

1. Instalaciones necesarias

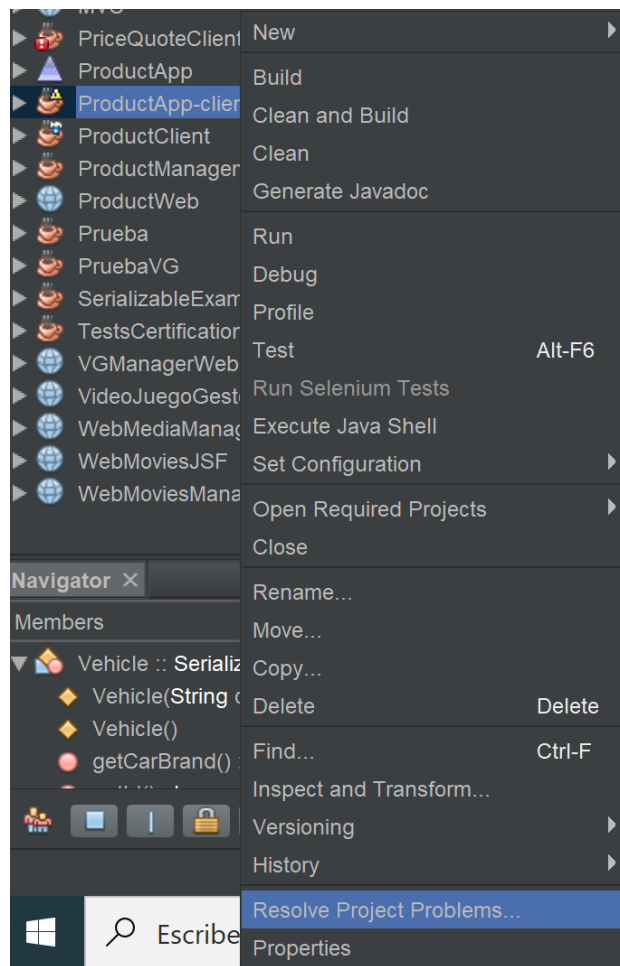
- Netbeans 14 <https://netbeans.apache.org/download/index.html>
- Jdk 1.8 <https://www.oracle.com/es/java/technologies/javase/javase8-archive-downloads.html>
- Servidor glassfish 5.1
<https://www.eclipse.org/downloads/download.php?file=/glassfish/glassfish-5.1.0.zip>

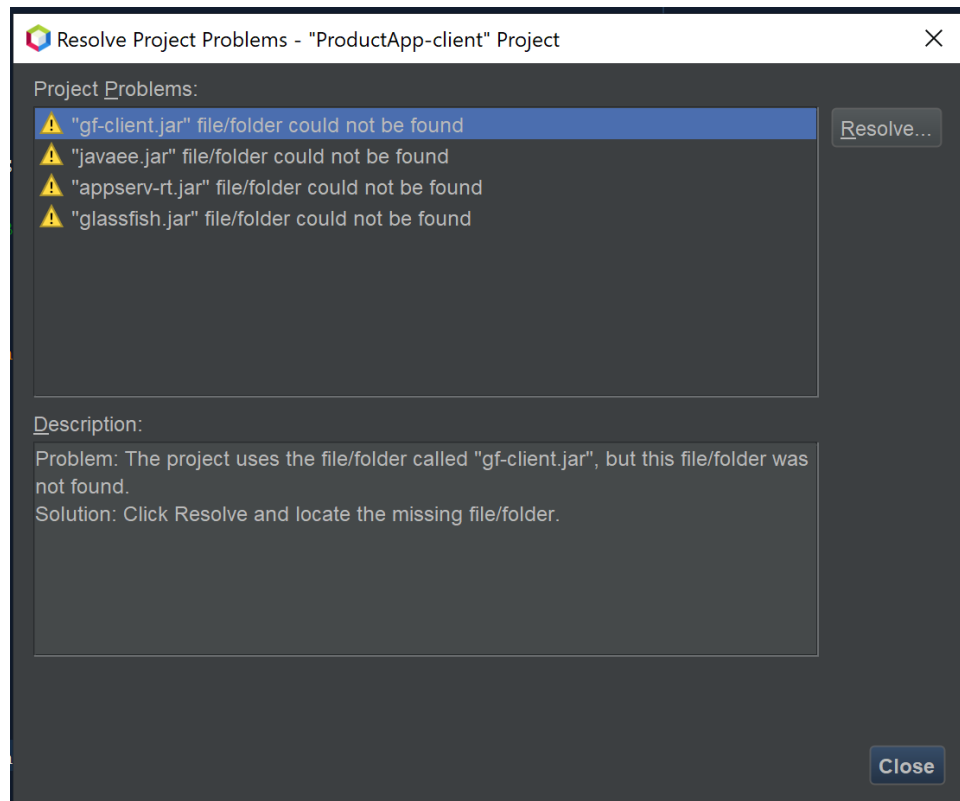
Este servidor presenta un bug, por lo que una vez descargado él .zip y descomprimido en una ruta de vuestro ordenador, hay que sustituir el fichero: `./glassfish5/glassfish/modules/bean-validator-cdi.jar`

Por el que está en la carpeta recursos.

