MySQL y JDBC (I)

Bases de datos y Java

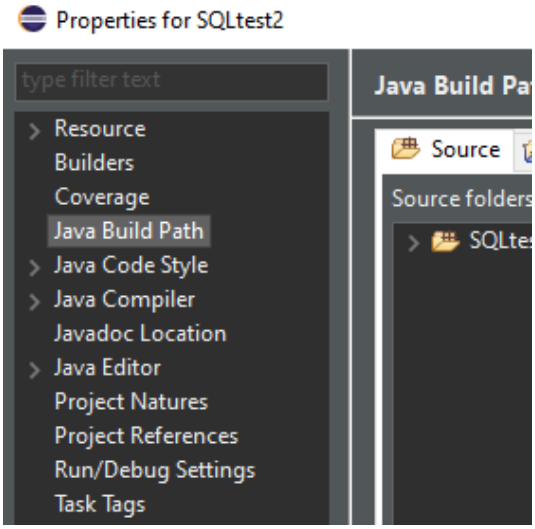
Para conectar a una base de datos en Java, necesitamos contar con JDBC (Java DataBase Connectivity). Esta librería se puede descargar desde dev.mysql.com (Elegir "plataforma independiente" y descargar el ZIP, descomprimirlo para obtener el JAR) o bien desde El repositoriocentral de Maven (gestor de dependencias).

La importación de la Biblioteca JAR en Eclipse puede hacerse de varias maneras:

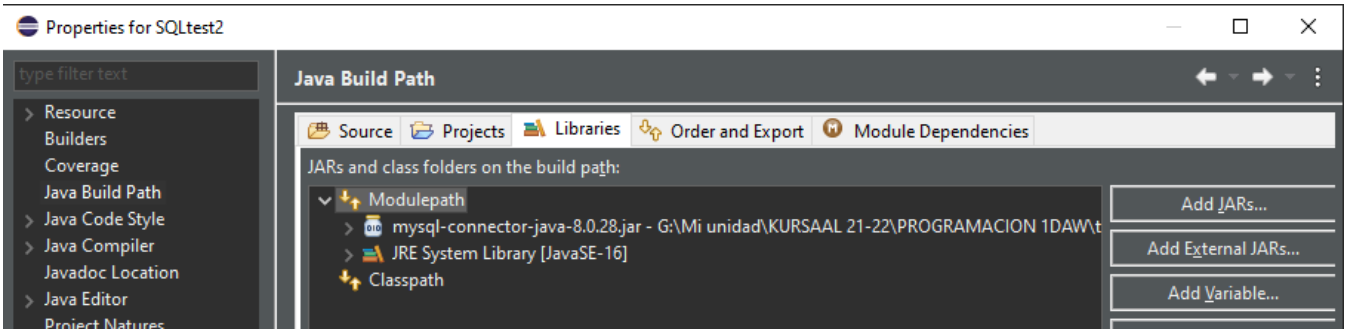
1. Utilizando un gestor de dependencias como Maven o Gradle..
2. Importando manualmente la dependencia.

Instalar manualmente JDBC

1. Descomprimir el archivo ZIP *descargado para que el JAR del mismo nombre que contiene sea visible (en abril de 2022 la versión es mysql-connector-java-8.0.28.jar)*
2. En la columna izquierda de Eclipse, botón derecho sobre el proyecto y última opción Propiedades
3. Elegir Java Build Path en la columna izquierda

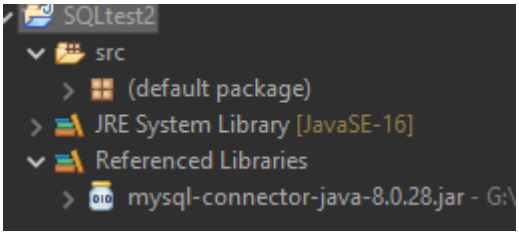


4. Elegir pestaña Libraries → Abajo seleccionar Modulepath → a la derecha botón Add External JARS…



5. Seleccionar el JAR y pulsar en Apply and Close en la ventana del punto 4

6. Si todo ha ido bien en la columna izquierda del proyecto de Eclipse aparecerá añadido el JAR



Dar permiso a un usuario

En nuestros primeros ejemplos usaremos el usuario root y contraseña vacía, que viene por defecto en MySQL.

