



Práctica Guiada: Conexión a Base de Datos con JDBC en un Proyecto Maven

Objetivos de la práctica

- Aprender a crear un proyecto Java con **Maven**.
- Incorporar dependencias necesarias (driver JDBC de MySQL).
- Programar una clase de conexión a la base de datos.
- Ejecutar operaciones básicas con JDBC.
- Reforzar la teoría de **JDBC** y su ciclo de uso.

1. Creación del Proyecto Maven

Teoría: Maven es una herramienta de gestión de proyectos en Java. Nos permite manejar dependencias externas (librerías) de forma sencilla a través del archivo `pom.xml`.

Pasos:

1. Abre tu IDE (IntelliJ, Eclipse o NetBeans).
2. Crea un **nuevo proyecto Maven** llamado `jdbc-demo`.
3. Define el paquete base: `com.ejemplo.jdbc`.

Tu estructura inicial debería ser algo así:

```
jdbc-demo/  
└─ src/  
    └─ main/  
        └─ java/  
            └─ com/ejemplo/jdbc/
```



2. Añadir Dependencia JDBC (MySQL)

Teoría: JDBC usa **drivers** específicos para cada gestor de bases de datos. El de MySQL se llama *MySQL Connector/J*.

En **pom.xml**, añade:

```
<dependencies>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.33</version>
  </dependency>
</dependencies>
```

Después de guardar, Maven descargará el driver automáticamente.

3. Crear Base de Datos de Prueba

Teoría: Vamos a trabajar con una base de datos muy sencilla para probar la conexión.

SQL:

```
CREATE DATABASE escuela;
USE escuela;

CREATE TABLE alumnos (
  id INT PRIMARY KEY AUTO_INCREMENT,
  nombre VARCHAR(50),
  edad INT
);
```



```
INSERT INTO alumnos (nombre, edad) VALUES  
( 'Ana', 20),  
( 'Luis', 22),  
( 'María', 19);
```

4. Clase de Conexión JDBC

Teoría: Para conectarnos con JDBC seguimos el flujo:

1. **DriverManager** obtiene la conexión.
2. Usamos un objeto **Connection**.
3. Creamos **Statement** o **PreparedStatement** para ejecutar SQL.
4. Procesamos resultados con **ResultSet**.
5. Cerramos los recursos.

Código – **DatabaseConnection.java**

```
package com.ejemplo.jdbc;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
public class DatabaseConnection {  
    private static final String URL = "jdbc:mysql://localhost:3306/escuela";  
    private static final String USER = "root"; // cambia según tu instalación  
    private static final String PASSWORD = "1234"; // cambia según tu instalación
```



```
public static Connection getConnection() {  
    Connection conn = null;  
    try {  
        conn = DriverManager.getConnection(URL, USER, PASSWORD);  
        System.out.println("✅ Conexión establecida correctamente");  
    } catch (SQLException e) {  
        System.out.println("❌ Error al conectar: " + e.getMessage());  
    }  
    return conn;  
}  
}
```

5. Clase para Probar la Conexión

Teoría: Una vez tenemos la conexión, podemos ejecutar consultas SQL.

Código – App.java

```
package com.ejemplo.jdbc;  
  
import java.sql.Connection;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
  
public class App {  
    public static void main(String[] args) {  
        try (Connection conn = DatabaseConnection.getConnection()) {  
            if (conn != null) {  
                // Consulta de todos los alumnos  
                String sql = "SELECT * FROM alumnos";  
                PreparedStatement stmt = conn.prepareStatement(sql);  
                ResultSet rs = stmt.executeQuery();  
            }  
        }  
    }  
}
```

```
while (rs.next()) {  
    int id = rs.getInt("id");  
    String nombre = rs.getString("nombre");  
    int edad = rs.getInt("edad");  
    System.out.println(id + " - " + nombre + " - " + edad);  
}  
}  
} catch (SQLException e) {  
    e.printStackTrace();  
}  
}  
}
```

6. Ejecución

1. Arranca el servidor MySQL.
2. Ejecuta la clase **App**.
3. Si todo funciona, deberías ver en consola los alumnos insertados en la base de datos.

Ejemplo de salida:

```
Conexión establecida correctamente  
1 - Ana - 20  
2 - Luis - 22  
3 - María - 19
```



7. Retos Finales

1. Modifica el programa para insertar un nuevo alumno usando **PreparedStatement**.
2. Implementa una consulta que muestre solo a los alumnos mayores de 20 años.
3. Añade un método **cerrarConexion()** en **DatabaseConnection** para gestionar el cierre de forma explícita.
4. Crea una nueva clase para conectarte a una base de datos con JDBC. Esta vez vas a realizar las operaciones sobre tu propia base de datos. Tus tareas son:
 - a. Método para consultar los elementos de una tabla y mostrarlos por consola.
 - b. Método para insertar elementos nuevos en dicha tabla.
 - c. Método para borrar un elemento de dicha tabla.
 - d. Método para borrar un elemento de dicha tabla.
 - e. ¿Qué pasaría si quieres introducir o hacer modificaciones sobre una tabla que tiene relaciones ?

8. Preguntas para Reflexionar

1. ¿Qué ventajas tiene usar **PreparedStatement** frente a **Statement**?
2. ¿Por qué es importante cerrar la conexión después de usarla?
3. ¿Qué pasaría si se deja el **autoCommit** desactivado y no se hace **commit()**?
4. ¿Cuáles son las diferencias principales entre **conectar con JDBC** y usar un **ORM** como Hibernate?
5. ¿Qué pasos extra añadirías si este programa tuviera que usarse en una aplicación con muchos usuarios simultáneos?