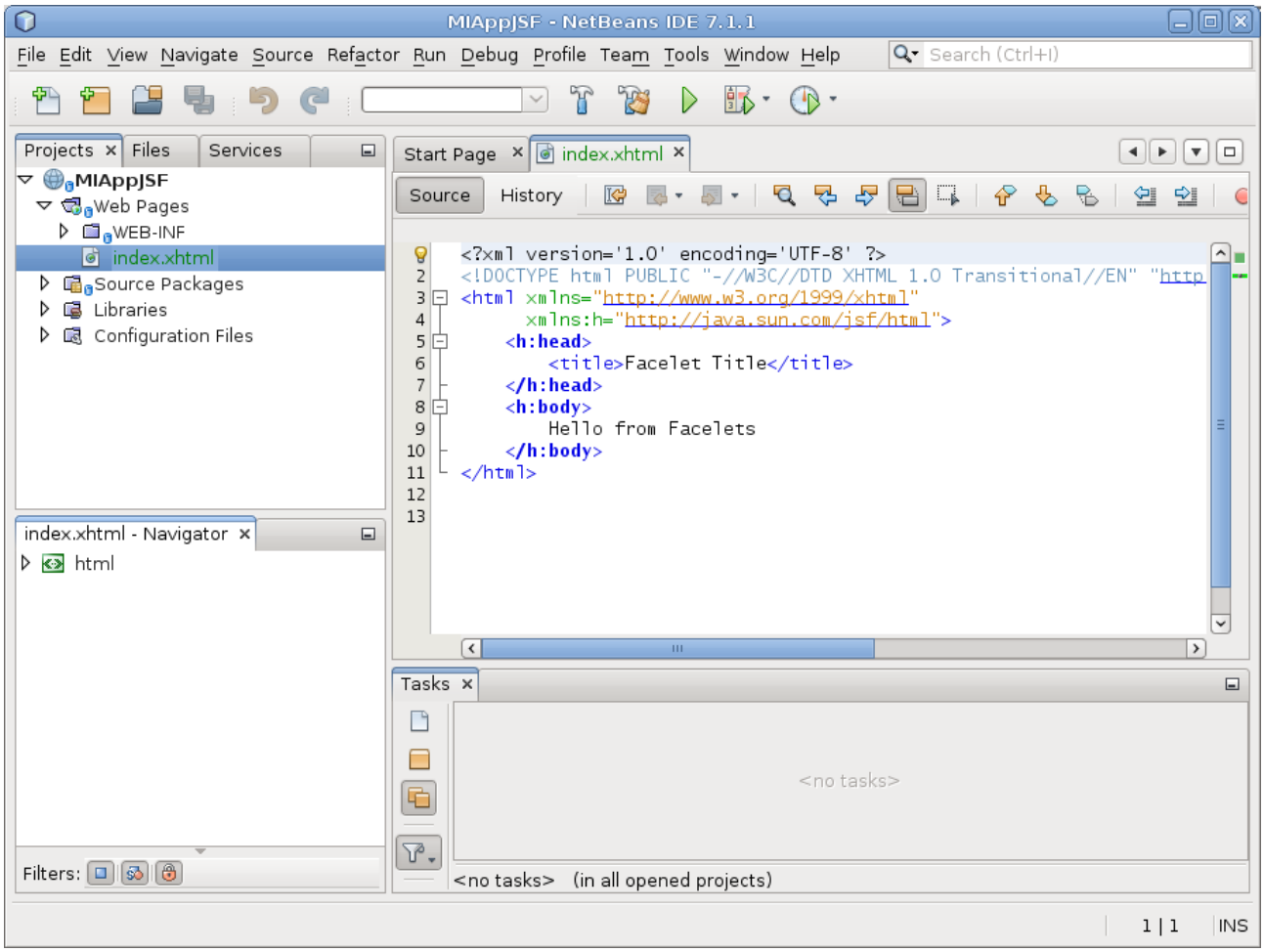
Indicamos el servidor a utilizar, el tipo de aplicación a desarrollar (J2EE 6), etc:



Indicamos el framework JSF a utilizar (JSF 2.0):

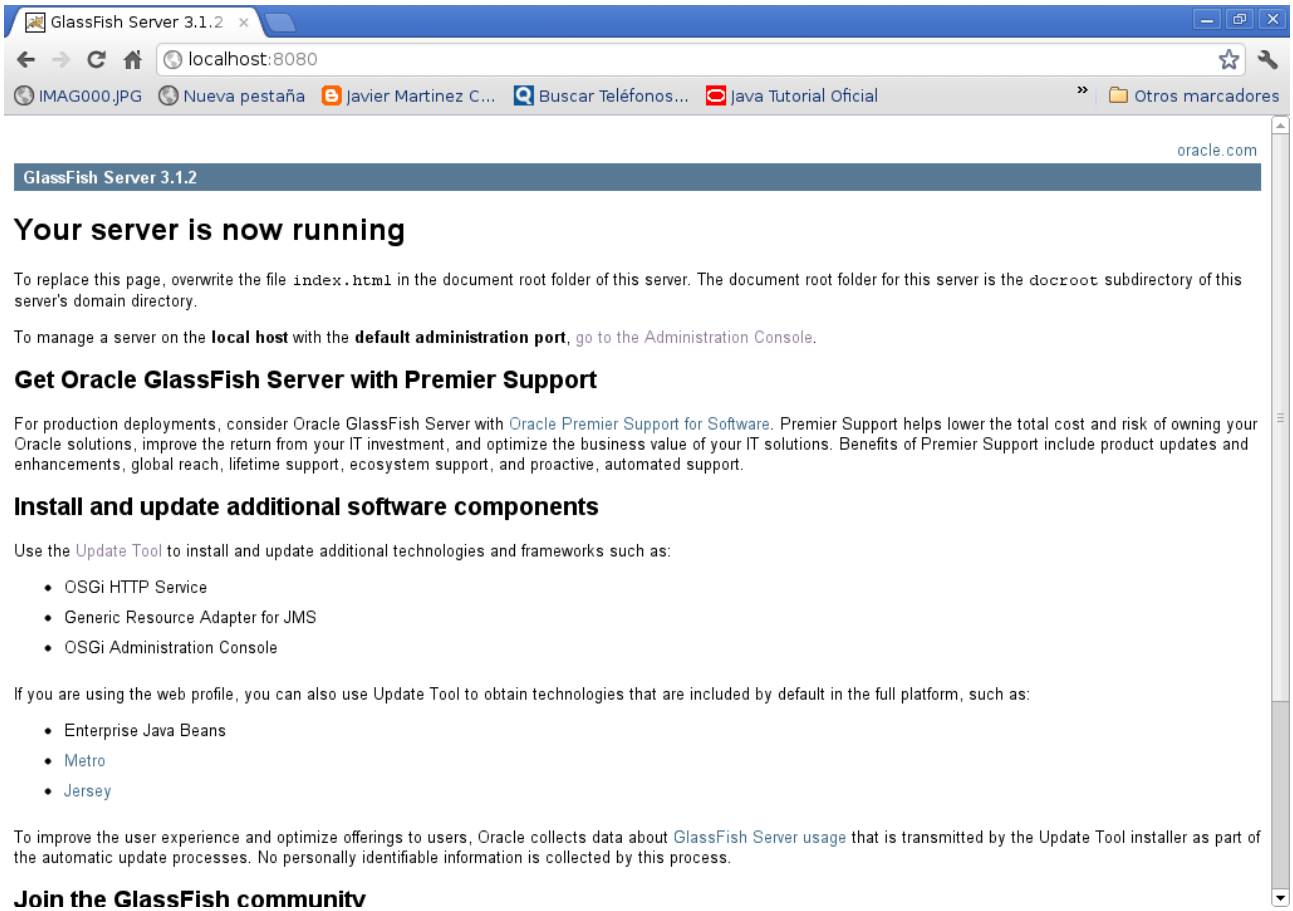


Presionamos “Finish”, se genera el proyecto con una simple pagina JSF que dice “Hello from Facelets”. Revise todo el contenido del proyecto:

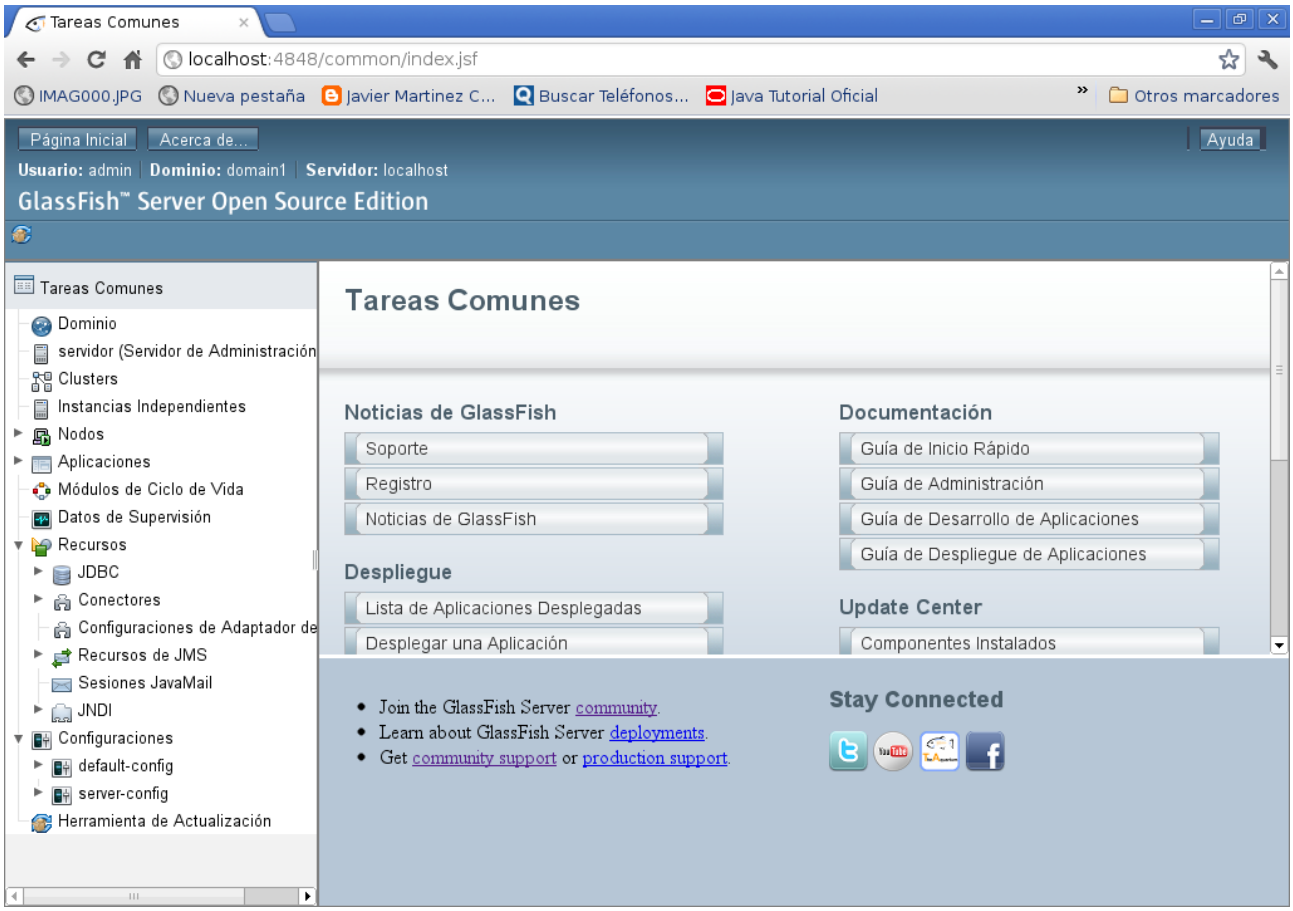


Ejecute el proyecto:

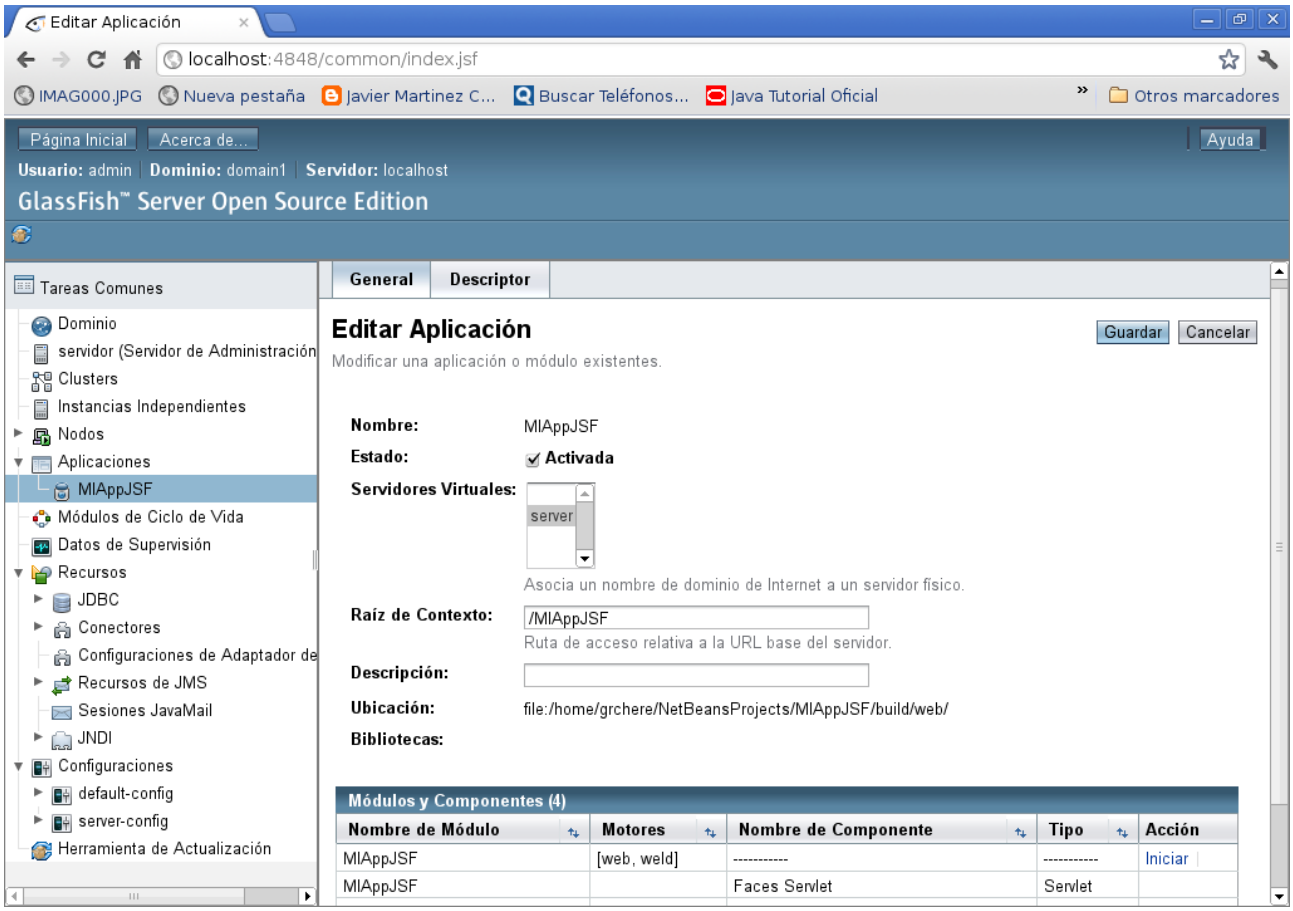
El ver el proyecto en ejecución implica también que se esta ejecutando el servidor GF, podemos acceder al mismo desde http://localhost:8080 :



También podemos acceder a la consola de administración de GF desde http://localhost:4848



NB ha compilado nuestro proyecto y lo ha publicado (deploy) en el servidor GF, aquí podemos ver nuestra aplicación desde GF:

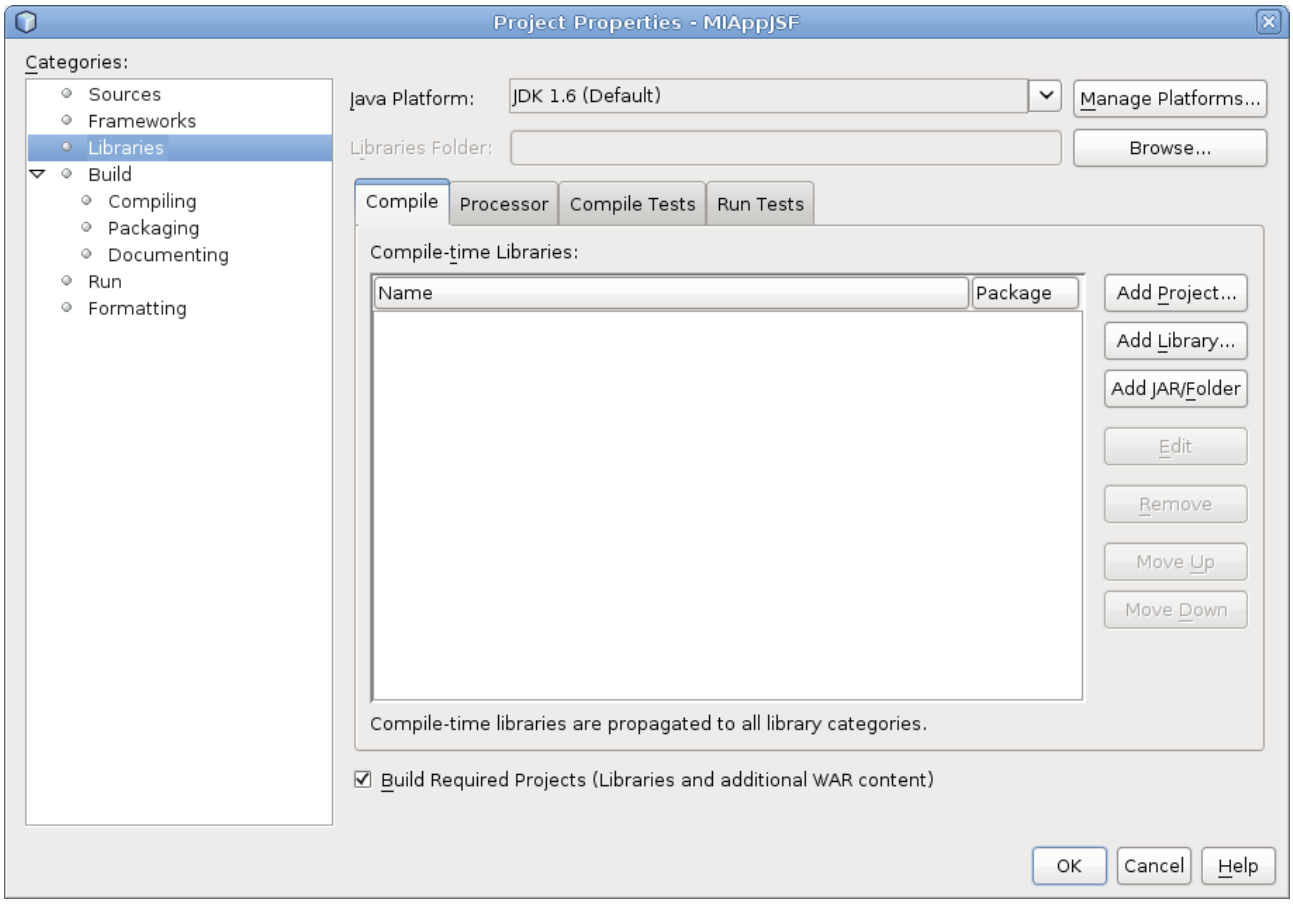

Editar Aplicación
localhost:4848/common/index.jsf
IMAG000.JPG
Nueva pestaña
Javier Martinez C...
Buscar Teléfonos...
Java Tutorial Oficial
Otros marcadores
Página Inicial
Acerca de...
Ayuda
Usuario: admin | Dominio: domain1 | Servidor: localhost
GlassFish™ Server Open Source Edition
Tareas Comunes
Dominio
servidor (Servidor de Administración
Clusters
Instancias Independientes
Nodos
Aplicaciones
MIAppJSF
Módulos de Ciclo de Vida
Datos de Supervisión
Recursos
JDBC
Conectores
Configuraciones de Adaptador de
Recursos de JMS
Sesiones JavaMail
JNDI
Configuraciones
default-config
server-config
Herramienta de Actualización
General
Descriptor
Editar Aplicación
Guardar
Cancelar
Modificar una aplicación o módulo existentes.
Nombre: MIAppJSF
Estado: Activada
Servidores Virtuales: server
Asocia un nombre de dominio de Internet a un servidor físico.
Raíz de Contexto: /MIAppJSF
Ruta de acceso relativa a la URL base del servidor.
Descripción:
Ubicación: file:/home/grchere/NetBeansProjects/MIAppJSF/build/web/
Bibliotecas:
Módulos y Componentes (4)
Nombre de Módulo
Motores
Nombre de Componente
Tipo
Acción
MIAppJSF
[web, weld]

Iniciar
MIAppJSF
Faces Servlet
Servlet

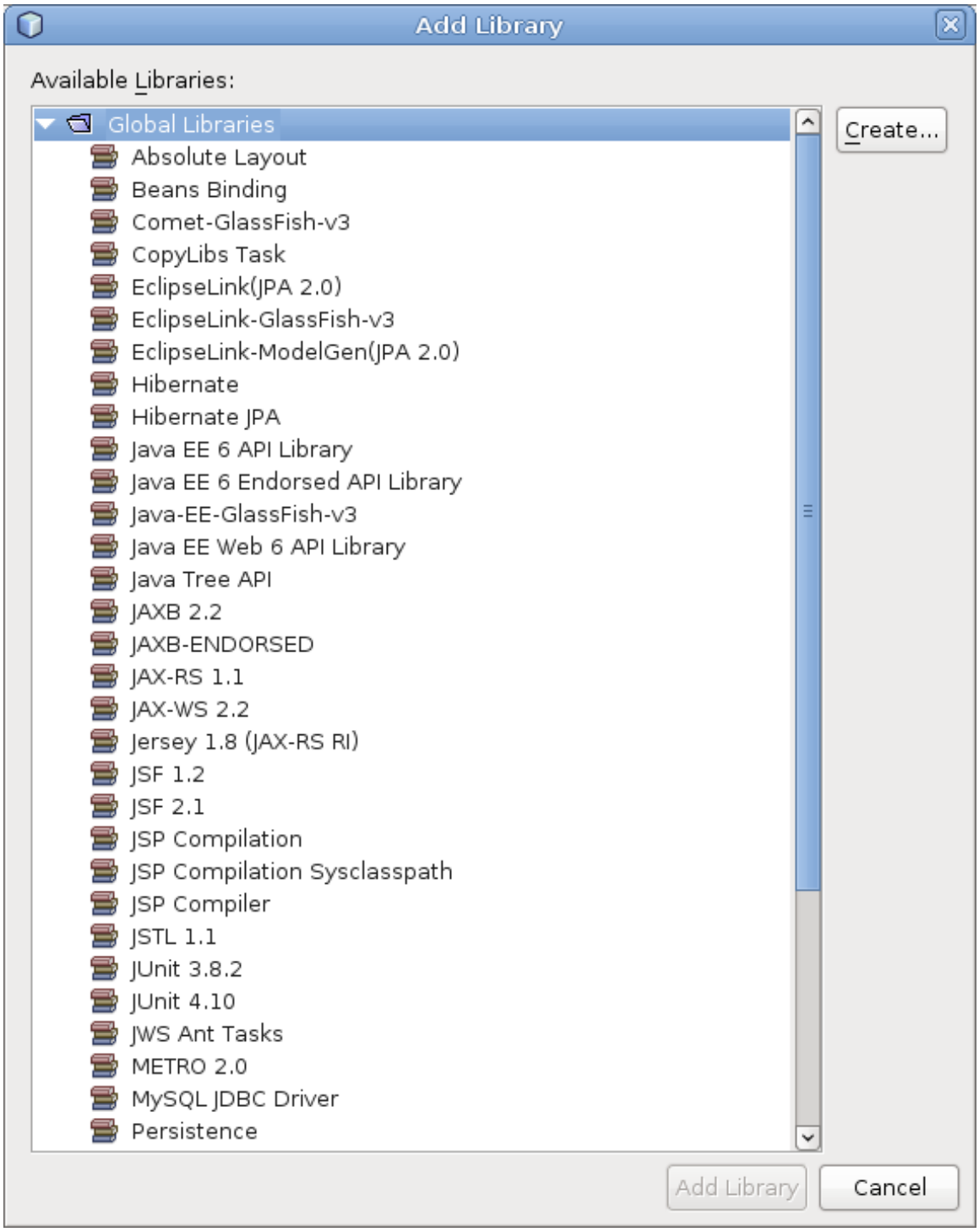
Incorporar PrimeFaces como Librería Global

Descargar primefaces 3.1.1 bundle (primefaces-3.1.1.zip) desde http://www.primefaces.org/downloads.html extrae primefaces-3.1.1.zip r en /home/<usuario>/primefaces-3.1 .

En NB, utilizamos la opción Tools / Libraries:

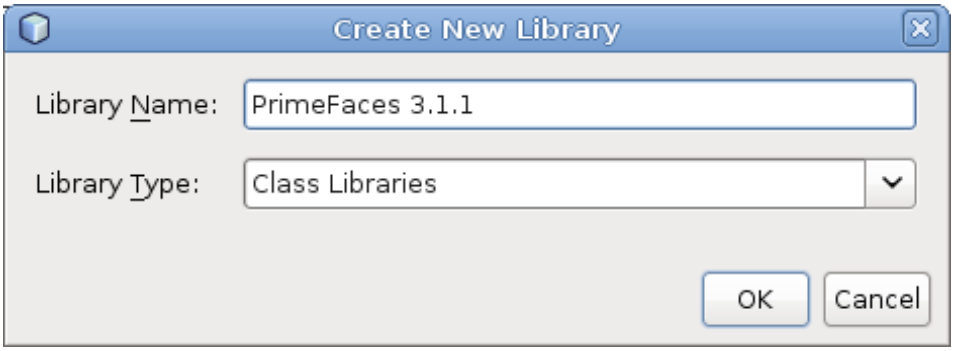


Click on “Add Library...” , aquí podemos ver todas las librerías globales (comunes a todos nuestros proyectos) que tenemos instaladas:

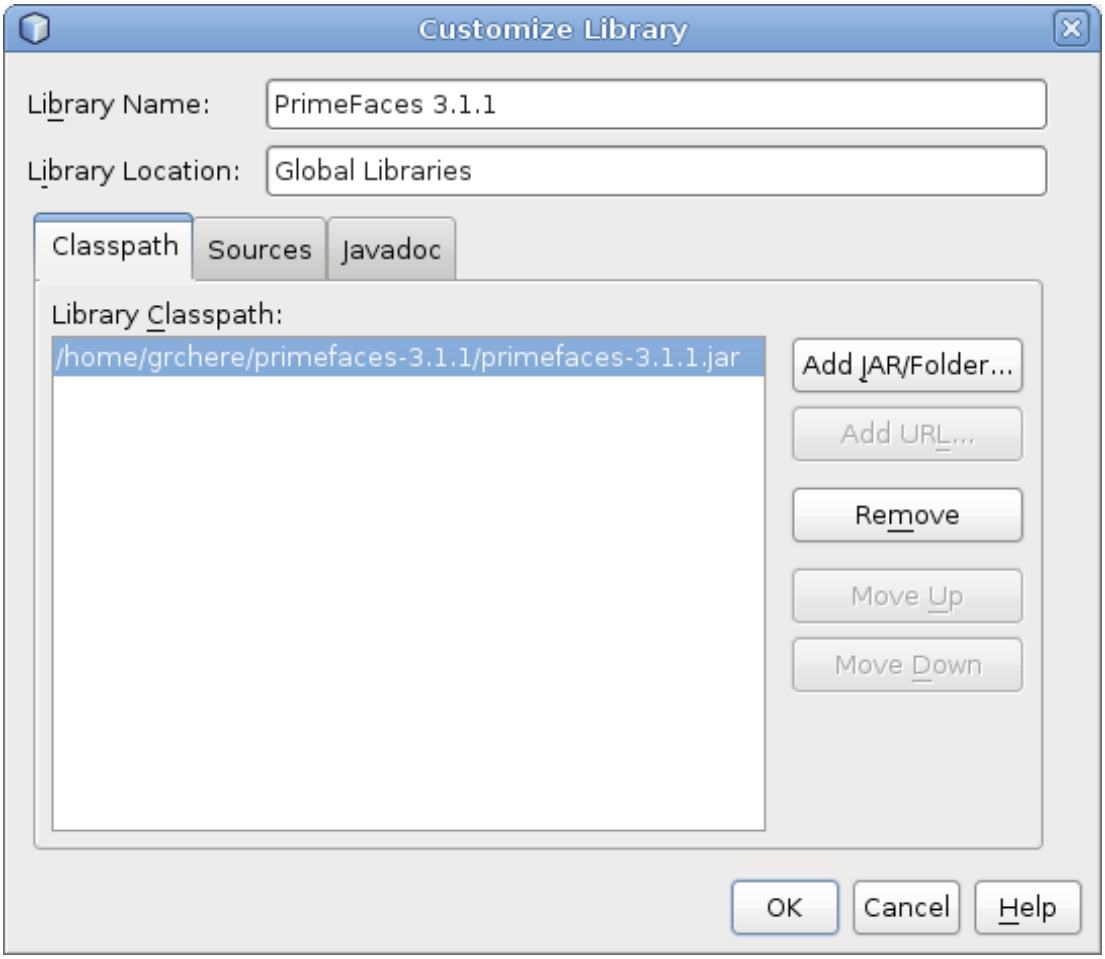


Clcik on “Create....”

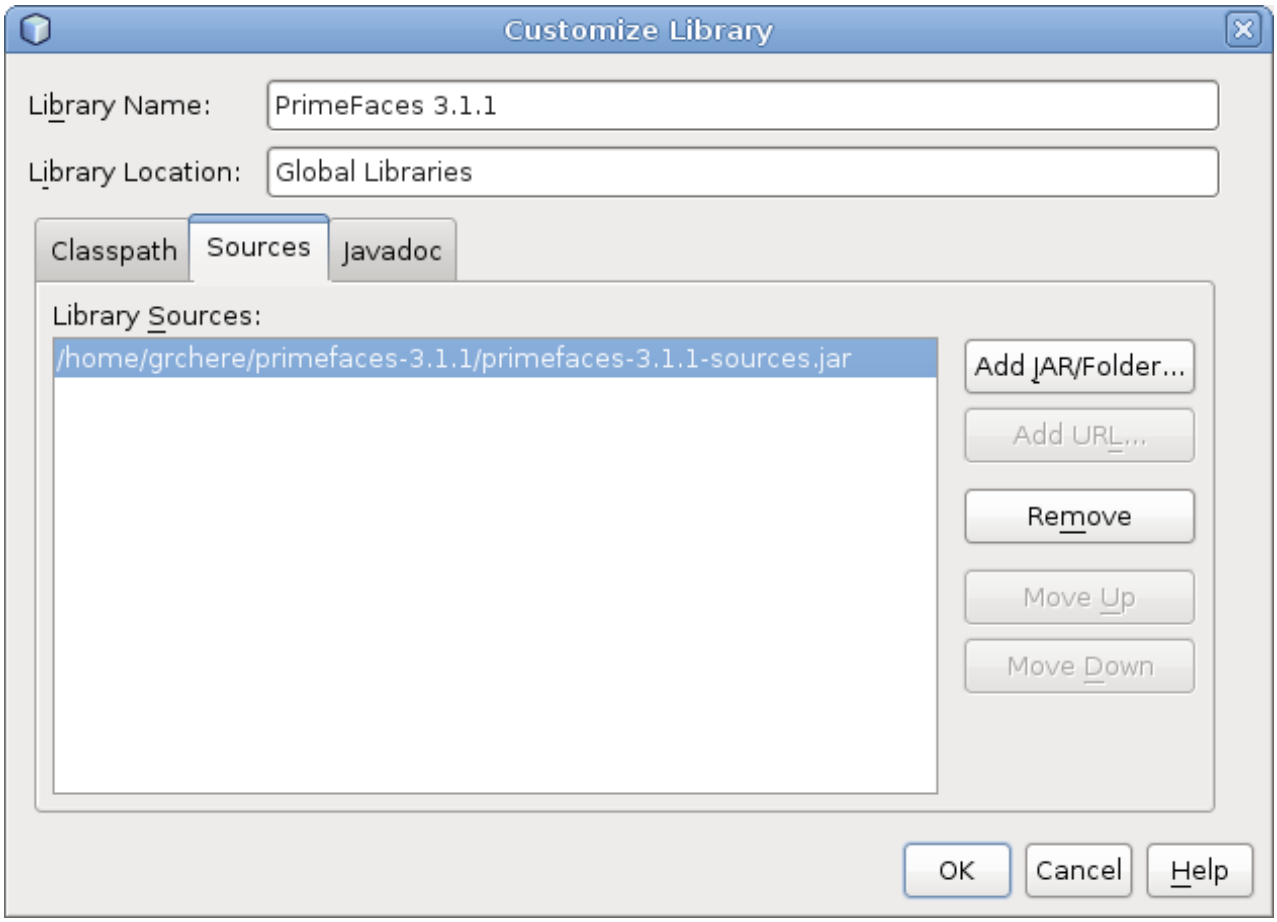
Indicamos el nombre de la librería global a crear, en este caso “PrimeFaces 3.1.1”:



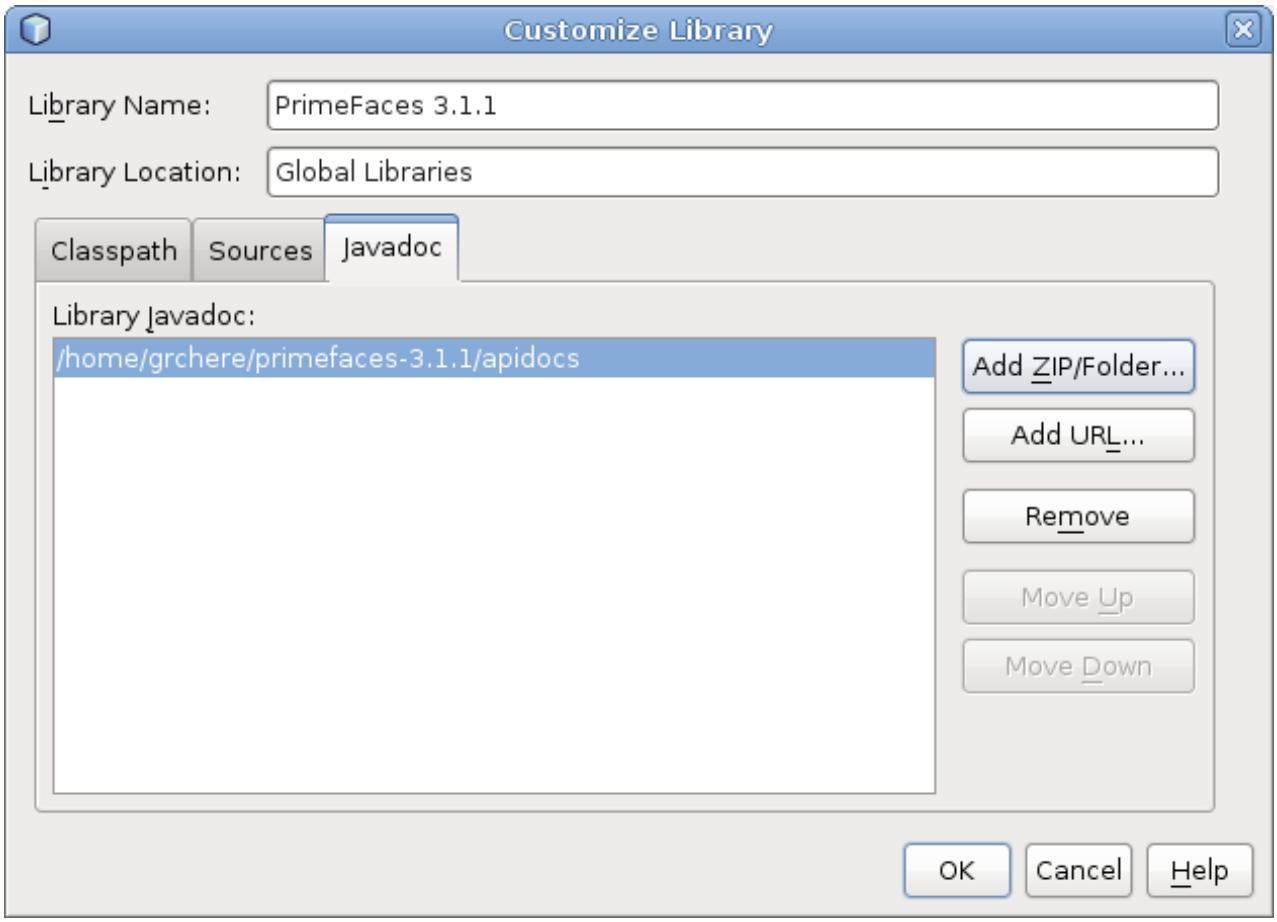
Click on “Ok”. En Classpath seleccionamos el archivo .jar de la librería desde el directorio en donde descomprimimos previamente a PF /home/<usuario>/primefaces-3.1



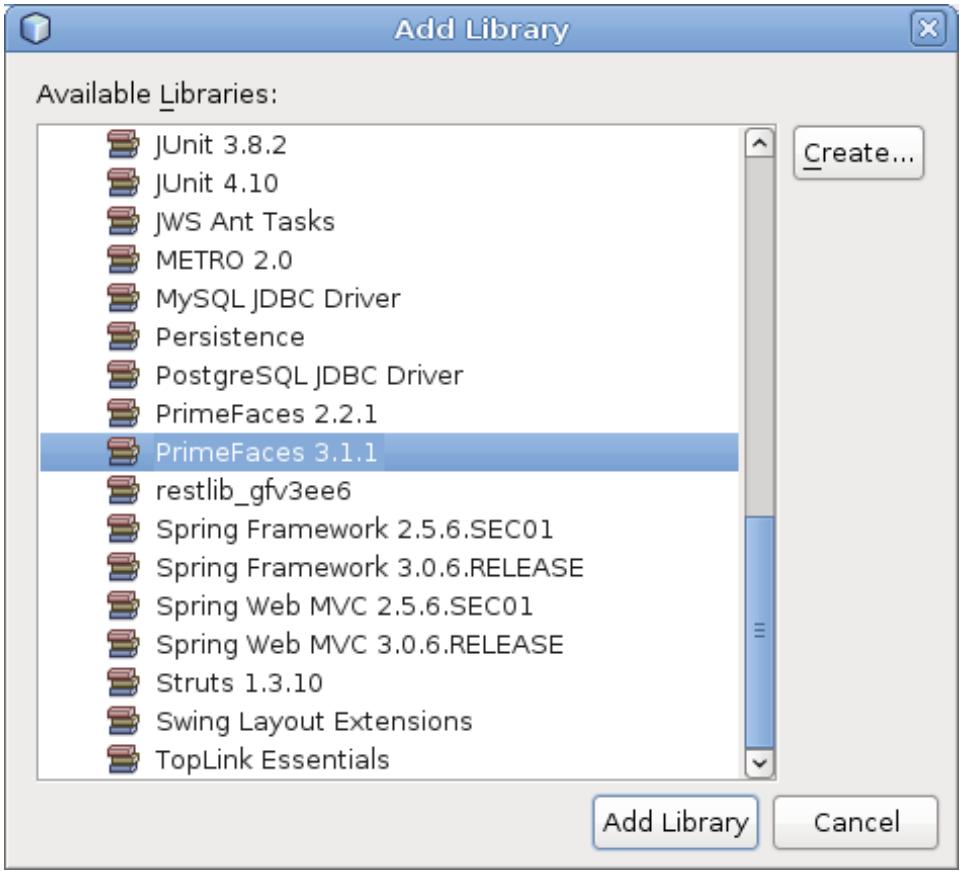
Hacemos algo similar con la lengüeta de Sources y seleccionamos el archivo .jar de fuentes de la librería:



Por último, seleccionamos la lengüeta de JavaDoc y seleccionamos la carpeta en donde se encuentra la documentación de la librería:



Presionamos Ok y ahora PF es una librería global de NB:


Add Library
Available Libraries:
JUnit 3.8.2
JUnit 4.10
JWS Ant Tasks
METRO 2.0
MySQL JDBC Driver
Persistence
PostgreSQL JDBC Driver
PrimeFaces 2.2.1
PrimeFaces 3.1.1
restlib_gfv3ee6
Spring Framework 2.5.6.SEC01
Spring Framework 3.0.6.RELEASE
Spring Web MVC 2.5.6.SEC01
Spring Web MVC 3.0.6.RELEASE
Struts 1.3.10
Swing Layout Extensions
TopLink Essentials
Create...
Add Library
Cancel

Incorporar JayBird 2.1.6 como Librería Global

Debemos hacer el mismo trabajo que hicimos anteriormente con PF, en este caso con el driver JDBC para el servidor Firebird SQL 2.5 JayBird 2.1.6.

Descargar driver jdbc de: http://www.firebirdsql.org/en/downloads/ hacer click on Connectivity / Jdbc Driver , descargar jaybird-2.1.6JDK_1.6.zip . Extraer jaybird-2.1.6JDK_1.6.zip en /home/<usuario>/jaybird-2.1.6.

En NB, utilizamos la opción Tools / Libraries y hacemos los mismos pasos que en el punto anterior, pero en este caso para JayBird, con los siguientes datos:

Nombre de librería: JayBird 2.1.6

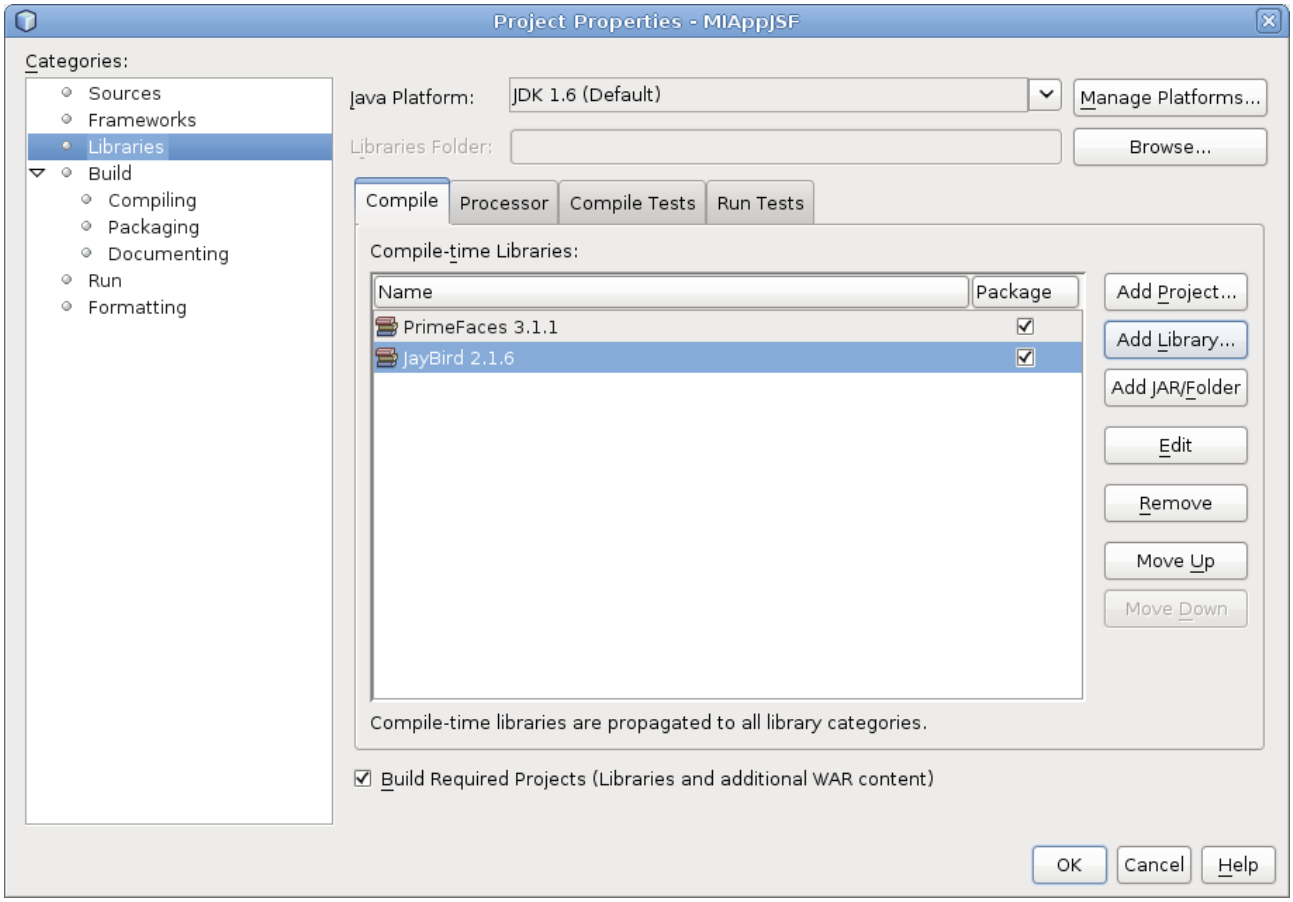
Archivo .jar de librería en classpath: jaybird-full-2.1.6.jar

Archivo .jar source: no tiene

Carpeta JavaDoc: /home/<usuario>/jaybird-2.1.6/docs/api

Incorporar PrimeFaces y JayBird al Proyecto

Debemos agregar estas dos librerías globales a nuestro proyecto, en la ventana de Projects seleccionamos nuestro proyecto, presionamos botón derecho, propiedades (o bien menú File / Properties), seleccionamos “Libraries” y hacemos click on “Add Library...” seleccionamos las librerías creadas en los dos puntos anteriores:

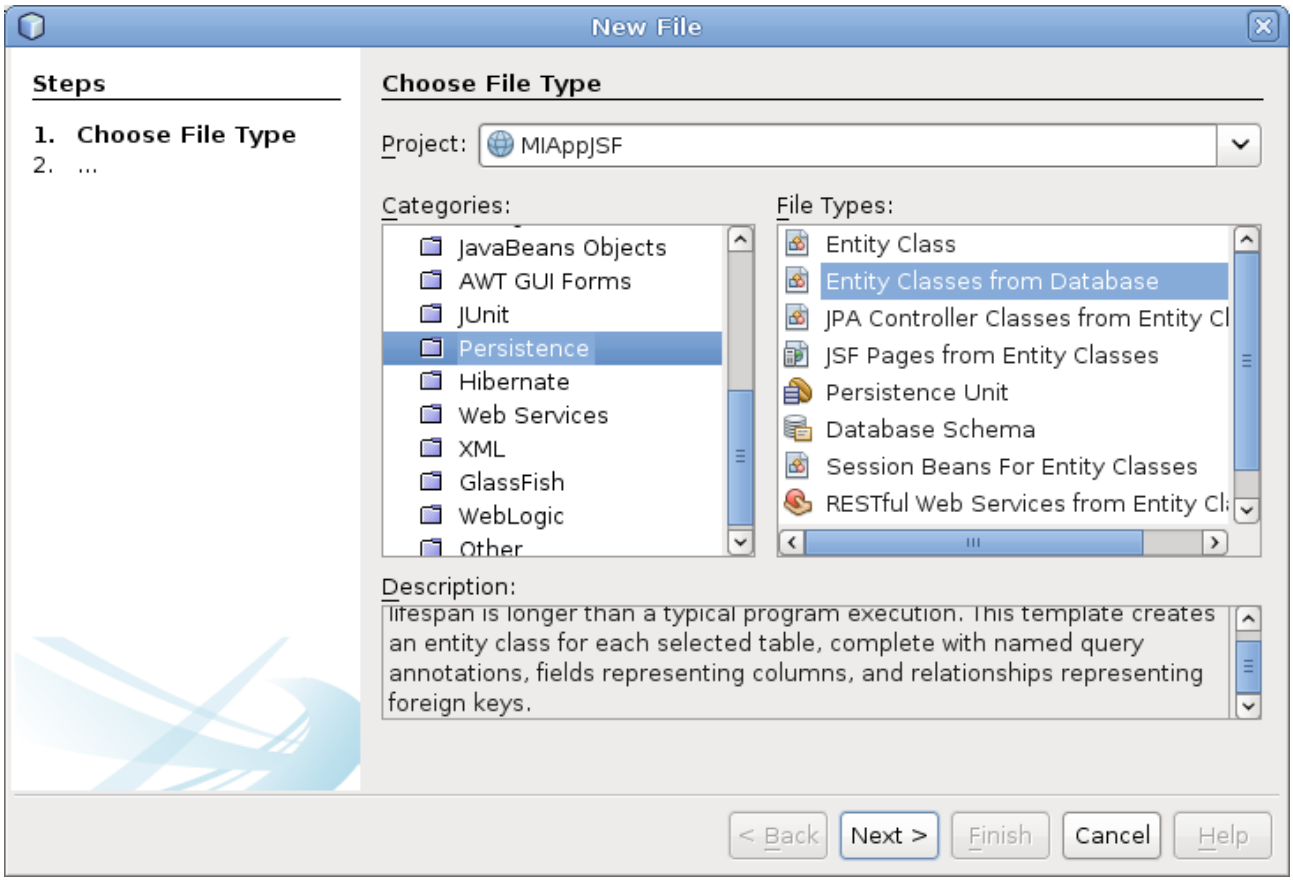


Volvemos a ejecutar el proyecto para verificar que todo marcha Ok:

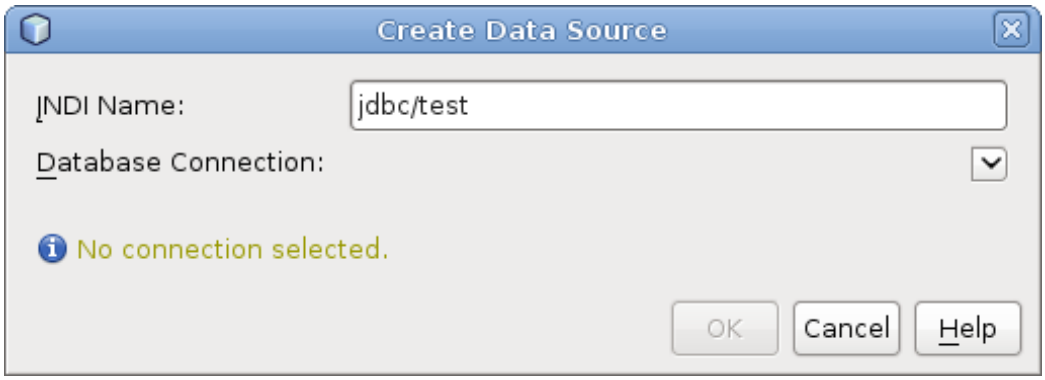
Hacer BACKUP.