2. Netbeans

1. Abrimos los proyectos.
 - 1.1. Nos aparecerán errores en dos de los proyectos puesto que nos faltan librerías.
Le damos click derecho en el nombre del proyecto y Resolve Project Problems

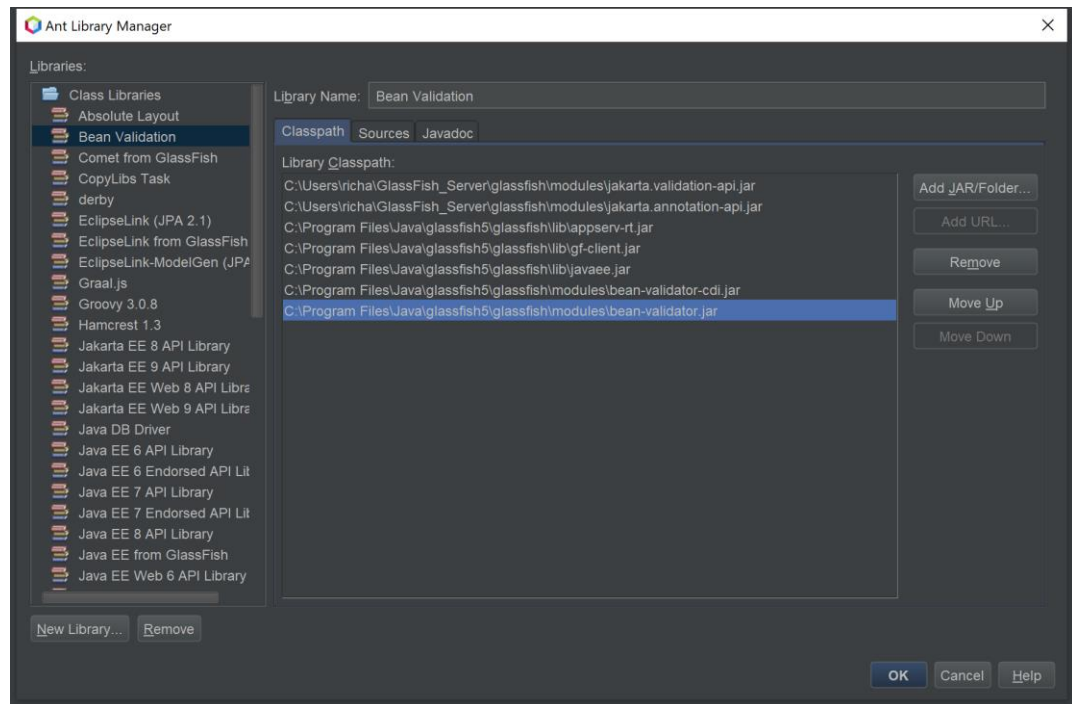




Le damos uno por uno a resolve y buscamos la libreria correspondiente en las rutas:

```
..\glassfish5\glassfish\lib\gf-client.jar  
..\glassfish5\glassfish\lib\javaee.jar  
..\glassfish5\glassfish\lib\appserv-rt.jar  
..\Users\<name>\GlassFish_Server\glassfish\modules\glassfish.jar
```

Para el otro proyecto hacemos la misma operación y creamos la libreria "Bean Validation" con rutas parecidas a las que aparecen en la imagen pero adaptadas a tu equipo:

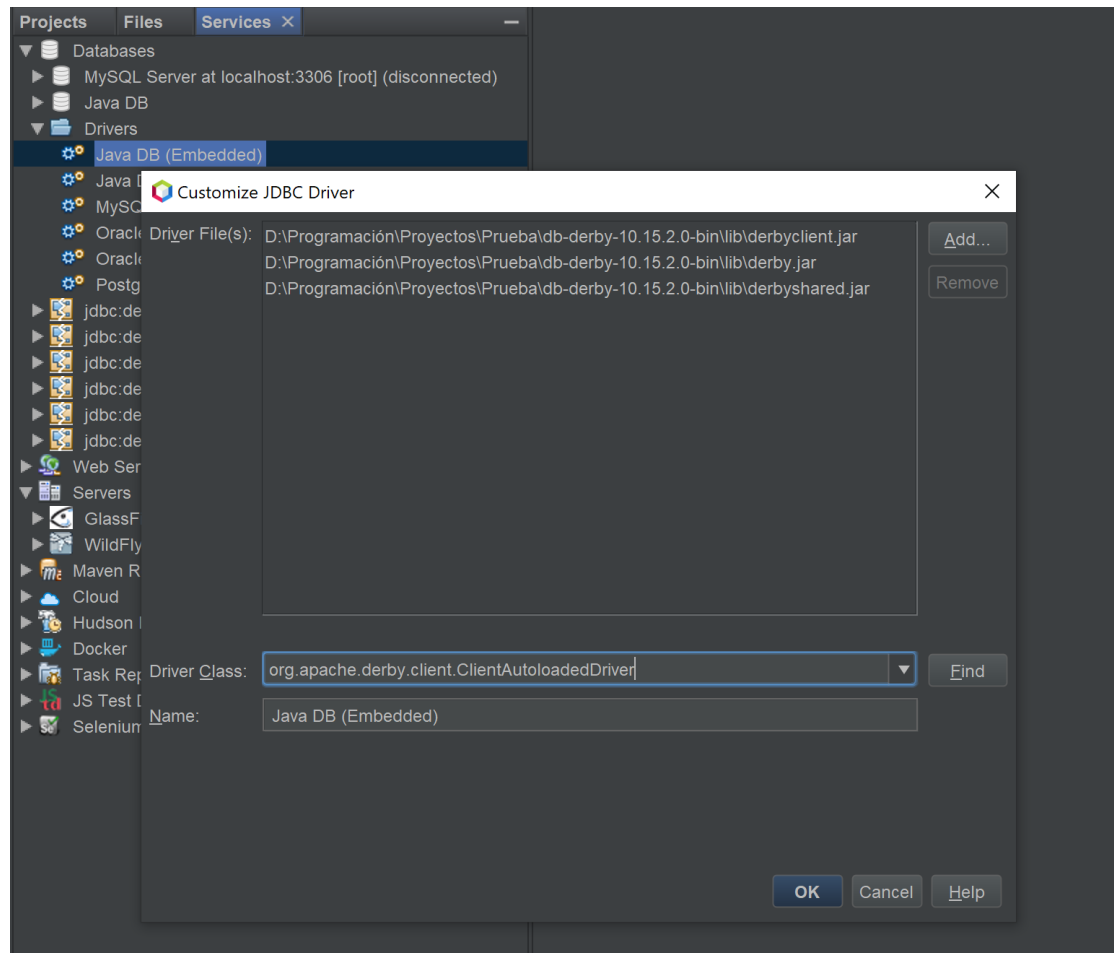


2. Configuración del servidor y BBDD.

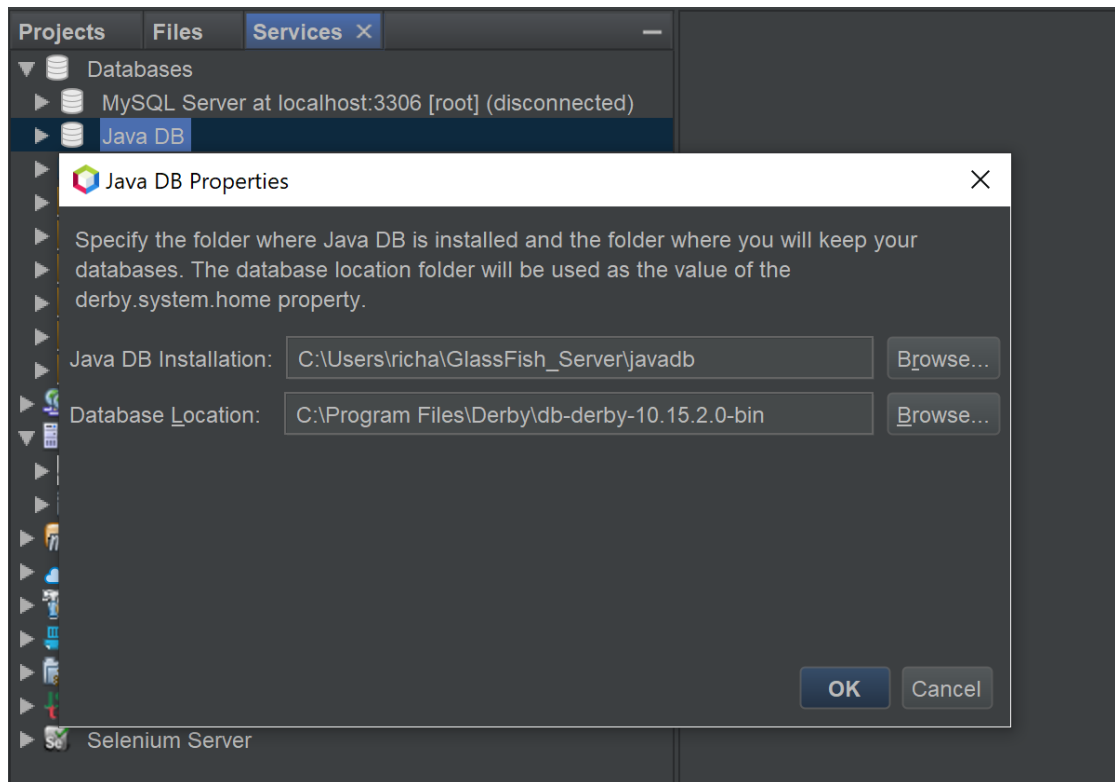
Una vez importados los proyectos anteriores, hay que configurar el servidor y la BBDD.

