

Guia Pool de Conexiones

Pool de conexiones en Glassfish (desde Glassfish y desde NetBeans)

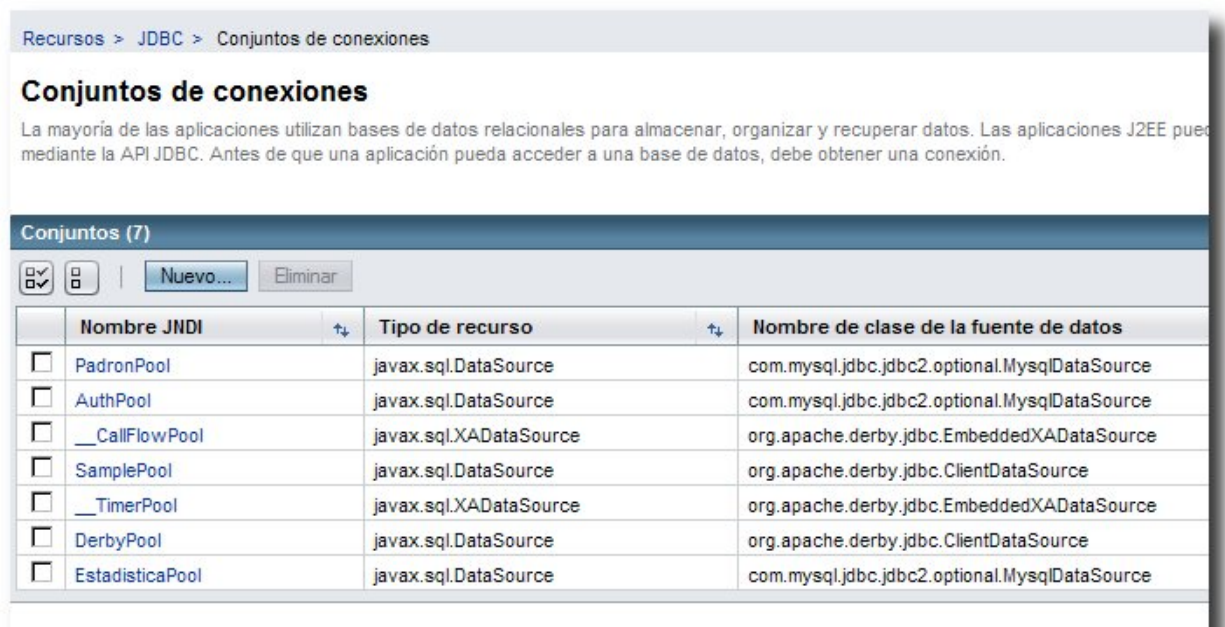
Glassfish tiene una manera peculiar de tratar los recursos JDBC: Tiene un pool de conexiones, y después un JDBC Resources. Supongo que es porque un pool de conexiones puede tener varios JDBC Resource. Los demás Servidores de Aplicaciones que he visto usan unicamente un JDBC Resource, y en ese mismo se configura la configuración del pool.

Lo que veremos ahora es como se crea un pool de conexion + jdbc resource en Glassfish. Existen 2 formas: 1.Directamente desde Glassfish, y 2. Desde NetBeans.