Creación de Entidades utilizando JPA 2.0

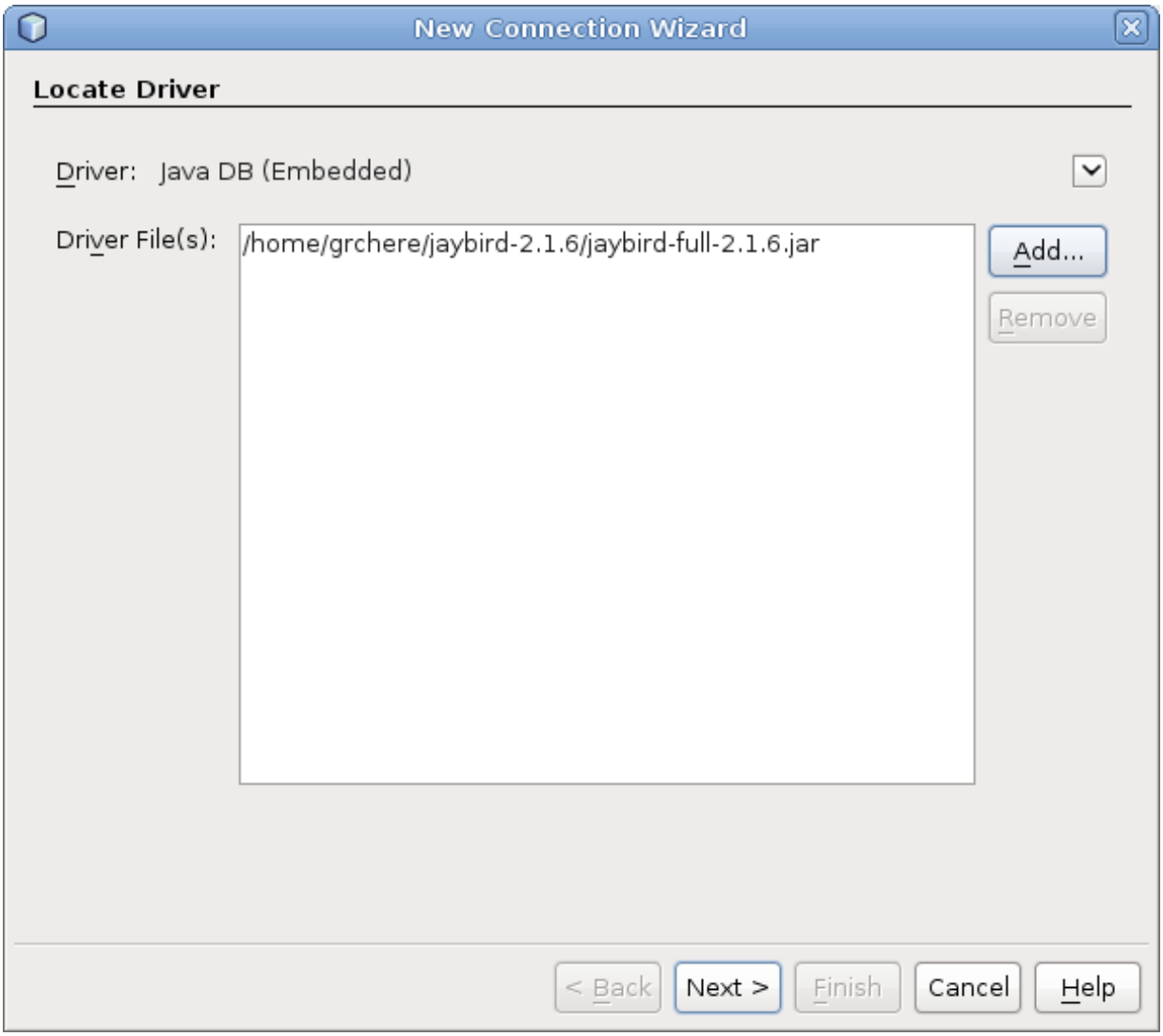
Presionar boton crear archivo (primer boton a la izquierda en la toolbar de NB), seleccionar Persistance / Entity Classes from Database:



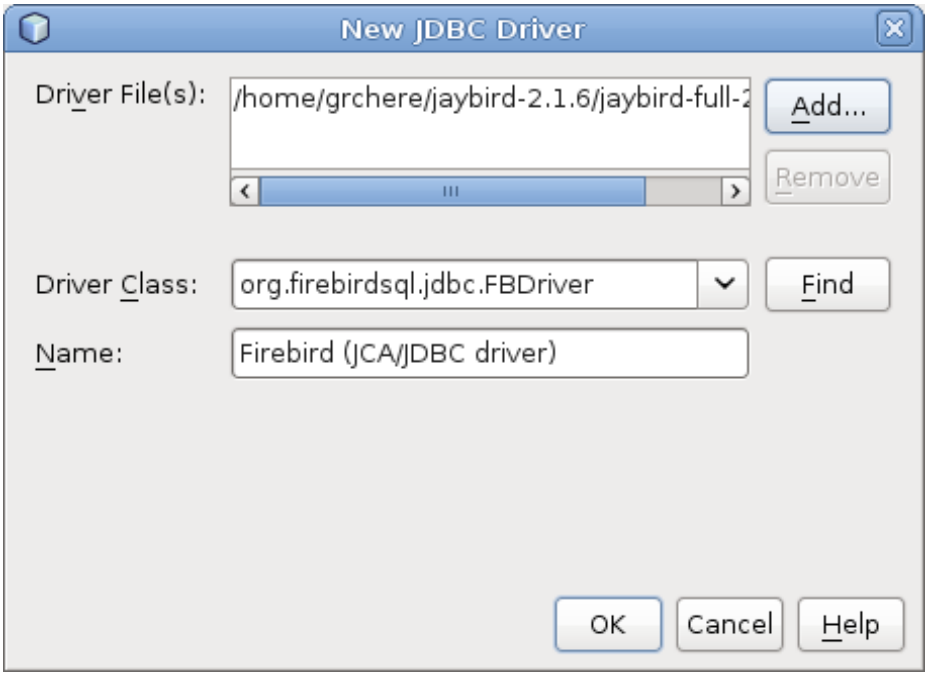
Desplegar combo Data Source: y seleccionar New DataSource...



Colocar como JNDI Name: jdbc/test (o bien reemplazar test por otro pseudo-nombre asociado a su base de datos); como Database Connection: desplegar combo y seleccionar New Database Connection. En dialogo Locate Driver, desplegar combo Driver y seleccionar New Driver:



en Driver Files, click en boton Add... y seleccionar home/<usuario>/jaybird-2.1.6/jaybird-full-2.1.6.jar ; el resto de los campos se autocompleta, presionar boton Ok, presionar Next (teniendo seleccionado a JayBird como driver jdbc a utilizar):



en dialogo Customize Connection, ingresar los siguientes datos:

User Name: sysdba

Password: masterkey

Click on remember password

en Jdbc url, hacer lo siguiente:

reemplazar <host> por localhost

reemplazar <port> por 3050

reemplazar <DB> por /var/lib/firebird/2.5/data/test.fdb (en este caso, seleccionar el archivo físico de base de datos que corresponda a su caso)

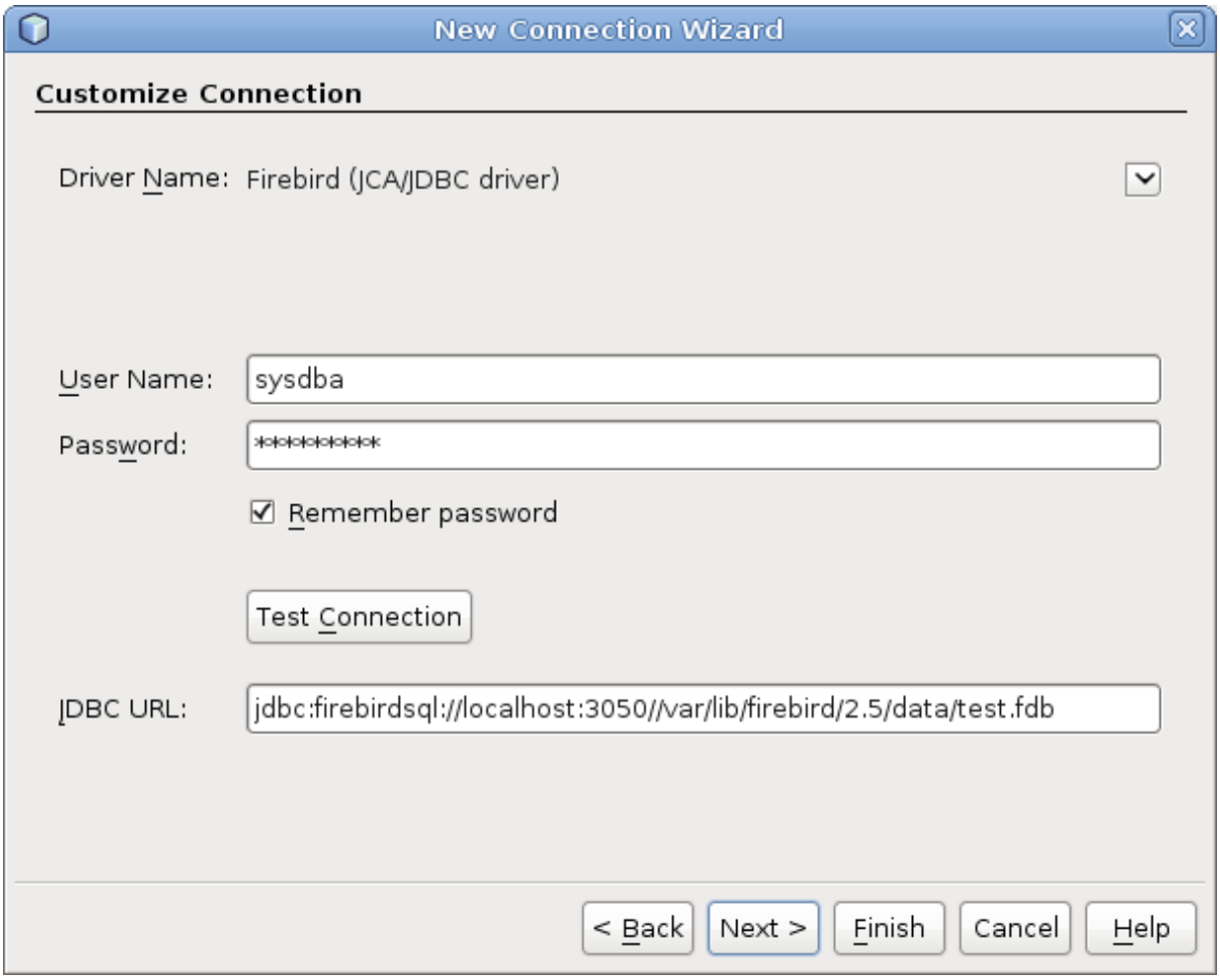
quite los corchetes ([])

presione boton de Test Connection para verificar que funciona la conexión jdbc con la base de datos FB

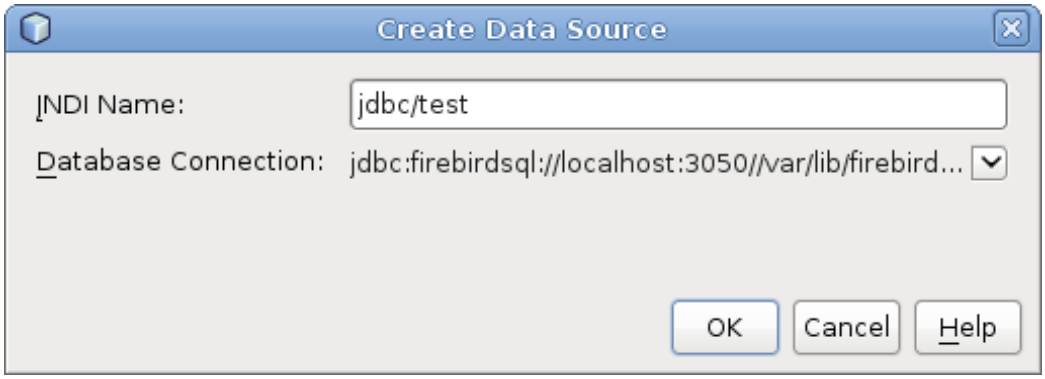
la url quedo como:

jdbc:firebirdsql://localhost:3050//var/lib/firebird/2.5/data/test.fdb

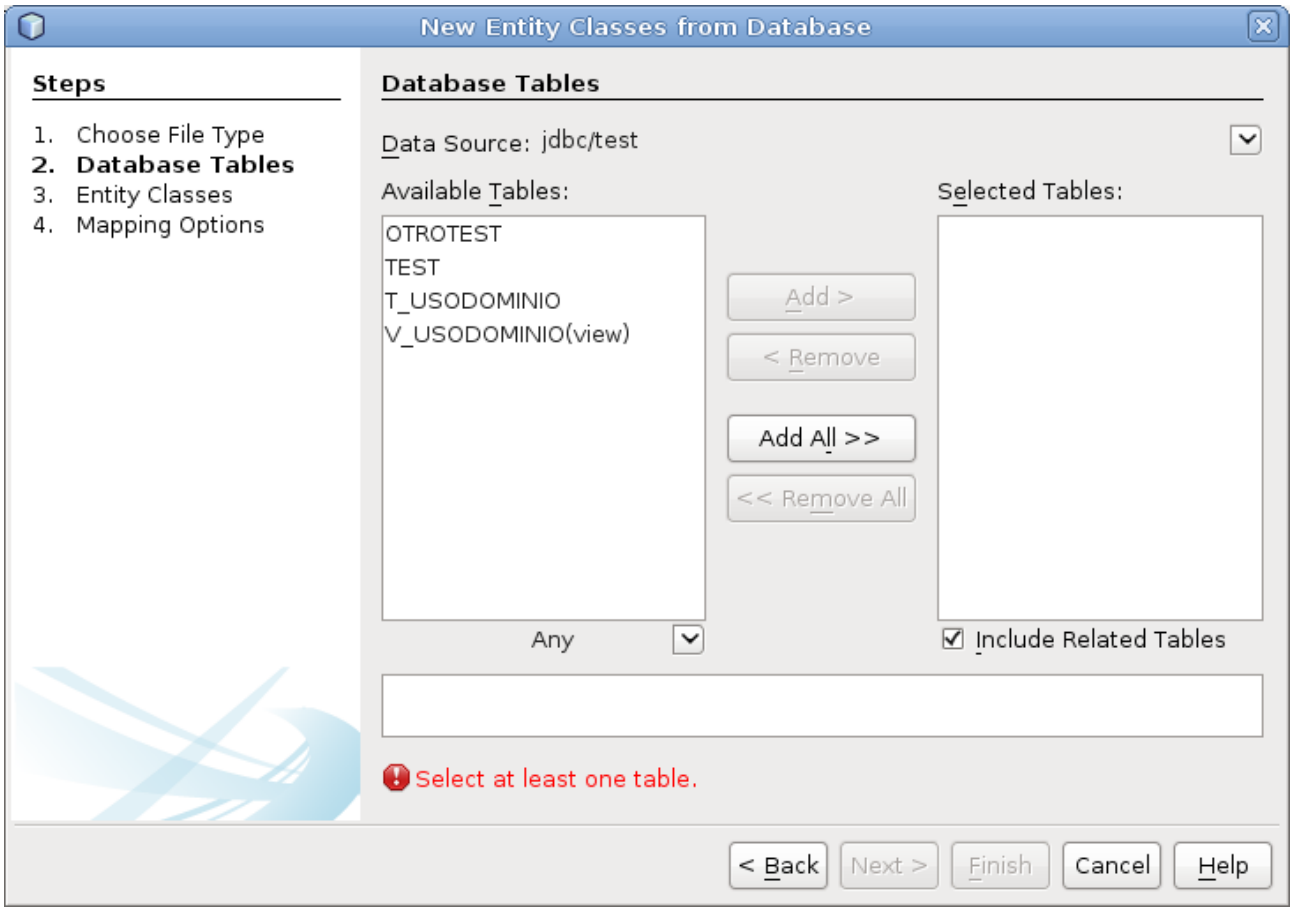
presione boton Next :