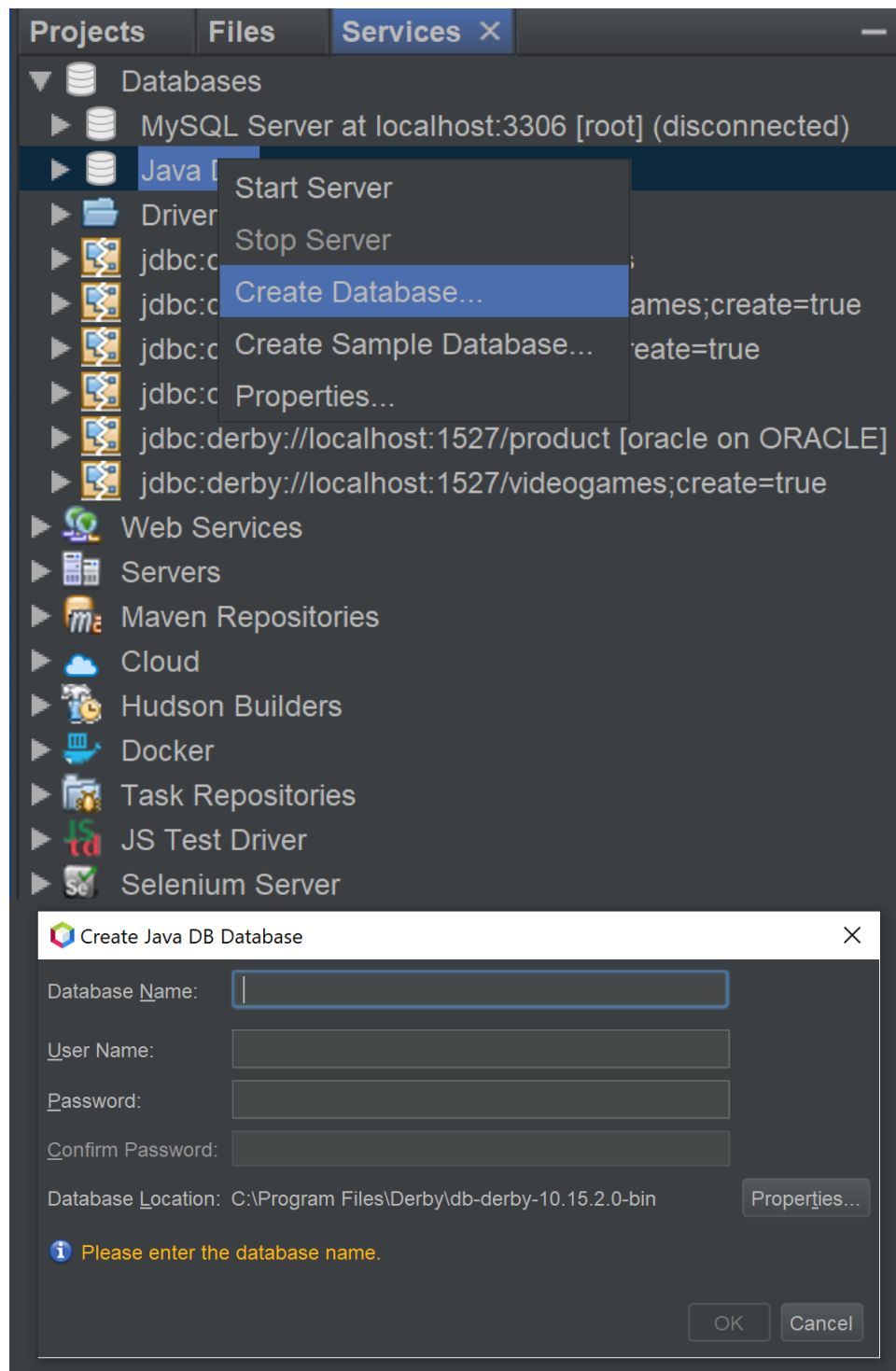
2.1 Java Databases:

Drivers:



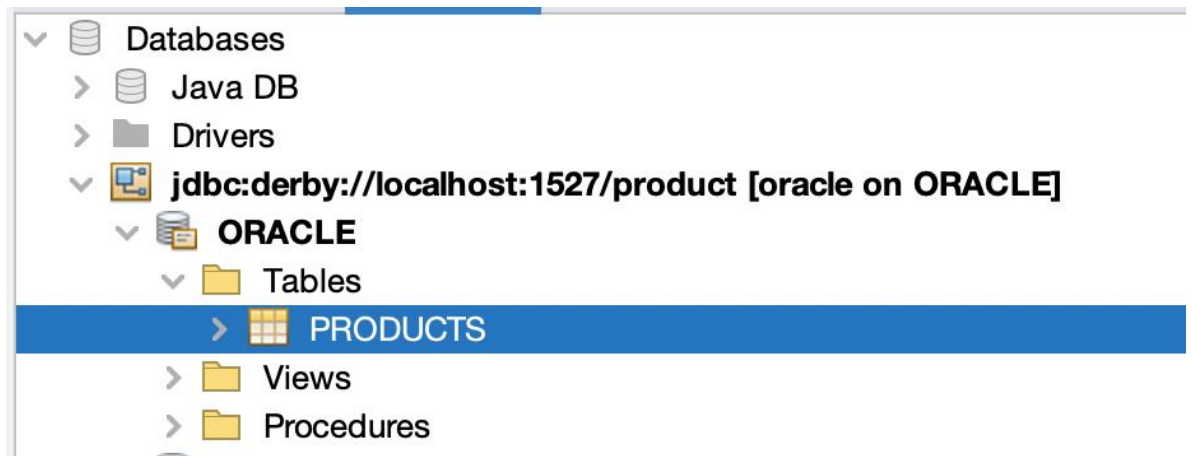
Java DB Properties:





Lo configuramos exactamente igual que en el libro de Oracle (página 15)
Una vez, creado, cargamos el fichero “createDBObject.sql” que lo tenéis la carpeta recursos:

Y ya tendremos la tabla "PRODUCTS"