1. Desde el mismo Glassfish

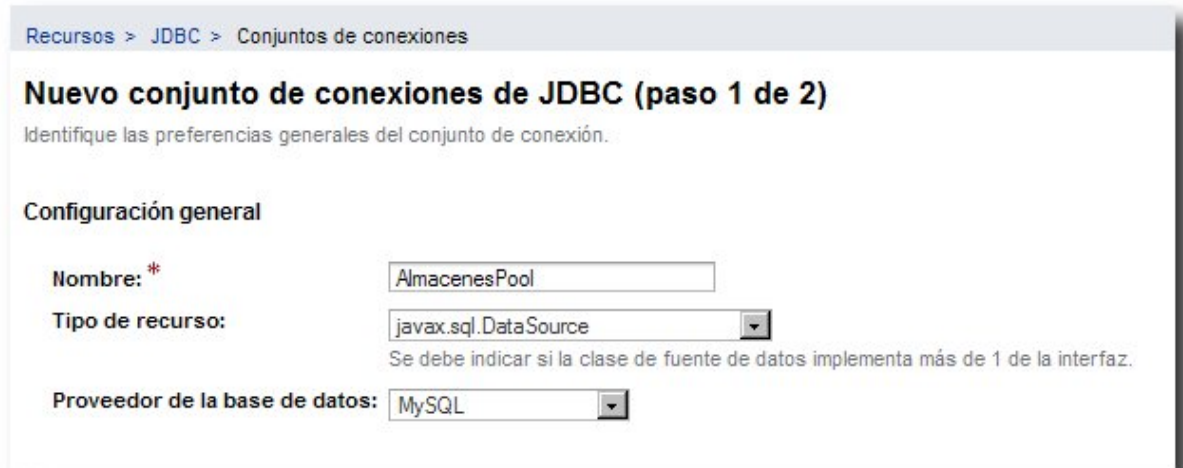
Creando el Pool de conexiones

1. Entramos a la consola de glassfish. Si está en nuestro equipo local, ingresar a <http://localhost:4848> Por defecto, el usuario es **admin** y la contraseña es **adminadmin**
2. En el panel izquierdo de la consola de glassfish, seleccionamos Resources > JDBC > Pool Connections:



3. Hacemos clic en el botón "New"

4. En el asistente para crear un nuevo pool de conexiones, escribimos el nombre del pool. Cualquier nombre es válido, pero recomendable que solo esté compuesto por letras y números y sin espacios. También seleccionamos el tipo de recurso, que generalmente es `java.sql.DataSource`, y por último seleccionamos cuál es el proveedor de nuestra base de datos. En mi caso usaré MySQL.



Recursos > JDBC > Conjuntos de conexiones

Nuevo conjunto de conexiones de JDBC (paso 1 de 2)

Identifique las preferencias generales del conjunto de conexión.

Configuración general

Nombre: *

Tipo de recurso: ▼

Se debe indicar si la clase de fuente de datos implementa más de 1 de la interfaz.

Proveedor de la base de datos: ▼

5. Clic en "Siguiente"
6. Luego, nos mostrará todas las propiedades de nuestra conexión a la base de datos. Generalmente muestra realmente todas las propiedades. Pero para este ejemplo solo usaré los que me interesa: username, password, databasename y Servername. Además activamos los checks "Permitir llamadores que no sean de componentes:" y "Conexiones no transaccionales:". Ver imagen en siguiente pagina

Validación de conexión

Validación de conexión:

☐ **Necesaria**

Validar conexiones; permite al servidor volver a conectar en caso de fallo

Método de validación:

auto-commit

Nombre de la tabla:

Si se selecciona la validación de tabla, especifique un nombre de tabla; éste sólo debe contener caracteres alfabéticos, guiones o puntos

Ante cualquier fallo:

☐ **Cerrar todas las conexiones**

Cierre todas las conexiones y vuelva a conectar en caso de fallo, o vuelva a conectar solo las que fallaron

Permitir llamadores que no sean de componentes:

☒ **Activado**

Permita que los llamadores que no sean de componentes como, ejemplo, los filtros de seguridad, accedan a la base de datos

Transacción

Conexiones no transaccionales: ☒ **Activado**

Devuelve conexiones que no son transaccionales

Aislamiento de la transacción:

Si no especificado, utilizar nivel predeterminado para controlador JDBC

Nivel de aislamiento:

☐ **Garantizado**

Todas las conexiones utilizan el mismo nivel de aislamiento; requiere aislamiento de transacción

Propiedades adicionales (4)



Agregar propiedad

Eliminar propiedades

	Nombre	Valor
<input type="checkbox"/>	DatabaseName	almacenes
<input type="checkbox"/>	Password	almacenes
<input type="checkbox"/>	ServerName	dsilva1
<input type="checkbox"/>	User	almacenes

Clic en Finish

Podemos hacer clic en el botón "ping" ("sondeo" en español) para probar si la conexión ha sido exitosa.