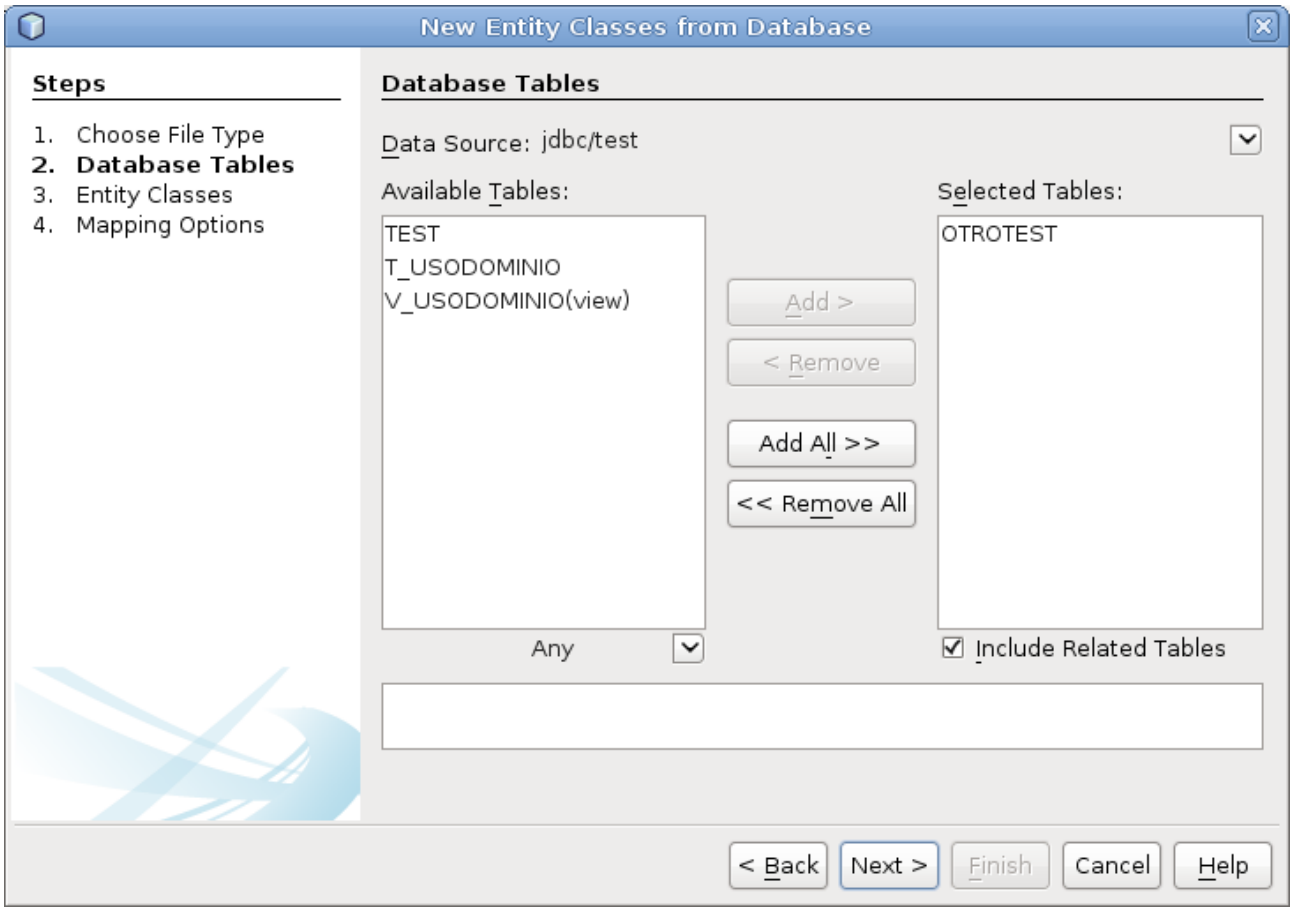


Presione boton Finish (en firebird no hay schema para elegir)
Presione boton Ok

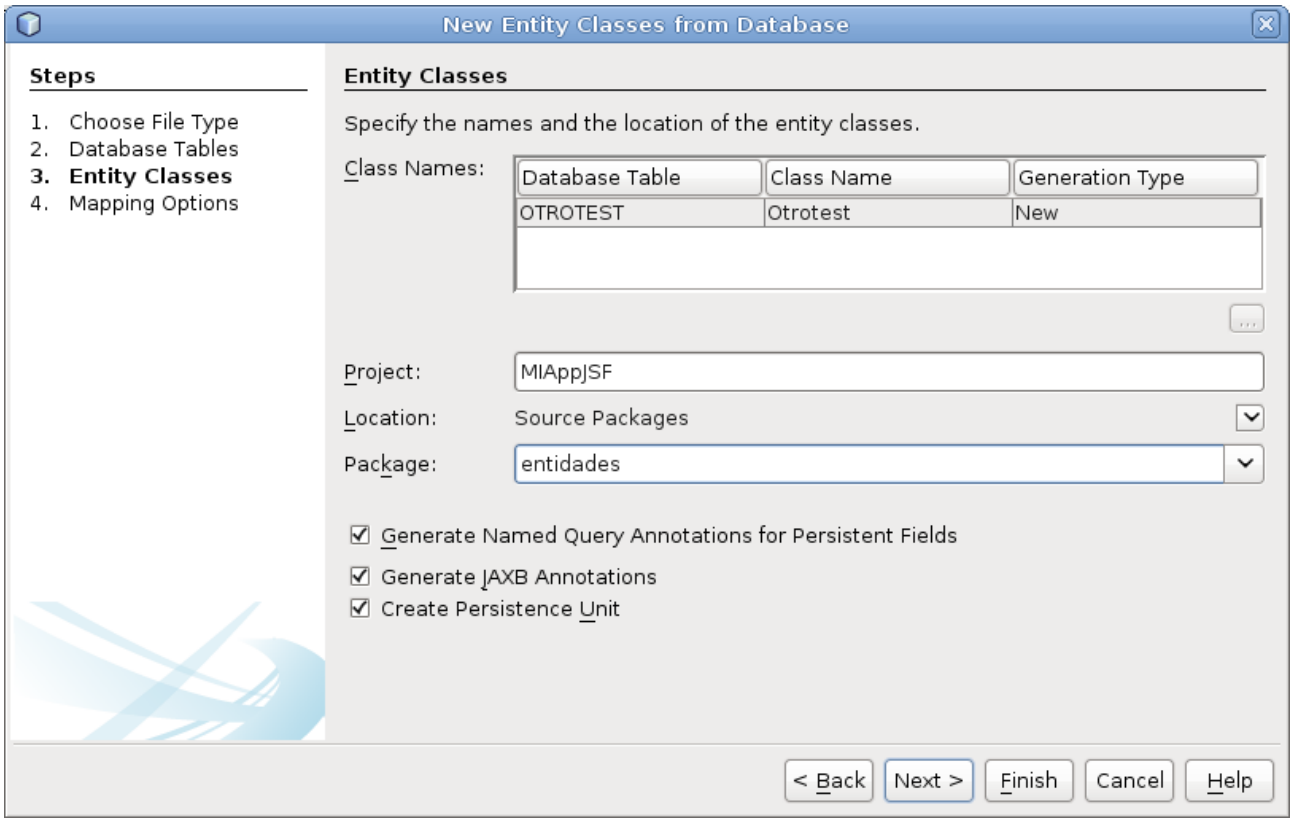


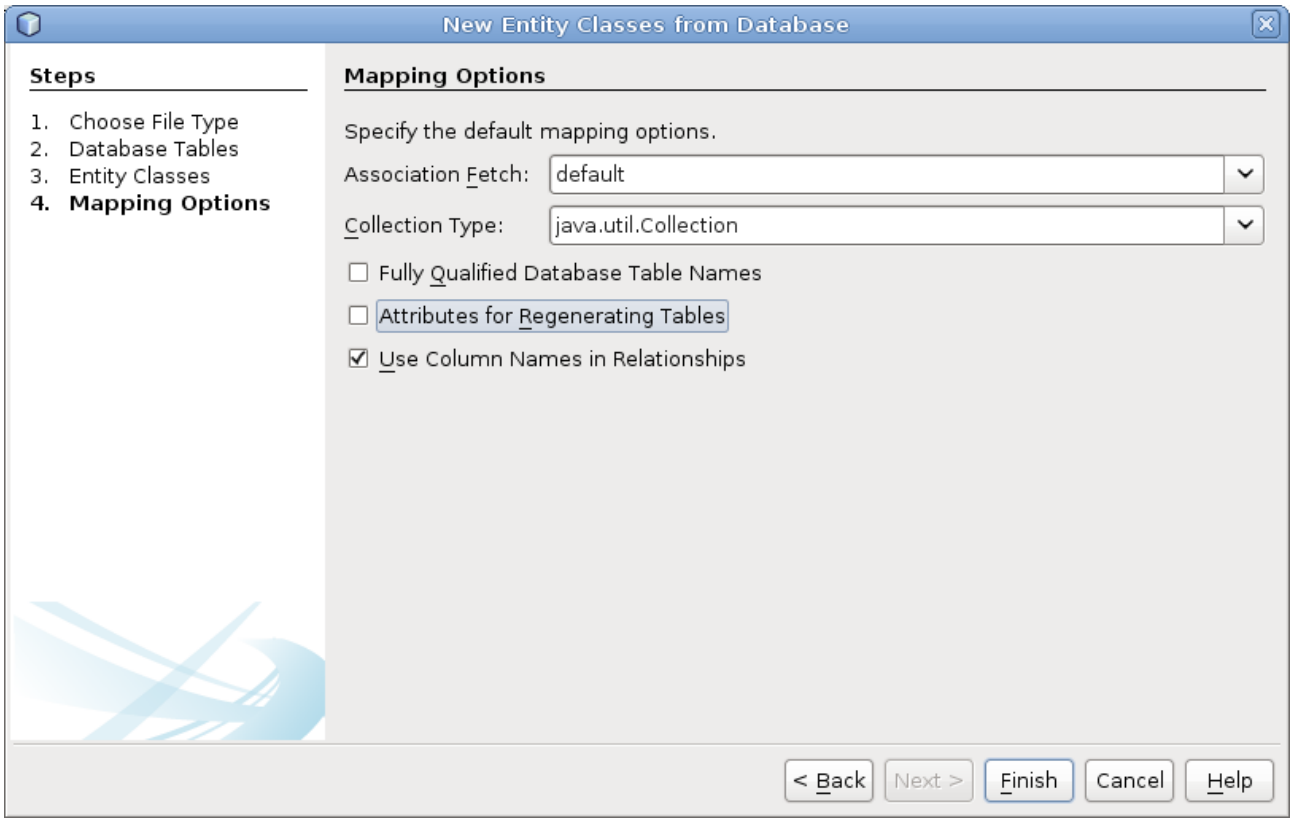
En dialogo New Entity Classes from Database ahora deben aparecer las tablas de la base de datos, seleccione una o mas tablas sobre las cuales generar entidades, presione Next:


New Entity Classes from Database
Steps
1. Choose File Type
2. Database Tables
3. Entity Classes
4. Mapping Options
Database Tables
Data Source: jdbc/test
Available Tables:
OTROTEST
TEST
T_USODOMINIO
V_USODOMINIO(view)
Add >
< Remove
Add All >>
<< Remove All
Selected Tables:
Any
Include Related Tables
Select at least one table.
< Back
Next >
Finish
Cancel
Help

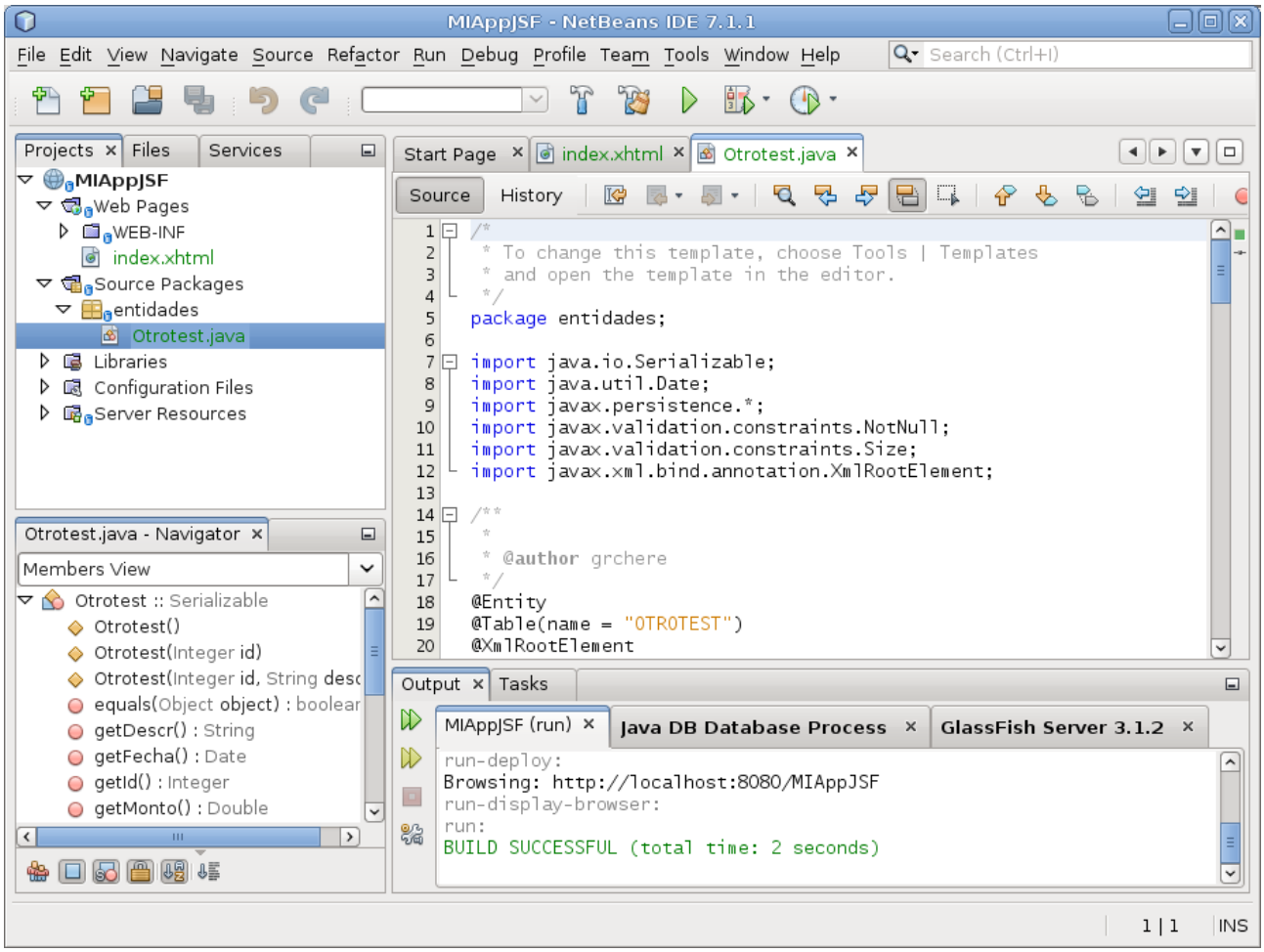


En dialogo Entity Classes, en Package Name tipee "entidades" (de esta forma se creará el package y quedarán allí todas la entidades a crear por este wizard), deje todas las opciones tildadas para que el wizard genere todo, click en boton Next, click en boton Finish:


New Entity Classes from Database
Steps
1. Choose File Type
2. Database Tables
3. Entity Classes
4. Mapping Options
Entity Classes
Specify the names and the location of the entity classes.
Class Names:
Database Table
Class Name
Generation Type
OTROTEST
Otrotest
New
Project:
MIAppJSF
Location:
Source Packages
Package:
entidades
Generate Named Query Annotations for Persistent Fields
Generate JAXB Annotations
Create Persistence Unit
< Back
Next >
Finish
Cancel
Help



Observe el package entidades, archivos de configuración y archivos fuente generados por el wizard que automatiza la creación de clases que utilizan JPA 2.0 para realizar ORM:



El wizard genera una clase por cada tabla, con las anotaciones requeridas por JPA 2.0. Se generan clases adicionales para aquellas entidades que tienen clave primaria compuesta (le agrega el sufijo PK al nombre de cada entidad).

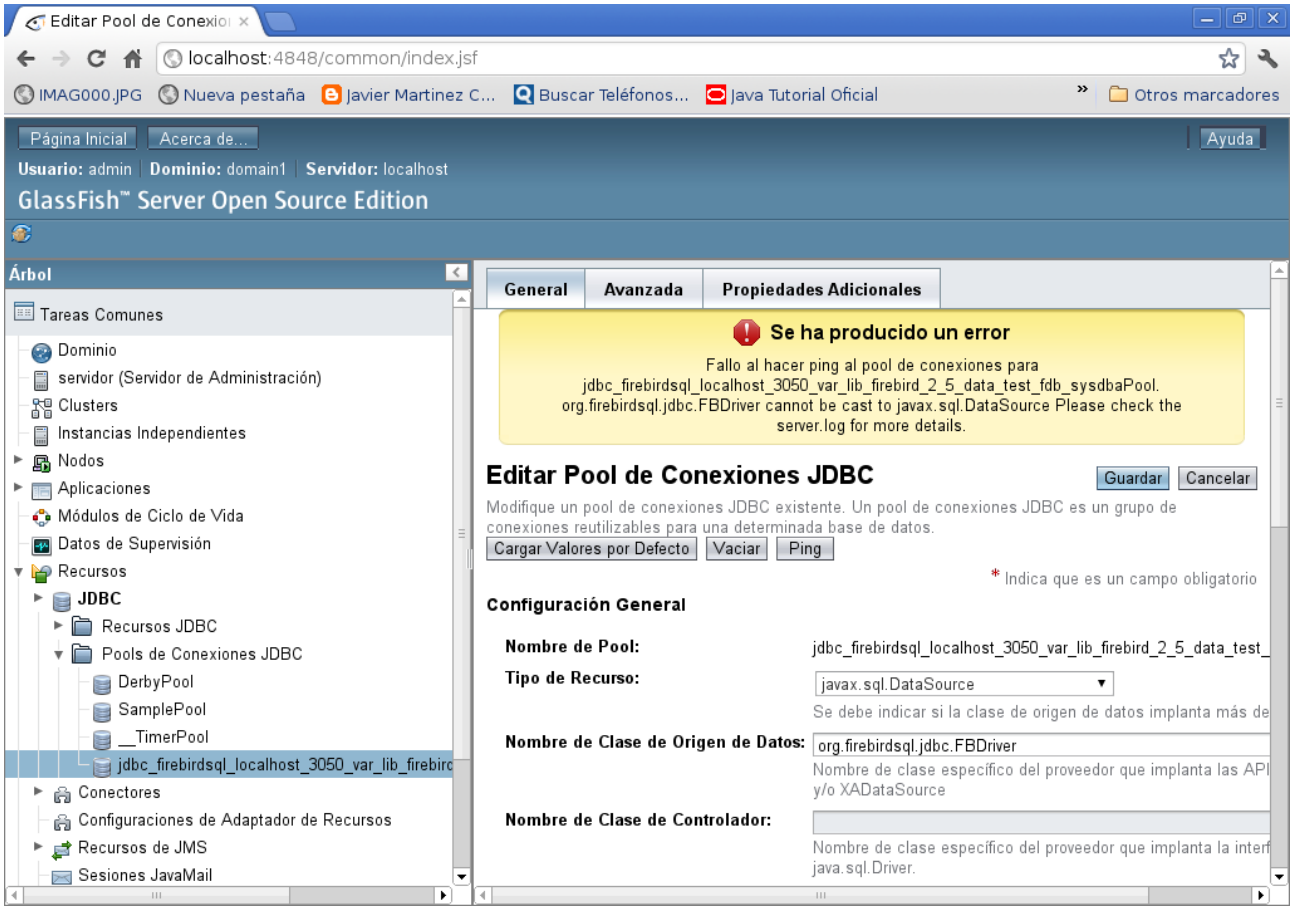
Ejecute el proyecto y verifique que el mismo continua funcionando (aunque visualmente no veamos diferencia, ya contamos con una representación en términos de objetos de una tupla proveniente de una tabla FB).