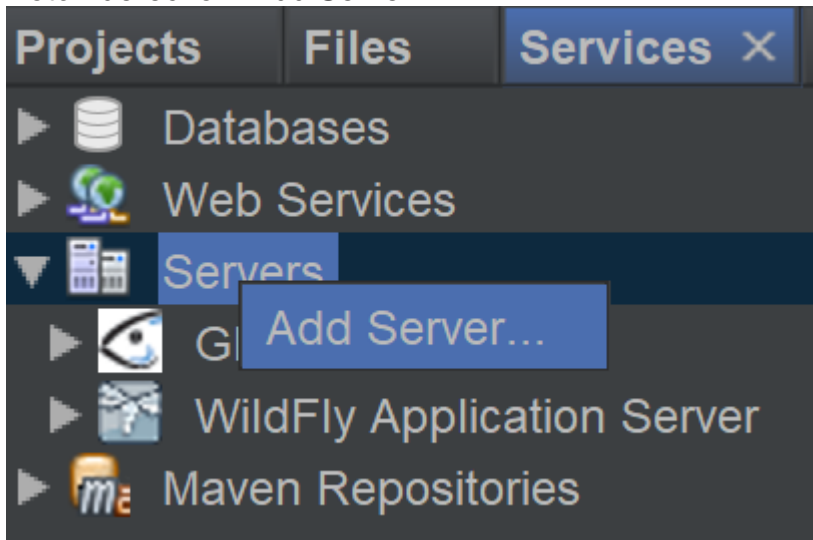


2.2 Servidor:

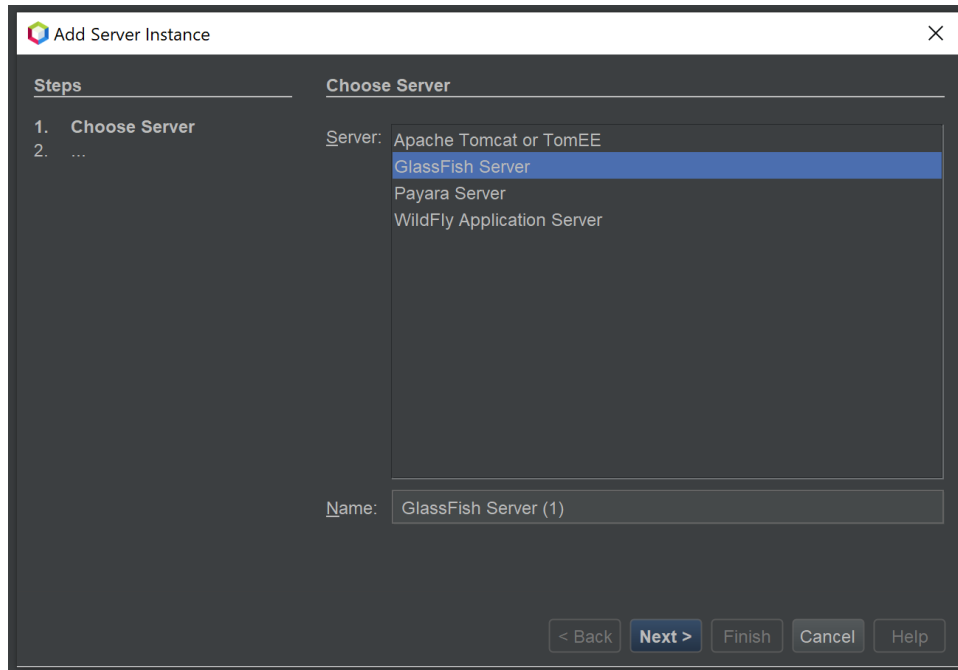
En NetBeans:

Pestaña Services > Servers >

Botón derecho > Add Server...

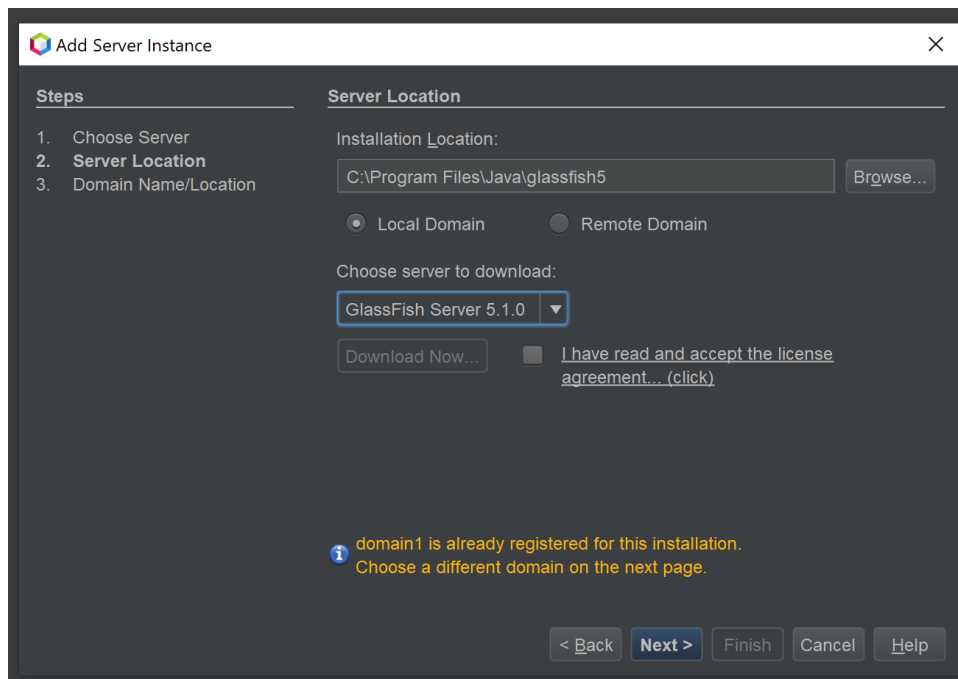


Elegimos GlassFish Server:



- Nos solicita donde tenemos instalado el glassfish que hemos descomprimido antes (también tendremos que elegir java 1.8 que hemos descargado antes en algún punto de esta instalación).