Si nos manda error, ya que la clase controladora de JDBC no existe dentro de Glassfish, entonces necesitamos agregar el .jar correspondiente en \$GLASSFISH_HOME/domains/domain/lib

Ya tenemos nuestro pool de conexiones. Ahora nos falta, el recurso JDBC. Esto es más fácil

Creando el recurso JDBC

1. Seleccionamos del panel lateral izquierdo: Resources > JDBC > JDBC Resources:



Hacemos clic en el botón "New"

2. Escribimos el nombre de nuestro recurso JDBC en formato JNDI. En mi ejemplo será jdbc/almacenes. Luego seleccionamos con cual pool de conexiones se asociará nuestro recurso JDBC.

Nombre JNDI: * jdbc/almacenes

Nombre de conjunto: * AlmacenesPool

Use la página [Conjunto de conexiones de JDBC](#) para crear nuevos conjuntos.

Descripción:

Estado: ☒ Activado

Clic en "Aceptar" y Listo.

Ya tenemos nuestra conexión a la base de datos.

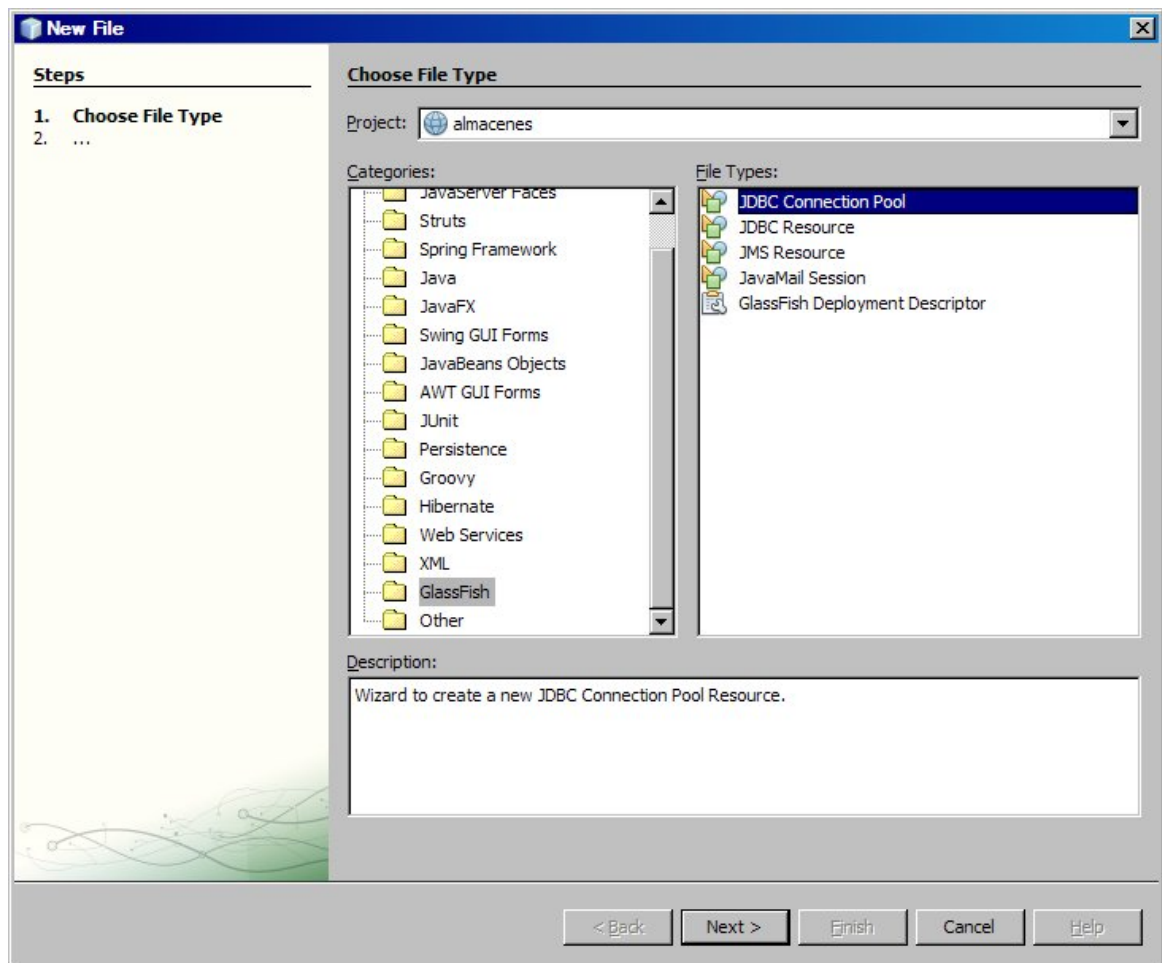
De ahora en adelante, cada vez que queramos usar la base de datos (en mi caso) almacenes, llamaré a jdbc/almacenes.