Para dar permiso a un usuario concreto en una BBDD, dependiendo del SGBD que estemos usando, variará la sintaxis del DDL. Si estamos usando MySQL5.x:

```
> GRANT ALL PRIVILEGES ON NombreBD.* TO 'usuario'@'localhost' IDENTIFIED BY 'P9$$w0rd';
```

Con esto creamos usuario y damos privilegios.

Si estamos usando MySQL 8.x:

```
> CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'P9$$w0rd';

> GRANT ALL ON NombreBD.* TO 'usuario'@'localhost';
```

Servidor remoto

En el ejemplo anterior, la base de datos está en *localhost*. Pero no siempre será así. El servidor puede estar en un servidor remoto. Para poder conectarnos en remoto, el usuario creado para la base de datos debe utilizar como host la IP de origen, o bien un comodín (%):

```
> CREATE USER 'usuario'@'%' IDENTIFIED BY 'P9$$w0rd';

> GRANT ALL ON NombreBD.* TO 'usuario'@'%';
```

Por defecto *MySQL* está configurado para aceptar conexiones locales. Si queremos realizar conexiones remotas, tenemos que modificar el archivo de configuración de MySQL (*/etc/mysql/mysql.conf.d/mysqld.cnf*). Los parámetros a tener en cuenta son:

port: por defecto es 3306, pero se puede modificar.
bind-address: para permitir conexiones desde direcciones externas, podemos especificar como dirección *0.0.0.0*.

> Nota: esta configuración no es segura, y debe combinarse con medidas de seguridad adicionales en un servidor en producción.

Base de datos local/remota

A continuación se muestra un ejemplo de uso, para una base de datos llamada pizzas. El SGBD se encuentra en la máquina local (localhost) y las credenciales de usuario son pizzauser/Perro20.

```
import java.sql.Connection;
import  java.sql.DriverManager;
import java.sql.ResultSet;
import  java.sql.SQLException;

public class DBConnection {
    static String db = "pizzas";
    static String login = "pizzasuser";
    static String password = "Perro20";
    static String url = "jdbc:mysql://localhost:3306/" + db + "?
useSSL=false";
    Connection conn = null;
    public  DBConnection(){

        try{
            conn  =  DriverManager.getConnection(url,login,password);

            if (conn!=null){
                System.out.println("Conexión ok: " + conn);

                ResultSet rs = conn.prepareStatement("SELECT nombre FROM
pizzas").executeQuery();
                while (rs.next()){
                    System.out.println(rs.getString("nombre"));
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } catch  (ClassNotFoundException e){
            e.printStackTrace();
        }
    }
}
```

En caso de estar accediendo a una base de datos remota, el parámetro *url* debe ser modificado: *jdbc:mysql://ip_host:3306/nombreBD?useSSL=false*

Consultas

El siguiente ejemplo muestra cómo hacer una consulta:

```
    String query = "SELECT ...";
    ResultSet result = conn.prepareStatement(query).executeQuery();
    while (result.next()){
        result.getString("Campo1"); // devuelve una cadena de texto correspondiente al campo Campo1
        result.getInt("Campo2"); // devuelve una cadena de texto correspondiente al campo Campo2
    }
```

Borrados, inserciones y actualizaciones

Para insertar, utilizamos el método *executeUpdate* sobre un objeto de la clase *PreparedStatement*. En el siguiente ejemplo se hace directamente en una sola línea. El método *executeUpdate* devuelve el número de tuplas modificadas.

```
    int totalRows= conn.prepareStatement(updateQuery).executeUpdate(); // devuelve el número de tuplas modificadas
```