Seleccionamos Local Domain y Next



- La ventana de Domain Location la dejamos tal cual, os saldrá algo así y pulsamos Finish (Ignorar el error de la imagen se debe a que en mi caso ya existe el dominio y no lo puede volver a crear):

Add Server Instance

Steps

1. Choose Server
2. Server Location
3. **Domain Name/Location**

Domain Location

Domain:

Host: ☒ Loopback

DAS Port: HTTP Port: ☐ Default

Target:

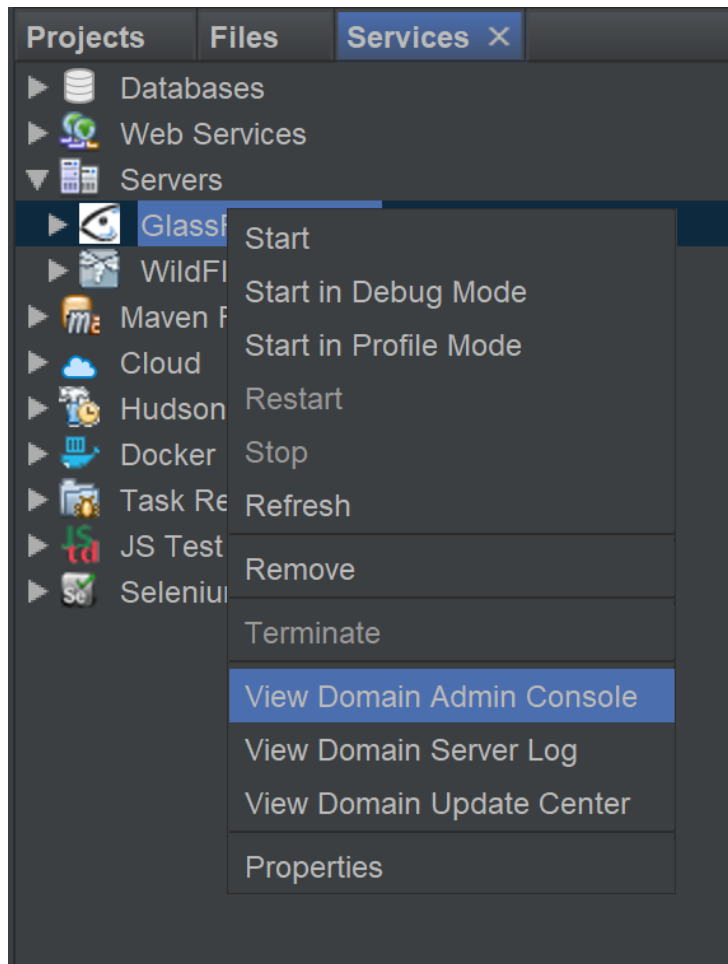
User Name:

Password:

Domain domain1 is already registered for this installation.