2. Desde NetBeans

Creando el Pool de Conexiones

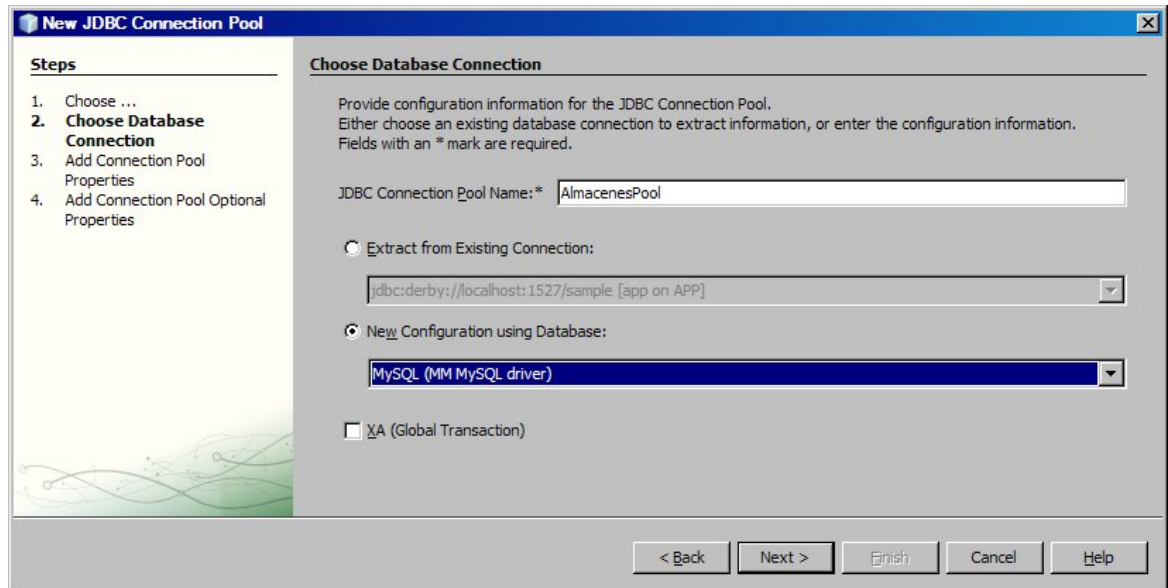
Para realizarlo desde NetBeans, necesitamos tener el proyecto que usará la base de datos abierto. Luego, presionamos Ctrl+N (File > New File)