Hacer BACKUP.

JayBird y el Pool de Conexiones JDBC de Glassfish

Las entidades requieren de JDBC, dentro de GF JDBC es un recurso compartido por muchos usuarios, por lo tanto, GF crea un pool de conexiones JDBC para balancear la carga de trabajo contra la base de datos FB utilizando el driver jaybird. El pool de conexiones de GF es algo de tipo javax.sql.DataSource ; el problema es que el driver JDBC JayBird es algo de tipo org.firebirdsql.jdbc.FBDriver y ello no tiene que ver con javax.sql.DataSource (no puede “castearse” como javax.sql.DataSource) y esto produce un error en la configuración de GF a traves de NB.

Debemos utilizar otra clase -dentro de JayBird- que pueda “castearse” a algo de tipo javax.sql.DataSource. Aquí podemos observar el pool de conexiones JDBC creado por el deployment de la aplicación y vemos el error que se genera en GF:



Para solucionar el problema debemos editar el archivo glassfish-resources.xml dentro del proyecto NB, carpeta Server Resources:

Observe las propiedades del tag <jdbc-connection-pool> (name por ejemplo, para ver el nombre que genero el wizard de NB), allí se encuentra la propiedad datasource-classname="org.firebirdsql.jdbc.FBDriver" la misma debe cambiarse por:

```
datasource-classname="org.firebirdsql.jdbc.FBWrappingDataSource"
```

Observe las propiedades dentro del tag <jdbc-connection-pool> y hacer lo siguiente:

Agregar -si no existe- la propiedad:

```
<property name="portNumber" value="3050" />
```

Agregar -si no existe- la propiedad (indicando la ubicación de la base de datos FB):

```
<property name="dataBaseName"
value="/var/lib/firebird/2.5/data/test.fdb"/>
```

Agregar -si no existen- las siguiente propiedades:

```
<property name="JDBC30DataSource" value="true"/>
<property name="lc_ctype" value="ISO8859_1" />
```

Cambiar la propiedad driverClass (no obstante, luego sera comentada) por:

```
<property name="driverClass"
value="org.firebirdsql.jdbc.FBWrappingDataSource"/>
```

Comentar (encerrar entre tags html de comentario <!-- ... -->) las propiedades URL y driverClass:

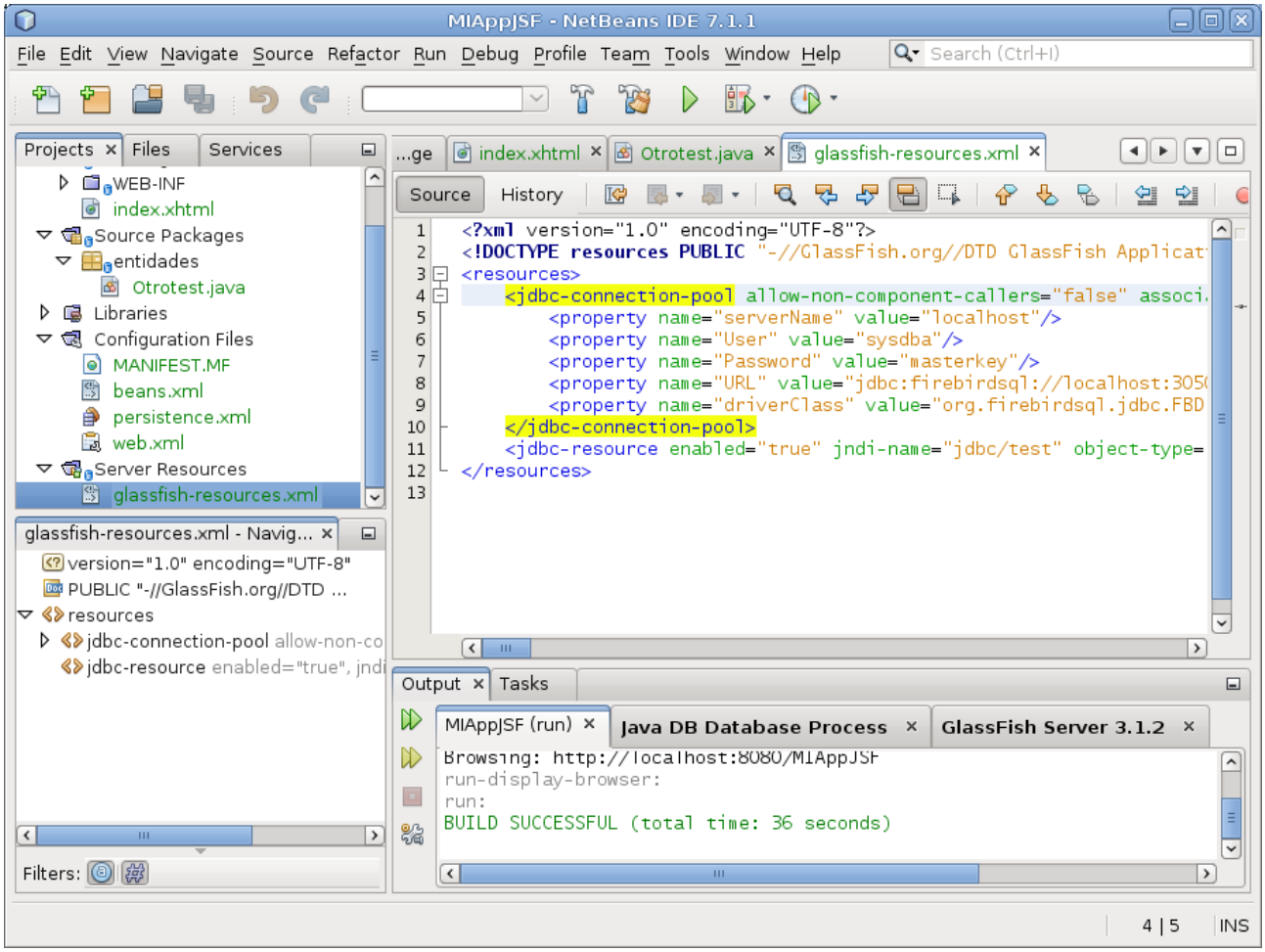
```
<!--
<property name="driverClass"
value="org.firebirdsql.jdbc.FBWrappingDataSource"/>
<property name="URL" value =
"jdbc:firebirdsql://localhost:3050//var/lib/firebird/2.5/data/test.fdb"/
>
-->
```

el editor va a "grisar" la visualizacion de esa parte del archivo xml

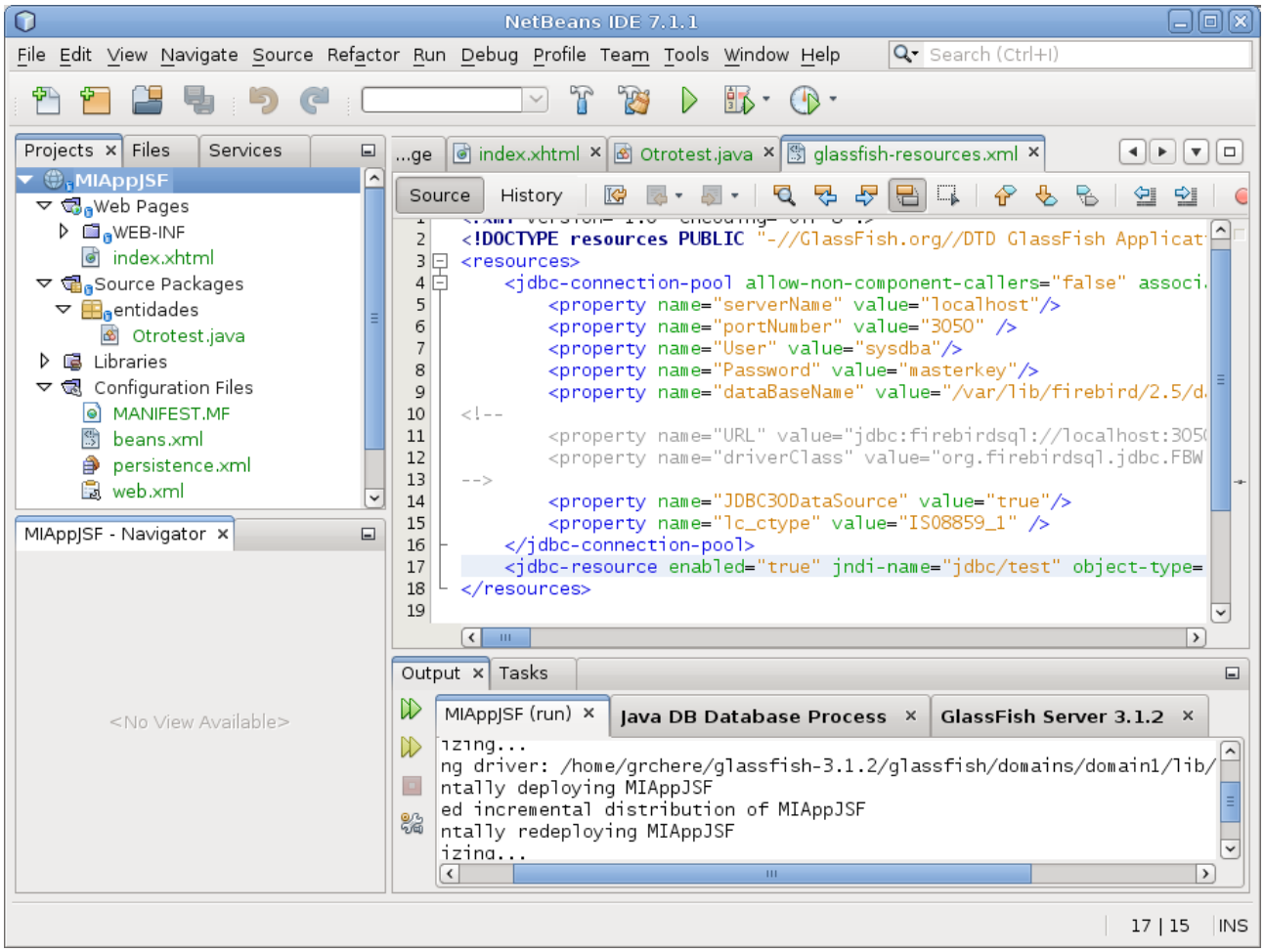
Grabar el archivo glassfish-resources.xml , el cual podría tener un contenido similar a este:

```
...
<resources>
    <jdbc-connection-pool allow-non-component-callers="false" associate-with-
thread="false" connection-creation-retry-attempts="0" connection-creation-
retry-interval-in-seconds="10" connection-leak-reclaim="false" connection-leak-
timeout-in-seconds="0" connection-validation-method="auto-commit" datasource-
classname="org.firebirdsql.jdbc.FBDriver" fail-all-connections="false" idle-
timeout-in-seconds="300" is-connection-validation-required="false" is-
isolation-level-guaranteed="true" lazy-connection-association="false" lazy-
connection-enlistment="false" match-connections="false" max-connection-usage-
count="0" max-pool-size="32" max-wait-time-in-millis="60000"
name="jdbc_firebirdsql_localhost_3050_var_lib_firebird_2_5_data_test_fdb_sysdba
Pool" non-transactional-connections="false" pool-resize-quantity="2" res-
type="javax.sql.DataSource" statement-timeout-in-seconds="-1" steady-pool-
size="8" validate-atmost-once-period-in-seconds="0" wrap-jdbc-objects="false">
        <property name="serverName" value="localhost"/>
        <property name="portNumber" value="3050" />
        <property name="User" value="sysdba"/>
        <property name="Password" value="masterkey"/>
        <property name="dataBaseName"
value="/var/lib/firebird/2.5/data/test.fdb"/>
<!--
        <property name="URL"
value="jdbc:firebirdsql://localhost:3050//var/lib/firebird/2.5/data/test.fdb"/>
        <property name="driverClass"
value="org.firebirdsql.jdbc.FBWrappingDataSource"/>
-->
```

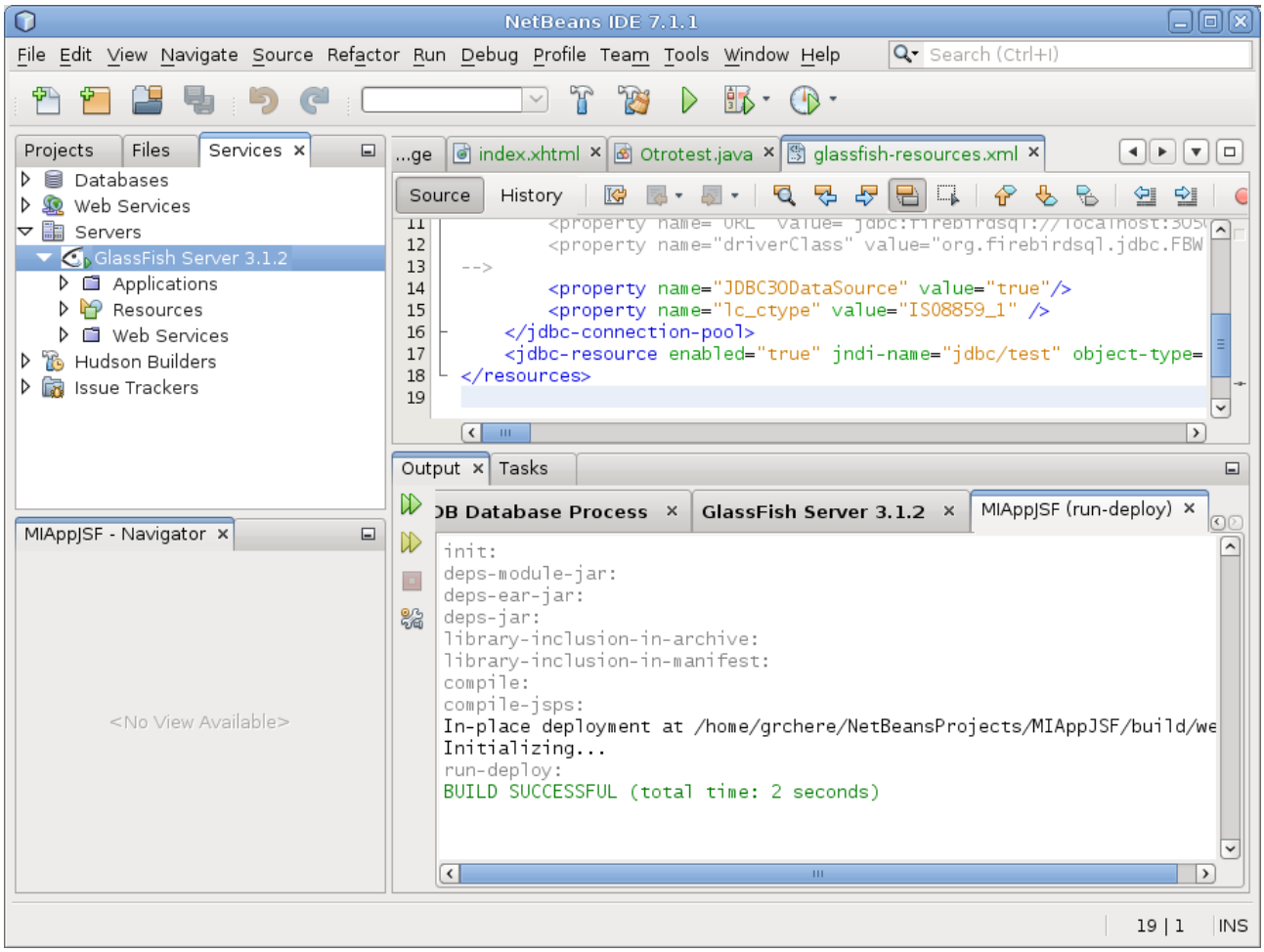
```
        <property name="JDBC30DataSource" value="true"/>
        <property name="lc_ctype" value="ISO8859_1" />
    </jdbc-connection-pool>
    <jdbc-resource enabled="true" jndi-name="jdbc/test" object-type="user"
pool-
name="jdbc_firebirdsql_localhost_3050_var_lib_firebird_2_5_data_test_fdb_sysdba
Pool"/>
</resources>
```