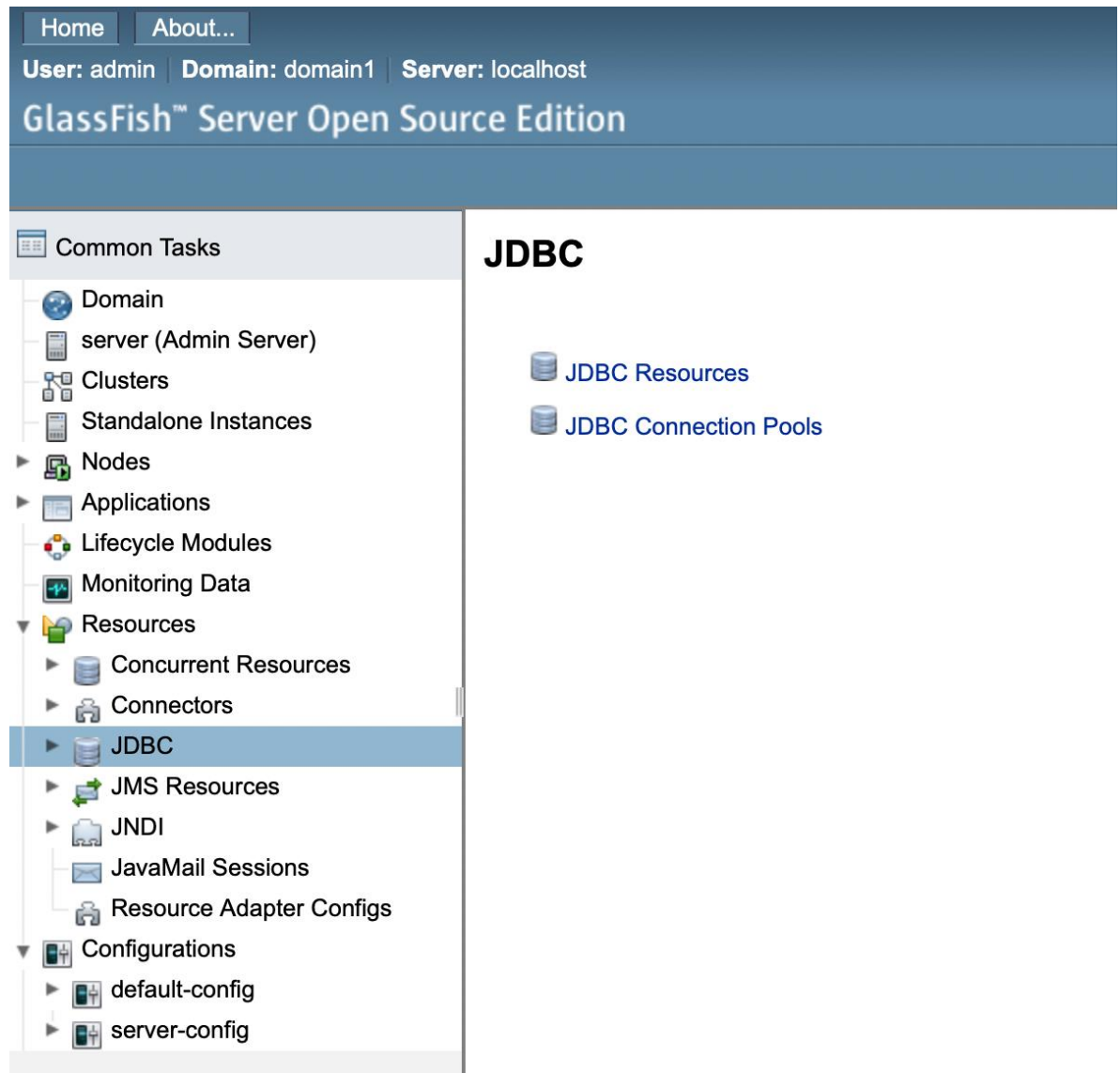
< Back Next > Finish Cancel Help

Una vez creado el servidor, lo iniciamos y con el botón derecho abrimos la consola.



En la consola:

Tenemos que ir a Resources – JDBC:





Y crear un JDBC Connection Pools y un JDBC Resources que irá asociado a dicho pool.

1. JDBC Connection Pools

-New...

JDBC Connection Pools

To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API. Before an application can access a database, it must get a connection.

Pools (4)				
  New... Delete				
Select	Pool Name	Resource Type	Classname	Description
<input type="checkbox"/>	DerbyPool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDriver	
<input type="checkbox"/>	SamplePool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource	
<input type="checkbox"/>	TimerPool	javax.sql.XADataSource	org.apache.derby.jdbc.EmbeddedXADataSource	
<input type="checkbox"/>	productosPool	javax.sql.DataSource	org.apache.derby.jdbc.ClientDataSource40	

Pool Name: productosPool
Resource Type: javax.sql.DataSource
Database Driver Vendor: Derby

New JDBC Connection Pool (Step 1 of 2)

Identify the general settings for the connection pool.

General Settings

Pool Name: * productosPool

Resource Type: javax.sql.DataSource
Must be specified if the datasource class implements more than 1 of the interface.

Database Driver Vendor: Derby
Select or enter a database driver vendor

Introspect: ☐
If enabled, data source or driver implementation class names will enable introspection.

Next.

New JDBC Connection Pool (Step 2 of 2)

Identify the general settings for the connection pool. Datasource Classname or Driver Classname must be specified for the connection pool.

General Settings

Pool Name: productosPool

Resource Type: javax.sql.DataSource

Database Driver Vendor: Derby

Datasource Classname: org.apache.derby.jdbc.ClientDataSource40
Select or enter vendor-specific classname that implements the DataSource and/or XADataSource APIs

Driver Classname:
Select or enter vendor-specific classname that implements the java.sql.Driver interface.

Ping: ☐
When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes

Description:

Datasource Classname: org.apache.derby.jdbc.ClientDataSource40

ServerName: localhost

Por defecto os saldrá algo así (Podéis editarlas e ir poniendo las que os he puesto, el resto nos dan igual):

Additional Properties (18)		
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="Add Property"/> <input type="button" value="Delete Properties"/>
Select	Name	Value
<input type="checkbox"/>	TraceFileAppend	false
<input type="checkbox"/>	SecurityMechanism	4
<input type="checkbox"/>	ConnectionAttributes	
<input type="checkbox"/>	Description	
<input type="checkbox"/>	TraceDirectory	
<input type="checkbox"/>	User	APP
<input type="checkbox"/>	DatabaseName	
<input type="checkbox"/>	Ssl	off
<input type="checkbox"/>	RetrieveMessageText	true
<input type="checkbox"/>	DataSourceName	
<input type="checkbox"/>	LoginTimeout	0
<input type="checkbox"/>	ShutdownDatabase	
<input type="checkbox"/>	TraceFile	
<input type="checkbox"/>	ServerName	localhost
<input type="checkbox"/>	CreateDatabase	
<input type="checkbox"/>	TraceLevel	-1
<input type="checkbox"/>	PortNumber	1527
<input type="checkbox"/>	Password	

Guardáis y podéis probar con el botón “Ping” que funciona correctamente.

General	Advanced	Additional Properties
---------	----------	-----------------------