1. Seleccionamos la categoría *Glassfish* y el tipo de archivo *JDBC Connection Pool*



Clic en *Next*

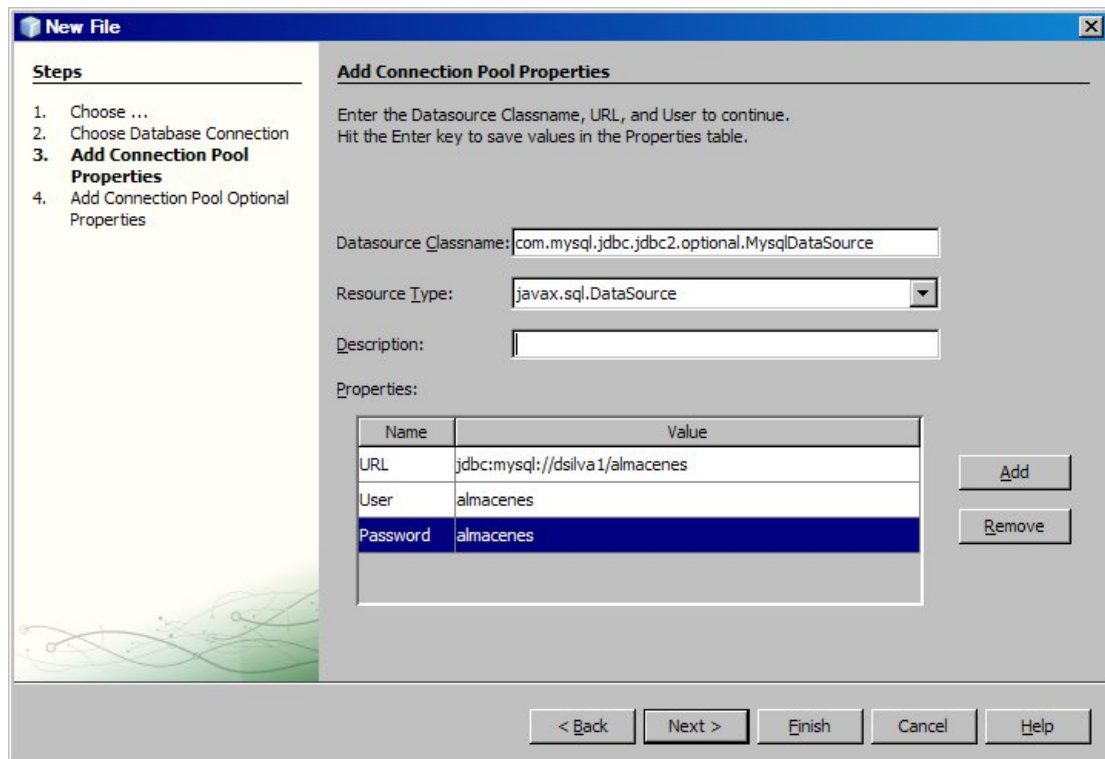
2. Escribimos el nombre de nuestro pool de conexiones. De igual manera, cualquier no es válido, pero se recomienda que sea en base a letras y números. Además seleccionamos cual es la base de datos que usaremos. Como es una conexión nueva, entonces seleccionamos "New configuration using database", y - en mi caso - selecciono MySQL:



The "New JDBC Connection Pool" dialog box is shown. It has a "Steps" pane on the left with four steps: 1. Choose ..., 2. Choose Database Connection (highlighted), 3. Add Connection Pool Properties, and 4. Add Connection Pool Optional Properties. The main area is titled "Choose Database Connection" and contains instructions: "Provide configuration information for the JDBC Connection Pool. Either choose an existing database connection to extract information, or enter the configuration information. Fields with an * mark are required." There are two radio buttons: "Extract from Existing Connection:" (unselected) and "New Configuration using Database:" (selected). The "JDBC Connection Pool Name:" field contains "AlmacenesPool". The "Extract from Existing Connection:" section has a text field with "jdbc:derby://localhost:1527/sample [app on APP]". The "New Configuration using Database:" section has a dropdown menu showing "MySQL (MM MySQL driver)". There is an unchecked checkbox for "XA (Global Transaction)". At the bottom are buttons: "< Back", "Next >", "Finish", "Cancel", and "Help".

Clic en *Next*

3. Seleccionamos el tipo de recurso (por omisión es `java.sql.DataSource`, escribimos el URL de nuestra conexión JDBC, el usuario y la contraseña.



The "Add Connection Pool Properties" dialog box is shown. It has a "Steps" pane on the left with four steps: 1. Choose ..., 2. Choose Database Connection, 3. Add Connection Pool Properties (highlighted), and 4. Add Connection Pool Optional Properties. The main area is titled "Add Connection Pool Properties" and contains instructions: "Enter the Datasource Classname, URL, and User to continue. Hit the Enter key to save values in the Properties table." There are three text fields: "Datasource Classname:" with "com.mysql.jdbc.jdbc2.optional.MysqlDataSource", "Resource Type:" with a dropdown showing "javax.sql.DataSource", and "Description:". Below these is a "Properties:" section with a table. The table has two columns: "Name" and "Value". It contains three rows: "URL" with "jdbc:mysql://dsilva1/almacenes", "User" with "almacenes", and "Password" with "almacenes". To the right of the table are "Add" and "Remove" buttons. At the bottom are buttons: "< Back", "Next >", "Finish", "Cancel", and "Help".

Name	Value
URL	jdbc:mysql://dsilva1/almacenes
User	almacenes
Password	almacenes

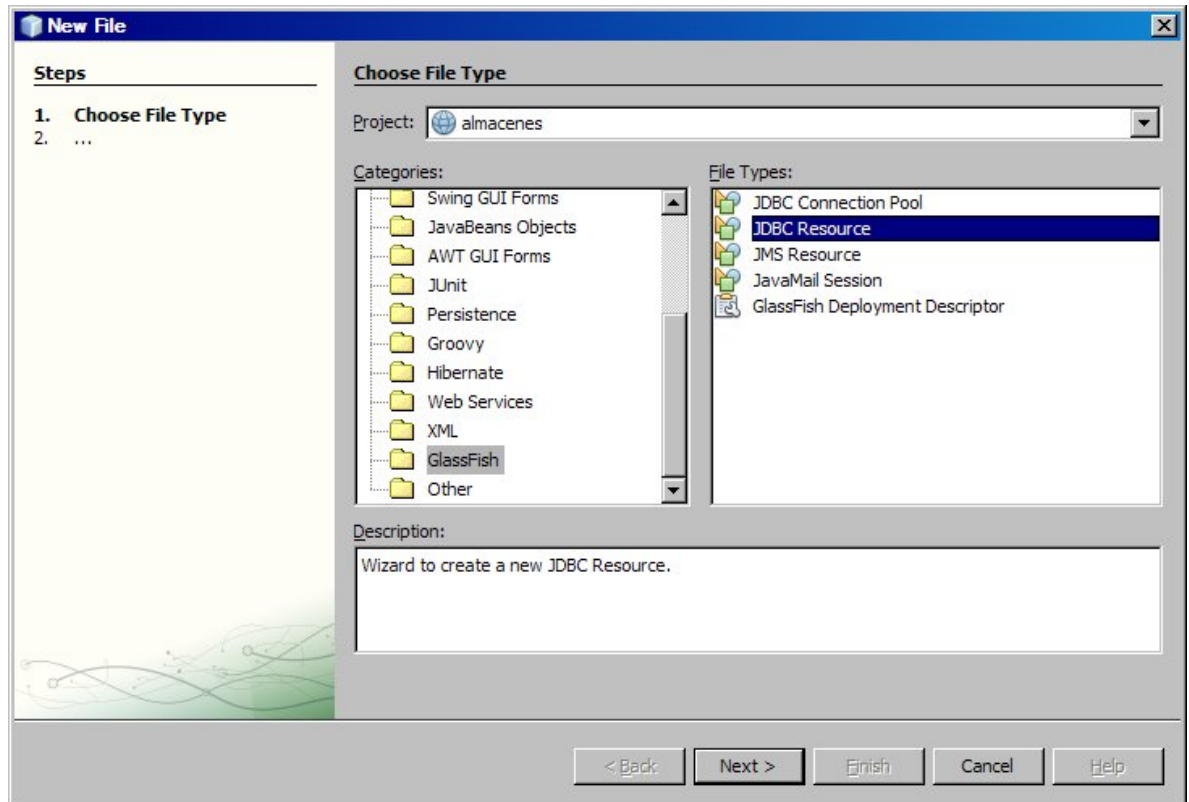
Clic en *Finish*

Listo, ya tenemos nuestro pool de conexiones creado para nuestro proyecto desde NetBeans

Creando el recurso JDBC

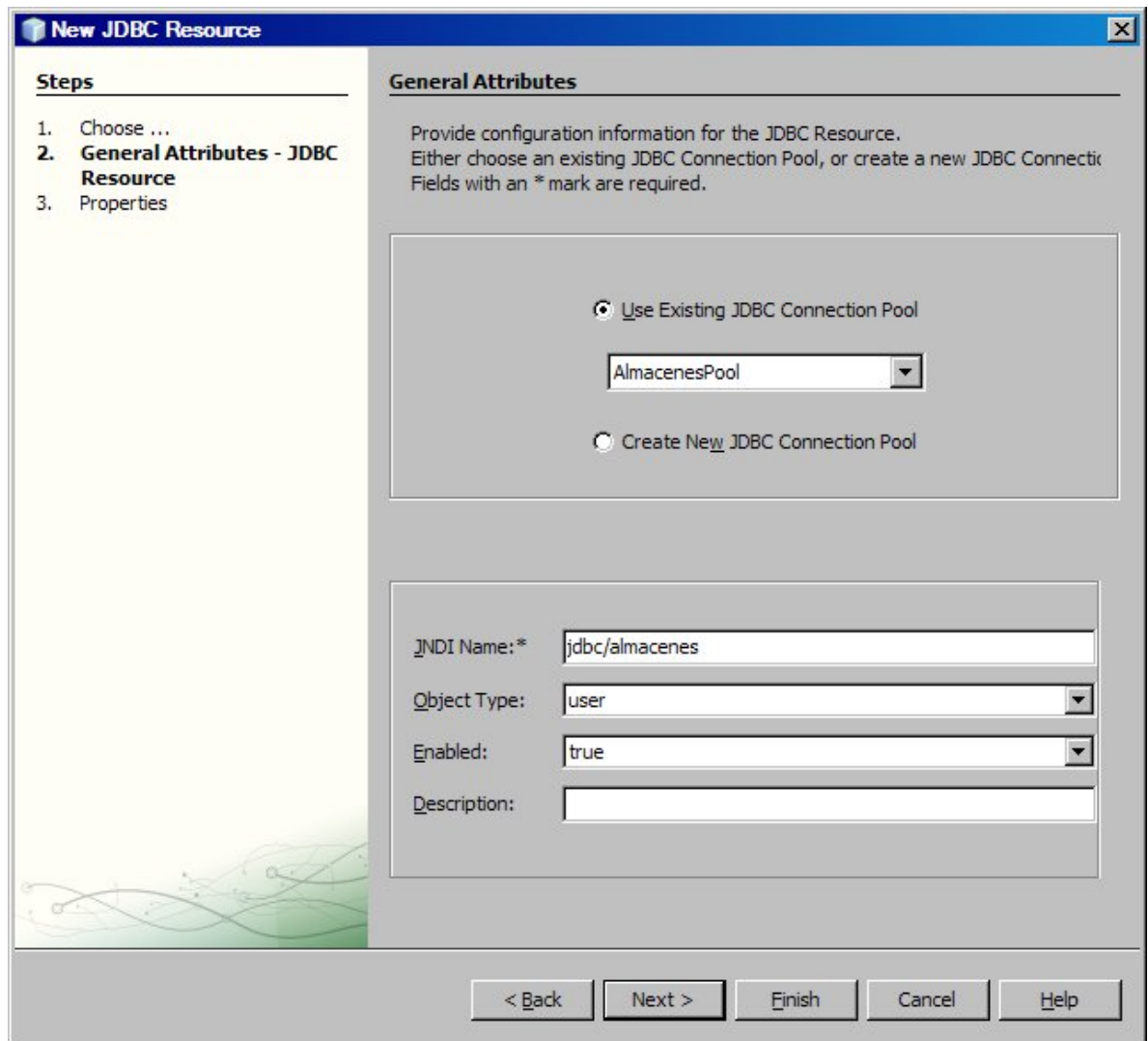
Esto también es tan fácil como cuando se hizo desde Glassfish.

1. Presionamos Ctrl+N (File > New File), seleccionamos la categoría *Glassfish* y el tipo de archivo *JDBC Resource*



Clic en *Next*

2. Seleccionamos el pool de conexiones que estará asociado a nuestro recurso JDBC. En mi caso es *AlmacenesPool*, y escribo el nombre en formato JNDI de nuestro recurso jdbc.



Clic en *Finish*

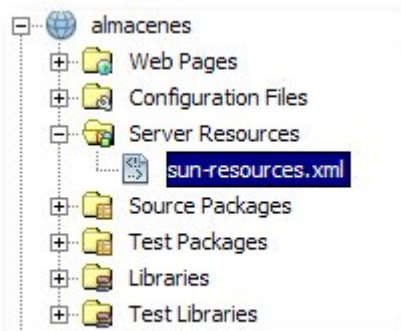
Listo, nada más. Ya tenemos nuestro recurso JDBC.

El recurso JDBC y el pool de conexiones creados desde NetBeans **se crearán en el glassfish unicamente cuando se despliegue el proyecto.**

¿Cuál es la diferencia en hacer desde Glassfish o desde NetBeans?

Cuando se hace desde NetBeans, **es solo para fines de desarrollo.**

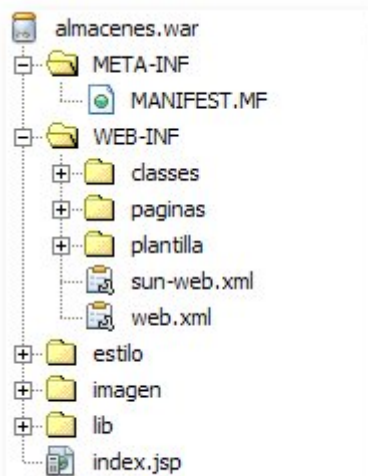
Notemos que se ha creado el archivo `sun-resources.xml` en nuestro proyecto. Dentro tiene toda la configuración que NetBeans usará para crear los recursos JDBC para que nuestra aplicación funcione en nuestro glassfish de desarrollo. Para nada más.



Cuando se crea el archivo .war para desplegar nuestra aplicación en un glassfish de producción, el archivo `sun-resources.xml` no existe dentro del paquete de instalación. Simplemente no hay.

Entonces ¿cómo nuestra aplicación puede conectarse a la base de datos estando en el glassfish de producción? Pues bien, en este glassfish de producción, creamos el JDBC resource directamente en el mismo glassfish.

Esto es bueno, porque cuando desarrollamos usamos una base de datos que es solo para desarrollo. Mientras que para poner en producción, usaremos otra conexión, otra base de datos que será para producción.



3. Creando Clase de Conexión

1. Crear proyecto web en java, y dentro de la carpeta Source Packages crear la siguientes clases en un paquete llamado `com.almacen.modelo`

Interface: `Conexion.java`

```
package com.almacen.modelo;
import java.sql.Connection;
public interface Conexion {
    public void conectar();
    public Connection getConexion();
    public void desconectar();
}
```

Clase: ConexiónPool.java

```
package com.almacen.modelo;

import java.sql.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;

public class ConexionPool implements Conexion{
    private Connection conn;

    public ConexionPool() {
        conn = null;
    }

    @Override
    public synchronized void conectar(){
        try {
            Context initCtx;
            initCtx = new InitialContext();
            DataSource ds = (DataSource)initCtx.lookup("jdbc/almacenes");
            conn=ds.getConnection();

        }catch(NamingException | SQLException e){
            System.out.println("db: " + e.getMessage());
        }
    }

    @Override
    public Connection getConexion(){
        return conn;
    }

    @Override
    public synchronized void desconectar(){
        try {
            conn.close();
        } catch (SQLException ex) {
            Logger.getLogger(ConexionPool.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
}
```

2. Elabore una pagina JSP para testear la conexión (seguir instrucciones del instructor)