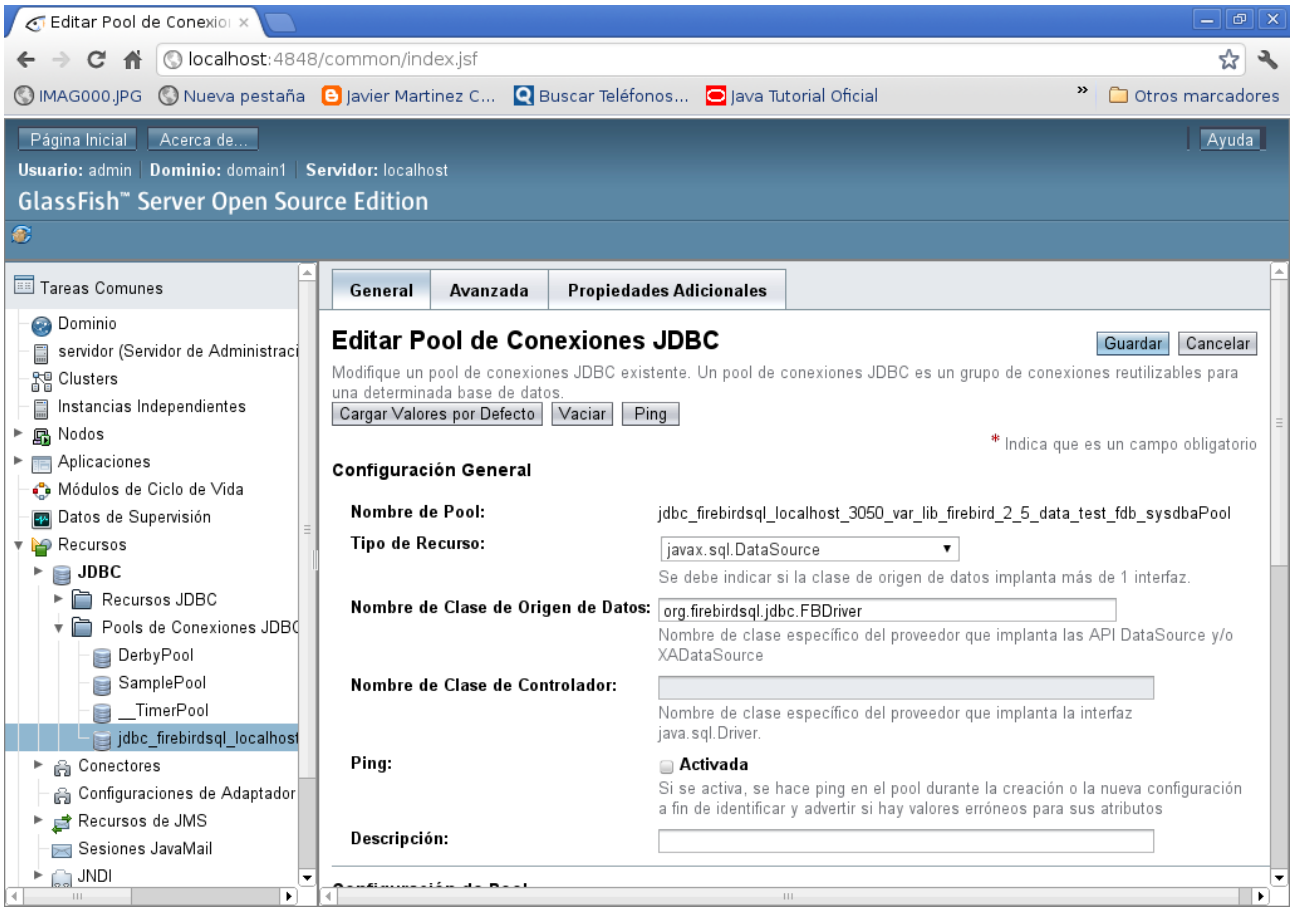
Luego de los cambios, el archivo glassfish-resources.xml podría tener este aspecto:



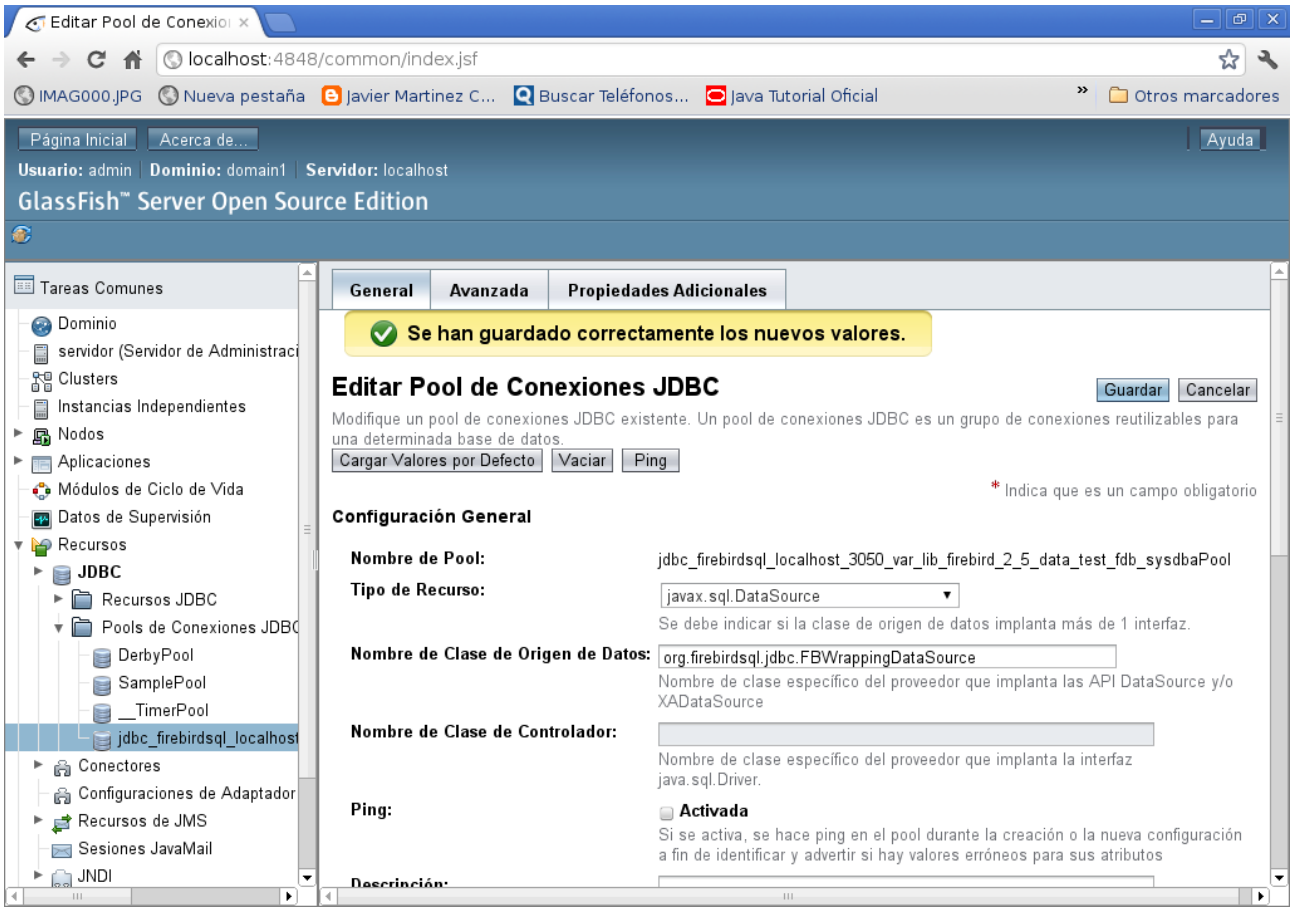
Es posible que deba detener y volver a arrancar el servidor GF (usar lengüeta Services en NB, seleccionar server, boton derecho, Restart) si el mismo no toma los cambios realizados en este archivo (luego de los cambios, el mismo debería ser deployed en GF) o bien modificarlo directamente en GF. No obstante siempre debemos tener estos cambios en el archivo glassfish-resources.xml puesto que NB siempre hara deployment del mismo sobre GF:



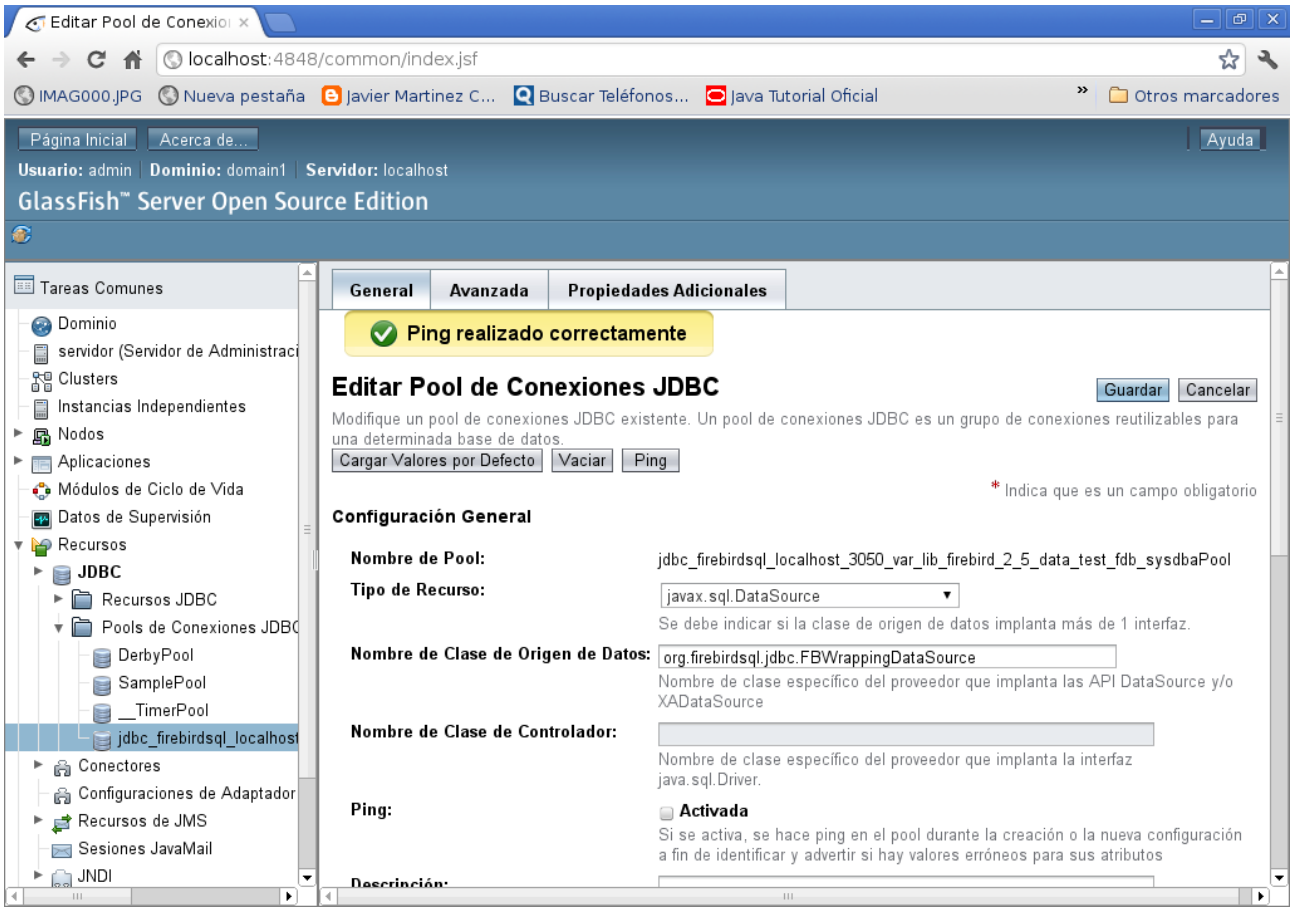
Aquí puede observarse que GF continua utilizando la clase org.firebirdsql.jdbc.FBDriver en vez de org.firebirdsql.jdbc.FBWrappingDataSource :



Una vez cambiada la clase, se debe presonar el boton Guardar para persistir los cambios en el server:



Por último, podemos presionar el boton Ping para chequear el pool de conexiones y verificar que el mismo funciona Ok y no emite mensaje de error:

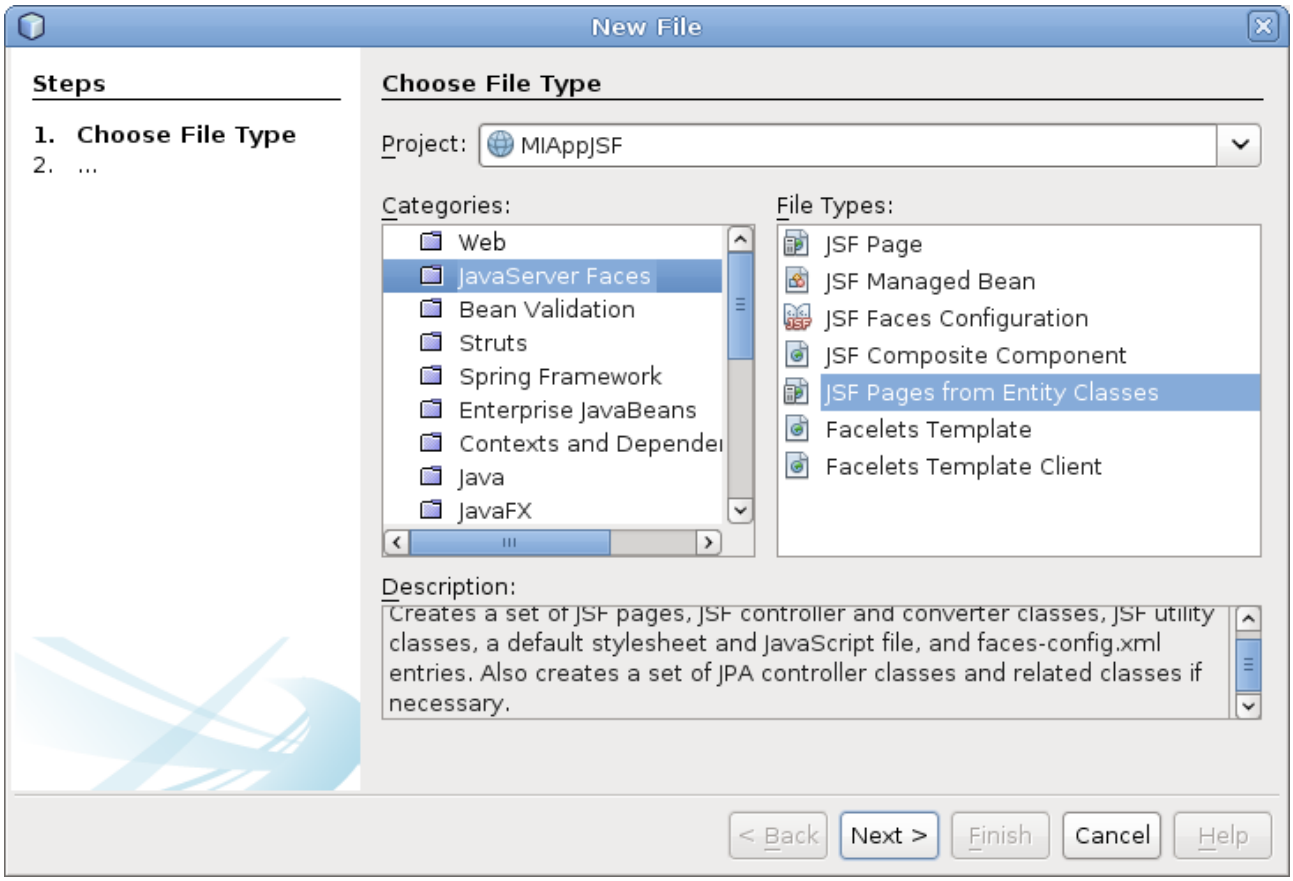


Ejecutar el proyecto nuevamente para verificar que el proceso de deployment funciona correctarmente.

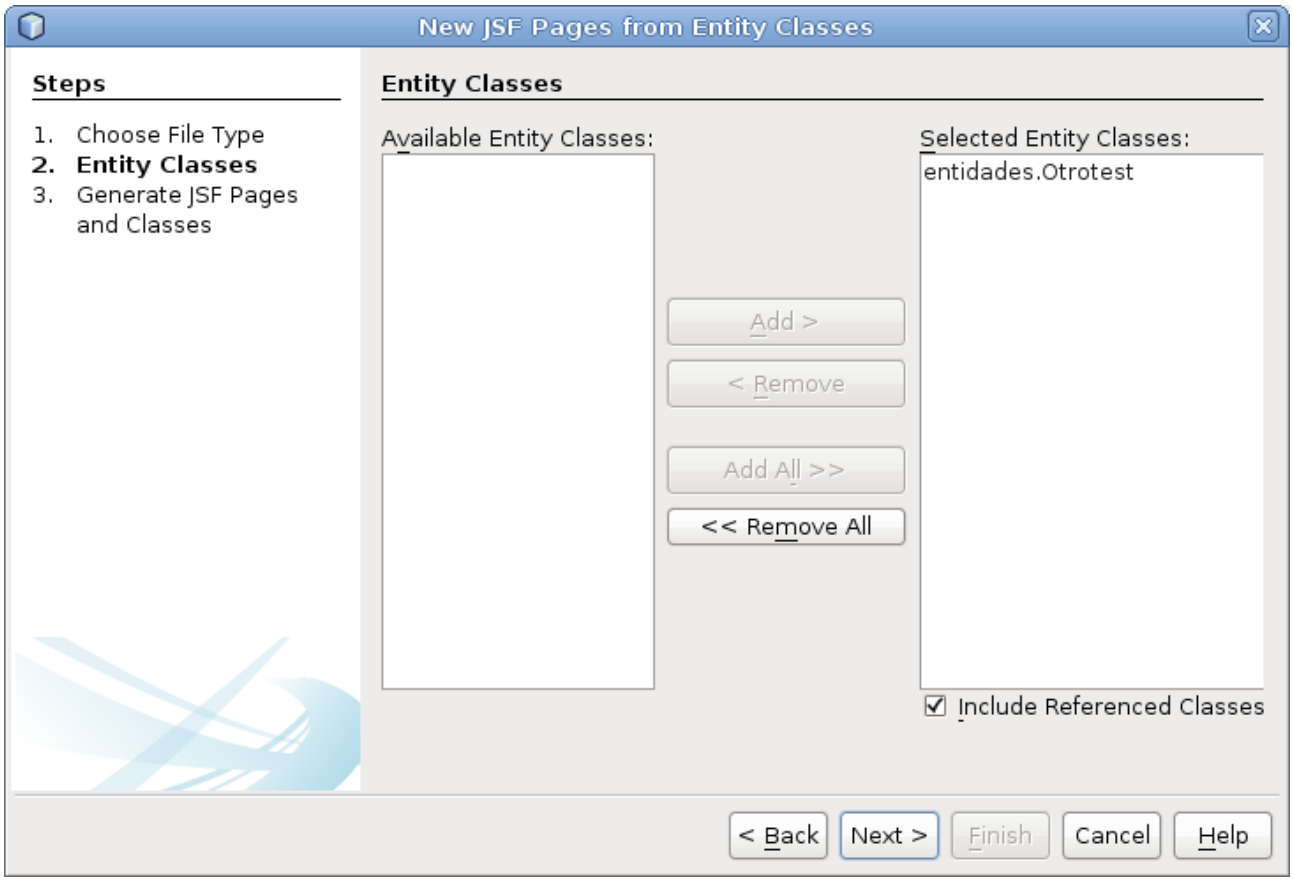
Hacer BACKUP.

Creación de Páginas JSF a partir de Entidades

Podemos crear paginas JSF a partir de las entidades que ya tenemos creadas en el proyecto, presionar boton crear, seleccionar JavaServer Faces / JSF Pages from Entity Classes:



Seleccione las entidades sobre las cuales pretende generar las paginas JSF, presione boton Add > , luego presione boton Next:



En dialogo Generate JSF Pages and Classes:

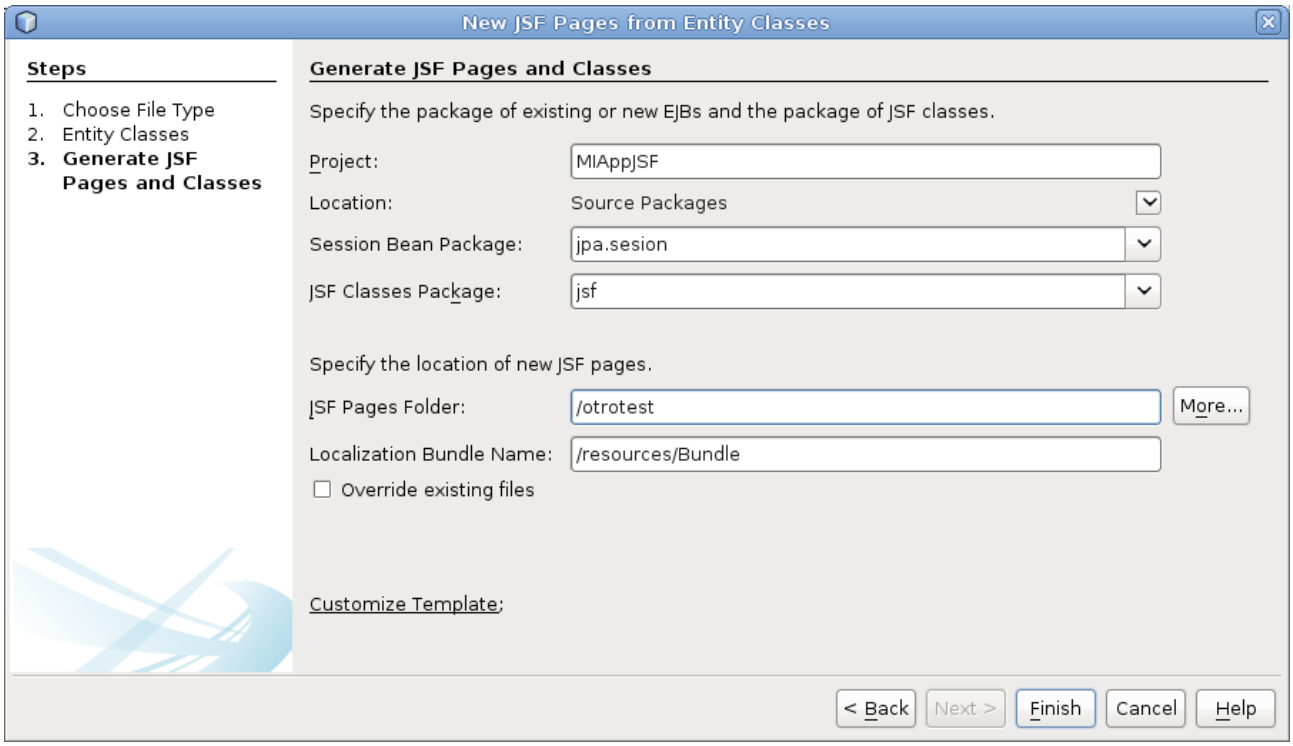
en Session Bean Package tipear "jpa.sesion"

en JSF Classes Package tipear "jsf"

en JSF Pages Folder no es necesario tipear nada, pues el wizard creara una carpeta llamada otrotest (coincidente con el nombre de entidad) para depositar allí todas las paginas generadas para manipular los datos de dicha entidad (en este caso, se creará una carpeta otrotest y dentro de la misma otra carpeta otrotest y dentro de la misma se encontrarán las paginas JSF generadas)

en Localization Bundle Name tipear "/resources/Bundle" (permite crear una carpeta llamada resources y dentro del mismo se generará el archivo Bundle.properties, caso contrario, Bundle.properties quedará creado en el package por defecto del proyecto o carpeta principal) . Aquí es conveniente crear un bundle distinto para cada pagina JSF de forma tal que cada pagina sea “customizable” a partir de este archivo.

Hacer click en boton Finish:



Observe los packages jsf, jpa.sesion, jsf.util, resources; las paginas generadas y las clases asociadas a las mismas.

Para cada clase entidad, el wizard generara lo siguiente:

Un stateless session bean

Una clase llamada AbstractFacade.java que contiene toda la lógica de negocio para crear, recuperar, modificar y destruir instancias de entidades (filas) y cada uno de los stateless session bean creados derivan de esta clase.

Un JSF session-scoped, managed bean

Un directorio (con el nombre de la entidad en cuestión) conteniendo 4 archivos Facelets para hacer CRUD (Create, Read, Update, Delete) sobre la entidad en cuestión y los mismos se llamarán respectivamente: Create.xhtml, Edit.xhtml, List.xhtml y View.xhtml

Creara también clases de utilería utilizadas por los JSF managed beans, llamadas JsfUtil y PaginationHelper

Un conjunto de propiedades que permitirá emitir mensajes acorde con el Location del usuario y los mismos tendrán su correspondiente entrada dentro del archivo de configuracion de JSF (faces-config.xml), el cual será creado si previamente no existía.

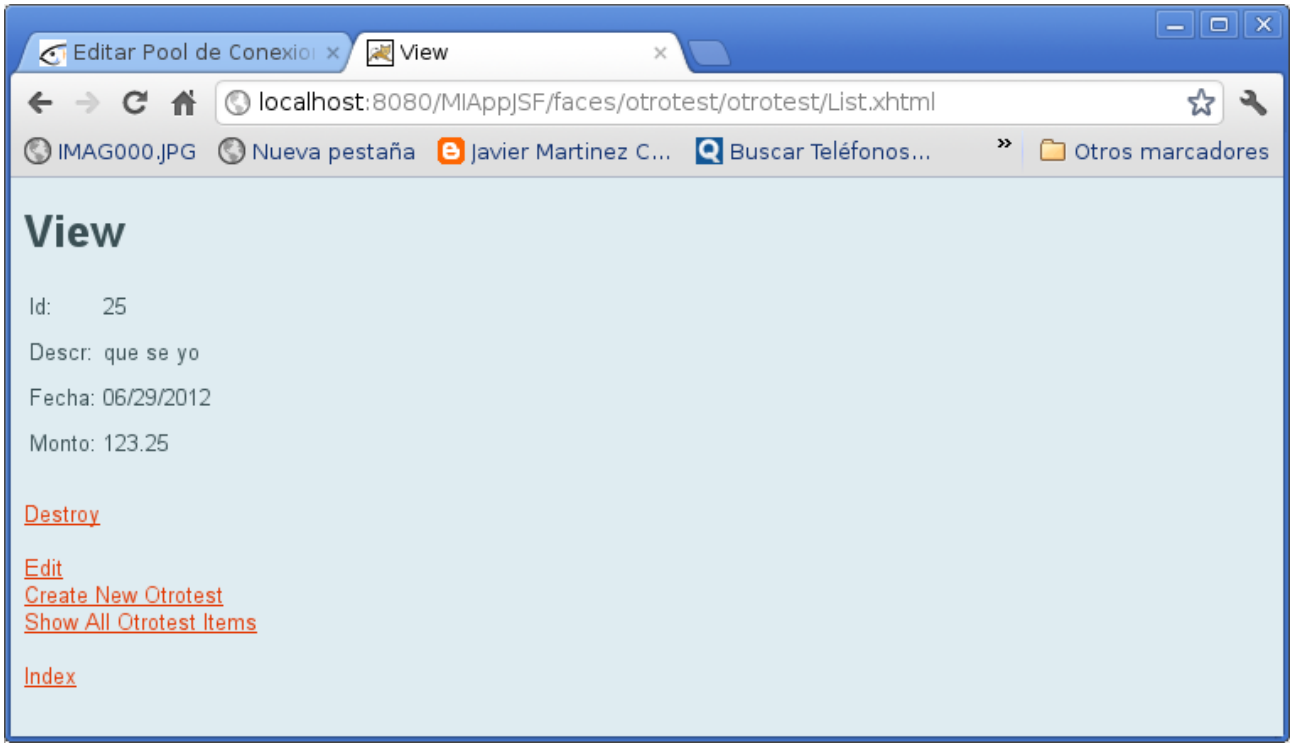
Archivos web auxiliares, un archivo stylesheet para “customizar” la visualización de componentes y un archivo template de Facelets

Luego de ejecutar el wizard de creación de paginas JSF a partir de entidades, si el proyecto no puede ser deployed en GF, verifique nuevamente el archivo glassfish-resources.xml, propiedad datasource-classname deber estar como:

```
datasource-classname="org.firebirdsql.jdbc.FBWrappingDataSource"
```

modifique el archivo, y corra nuevamente la aplicación (para mayor información, remítase al punto anterior de este documento).

Ejecute el proyecto y observe cómo funcionan las distintas paginas JSF que permiten hacer operaciones CRUD sobre tabla otrotest (en este caso):


Editar Pool de Conexio
View
localhost:8080/MIAppJSF/faces/otrotest/otrotest/List.xhtml
IMAG000.JPG
Nueva pestaña
Javier Martinez C...
Buscar Teléfonos...
Otros marcadores
View
Id: 25
Descr: que se yo
Fecha: 06/29/2012
Monto: 123.25
Destroy
Edit
Create New Otrotest
Show All Otrotest Items
Index

Desde FR puede hacer consultas a la base de datos FB en cuestión para verificar los cambios realizados desde la aplicación.

Utilizar Componentes PrimeFaces en Páginas JSF

Observe la página http://www.primefaces.org/showcase/ui/home.jsf allí podrá ver cada uno de los componentes de PrimeFaces, su aspecto visual, el tag xml asociado y un ejemplo de EJB asociado con dicho componente. Esta página es una fuente escencial de consulta para implementar componentes PrimeFaces junto con el manual de esta librería que permite hacer interfases gráficas web de mayor valor agregado utilizando J2EE.

Seleccione el componente Calendar, verifque el código JSF (CalendarBasic.xhtml) y el código EJB (CalendarBean.java) relacionado con el componente y razone su funcionamiento.

Edite el archivo Edit.xhtml (pagina JSF de edición) y observe como se ingresa la fecha:

```
                    <h:inputText id="fecha"
value="#{otrotestController.selected.fecha}"
title="#{bundle.EditOtrotestTitle_fecha}" >
                        <f:convertDateTime pattern="MM/dd/yyyy" />
                    </h:inputText>
```

ahora vamos a hacer que se edite usando un componente PrimeFaces y la fecha en formato dd/MM/yy ; reemplace el tag inputText anterior por:

```
<p:calendar    value="#{otrotestController.selected.fecha}"    id="fecha"
showOn="button" >
    <f:convertDateTime pattern="dd/MM/yy" />
</p:calendar>
```

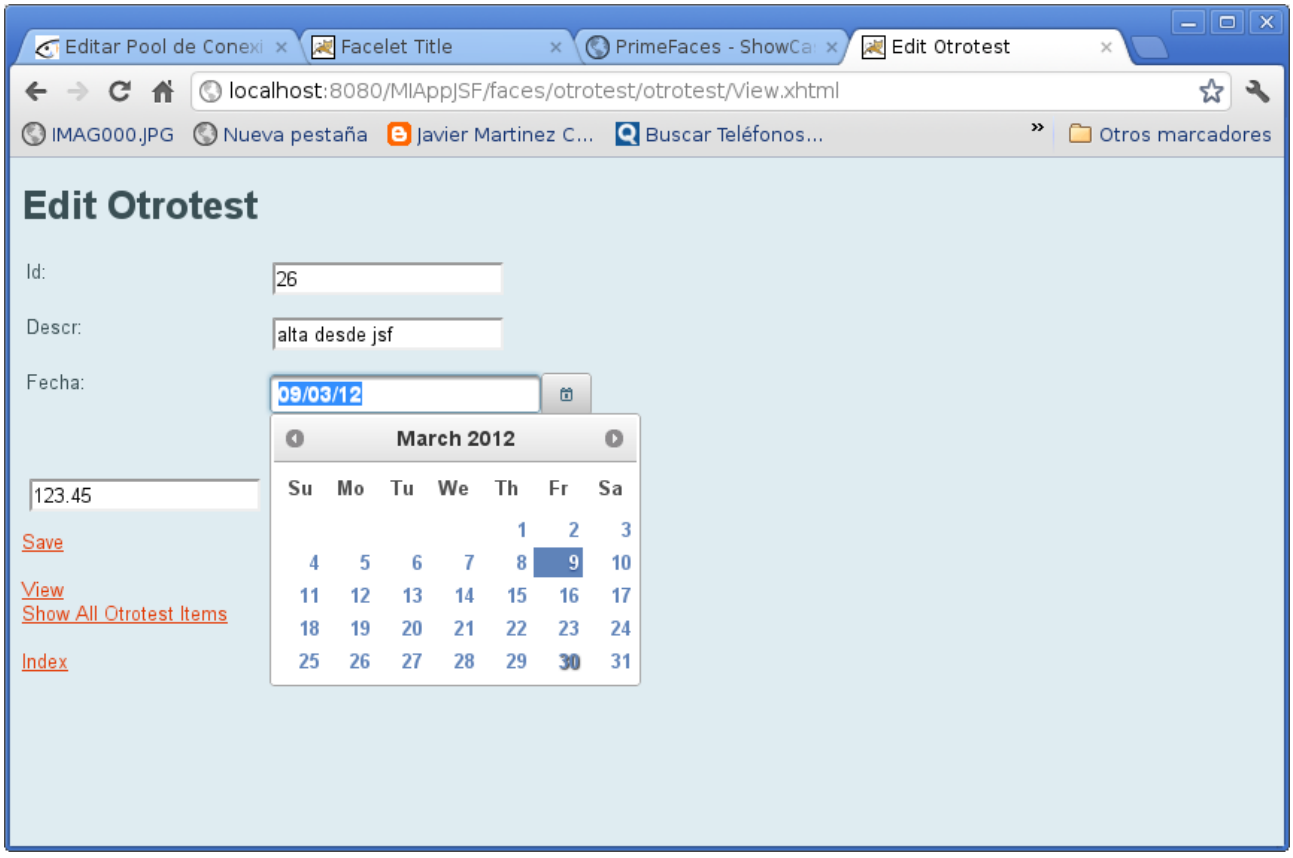
Agregue en tag <html>, el namespace correspondiente a PrimeFaces:

```
xmlns:p="http://primefaces.org/ui"
```

el tag <html> podría tener la siguiente forma:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui">
```

Grabe la página y ejecute el proyecto. Observe la página de edición, utilice componente el PrimeFaces Calendar para facilitar el ingreso de fechas:



Cambie la fecha y grabe registro. Observe que la visualización se realiza en formato MM/dd/yyyy. Edite View.xhtml y cambie el formato de la fecha para que se visualice de la forma apropiada.

Hacer BACKUP.

Conclusiones

Habiendo llegado satisfactoriamente al final de esta experiencia, Ud. esta en condiciones de continuar agregando componentes PrimeFaces para embellecer las páginas de su aplicación web J2EE. Se ha encontrado un problema entre los wizards de NB 7.1.1 y el driver JDBC JayBird 2.1.6 (el cual esta estable desde hace años), aun no esta resuelto el problema de forma tal de evitar su ocurrencia, no obstante, esta identificado y permite modificar el proyecto para corregir el bug. El precio que se paga en esta corrección bien vale la pena en comparación con no utilizar el wizard de NB y realizar todo el trabajo 100% en forma manual.

Se considera esta plataforma como una forma viable de desarrollo J2EE, el cual tiene su complejidad pero cuenta con un mercado atractivo para el desarrollo de aplicaciones empresariales. Espero que este documento haya sido de vuestro agrado y le permita ingresar al mundo J2EE.

Atte. Guillermo Cherencio
Práctica Profesional
ISFT N° 189