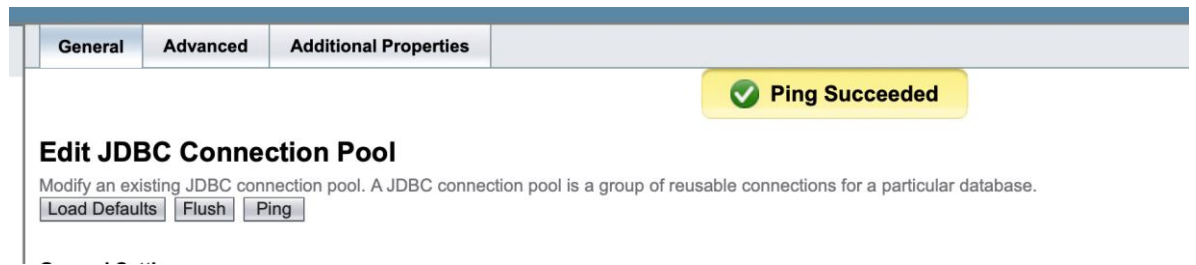
Edit JDBC Connection Pool

Modify an existing JDBC connection pool. A JDBC connection pool is a group

General Settings

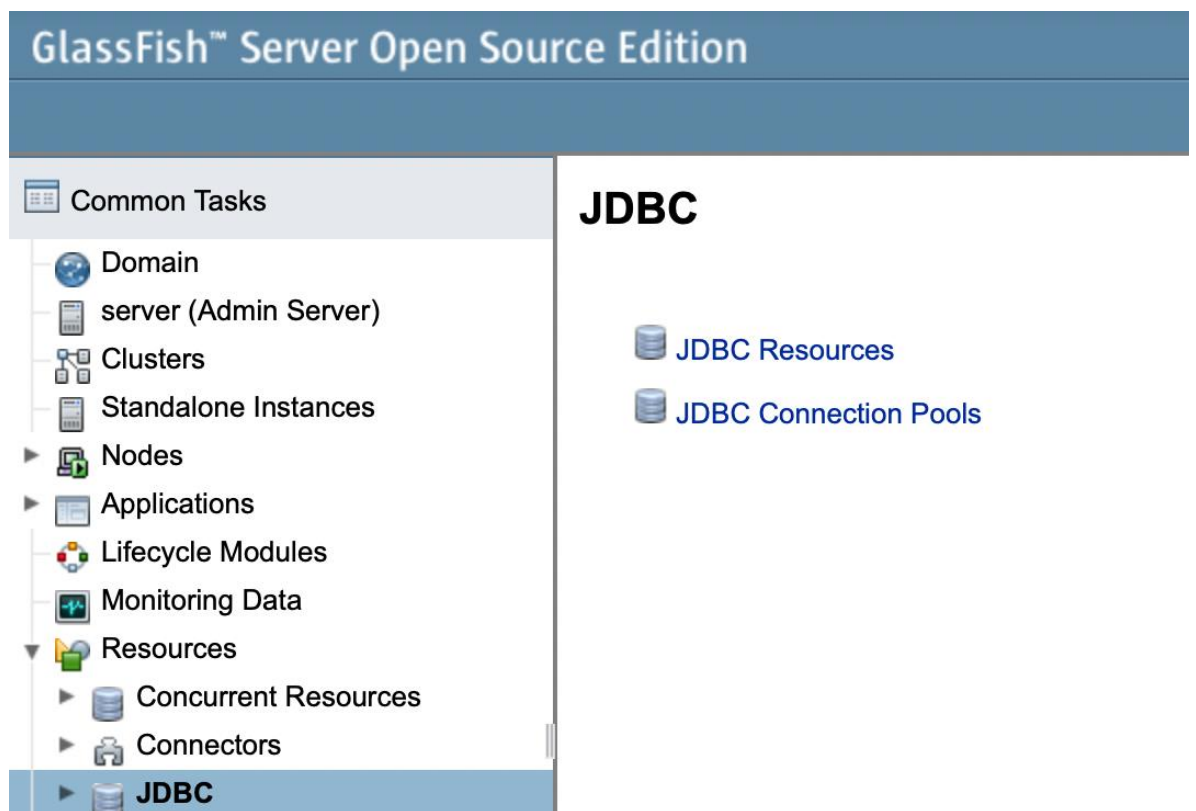
Pool Name: productosPool

Resource Type:



Si no os sale eso, revisar que estén las propiedades anteriores bien puestas.

2. JDBC Resources



Elegimos JDBC Resources

New...

JNDI Name: jdbc/productDB

PoolName: productosPool (el creado antes)

New JDBC Resource

OKCancel

Specify a unique JNDI name that identifies the JDBC resource you want to create. The name must contain only alphanumeric, underscore, dash, or dot characters.

JNDI Name: *

jdbc/productDB

Pool Name:

productosPool

Use the [JDBC Connection Pools](#) page to create new pools

Description:

Status:

☒

Additional Properties (0)

Add PropertyDelete Properties

Select	Name	Value	Description
No items found.			

Y pulsamos OK.

Con estos puntos ya podríamos probar las primeras prácticas.

Para el apartado JMS, habría que configurar lo siguiente:

3. JMS Resources

[Home](#) [About...](#)

User: admin | Domain: domain1 | Server: localhost

GlassFish™ Server Open Source Edition

Common Tasks

Domain

server (Admin Server)

Clusters

Standalone Instances

▶ Nodes

▶ Applications

Lifecycle Modules

Monitoring Data

▼ Resources

▶ Concurrent Resources

▶ Connectors

▶ JDBC

▶ JMS Resources

▶ JNDI

JavaMail Sessions

Resource Adapter Configs

JMS Resources

Connection Factories

Destination Resources

- Connection Factories

Edit JMS Connection Factory

Editing a Java Message Service (JMS) connection factory also modifies the associated connector connection pool and connector resource.

Load Defaults

General Settings

JNDI Name:

jms/productQueueFactory

Logical JNDI Name:

Resource Type:

javax.jms.QueueConnectionFactory

Description:

jms/productQueueFactory

Status:

☒

- Destination Resources

JNDI Name:	jms/productQueue
Physical Destination Name *	productQueue
	<small>Destination name in the Message Queue broker. If the destination does not exist, it will be created automatically when needed.</small>
Resource Type: *	javax.jms.Queue
Deployment Order:	100
	<small>Specifies the loading order of the resource at server startup. Lower numbers are loaded first.</small>
Description:	Cola para la practica del curso
Status:	<input checked="" type="checkbox"/>

Si todo va bien, cuando hagáis un deploy en las aplicaciones, os saldrán aquí:

