JDBC (Java Database Connectivity)

Comenzamos módulo nuevo, que es la conexión de nuestros programas a una base de datos, no importa la base de datos ni el servidor, siempre que tengamos algunas herramientas.

En el momento de comunicar Java con base de datos existen múltiples opciones:

1. **JDBC**: Es el acceso estándar de acceso a datos de Java. Digamos que todos los lenguajes tienen un jdbc parecido, es decir, una forma de conectar muy parecida a Java, C#, Python.
2. **Hibernate**: Hibernate es un Framework de acceso a datos de Java. Nos permite acceder a bases de datos y lectura de datos de una forma muy sencilla y escalar.

Para acceder a cualquier base de datos, lo primero que necesitamos son las librerías del fabricante.

Si vamos a consumir un Oracle, necesitamos las librerías JDBC de Oracle.

Si vamos a consumir un SQL Server, necesitamos las librerías JDBC de SQL Server.

Dichas librerías están en formato JAR.

Tenemos un modelo de objetos definidos muy claros.

* Registrar Driver. Para poder conectar, debemos indicar a Java que estamos utilizando un Driver JDBC.
* Connection: Necesitamos una conexión para acceder a los datos. Dicha conexión se abrirá cada vez que accedamos a los datos.
* Statement: Es un ejecutor de sentencias, se encarga de ejecutar la sentencia SQL que vaya en su interior.
* ResultSet: Es el objeto que nos permite almacenar los datos en su interior y poder recorrerlos, siempre que tengamos una consulta SELECT.

Creamos un nuevo proyecto **Java Project** llamado **proyectojdbc**

Copiamos el Driver de MySql dentro de nuestro proyecto para que, cuando lo subamos a GitHub siempre lo tengamos localizado en nuestra carpeta.

Vamos a crear una base de datos dentro de MySql y trabajar con esa base de datos.

Modelo de Objetos JDBC:

**DRIVERMANAGER**

Nos permite crear una conexión con la base de datos y recuperar dicha conexión para las consultas.

Dependiendo de la base de datos, necesitaremos una cadena de conexión u otra cadena de conexión.

**MYSQL**

jdbc:mysql://servidor:3306/bbdd

**ORACLE**

jdbc:oracle:thin:servidor:1521:bbdd

**CONNECTION**

Dicho objeto es generado a partir del DriverManager y es el encargado de poder ejecutar Sentencias/Statement.

**STATEMENT**

Es el objeto encargado de realizar las consultas sobre la base de datos.

Los Statement son creados a partir de un Connection.

Tenemos varios tipos de Statement:

* Statement: Consultas simples
* PreparedStatement: Se utiliza para las consultas parametrizadas
* CallableStatement: Se utiliza para llamar a procedimientos almacenados.

En el momento de ejecutar las consultas tenemos 3 opciones:

* executeQuery(): Se ejecuta con una consulta de selección (SELECT) y devuelve un ResultSet con los registros
* executeUpdate(): Ejecuta una consulta de acción (INSERT, UPDATE, DELETE) y nos devuelve un INT con el número de registros afectados por la consulta.
* execute(): Ejecuta cualquier consulta, no devuelve resultados.

**RESULTSET**

Es el objeto que contiene los datos de una consulta SELECT.

El objeto ResultSet se crea a partir de un Statement y el método executeQuery()

Es un cursor de datos.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Por defecto, es **forward-only**, es decir, solamente podemos recorrerlo hacia delante.

Tiene un método que es el método **next()** que nos permite ir de fila en fila y devuelve un Boolean indicando si quedan registros (true) o no (false)

Para acceder a los datos, tiene una serie de métodos que pueden recuperar directamente el tipado de una columna de base de datos.

Podemos indicar tanto índice (empieza en CERO) como nombre de Columna

getInt(0)

getString(0)

getInt(“DEPT\_NO”)

getString(“DEPT\_NO”)

Como norma, siempre ENTRAR/SALIR

Siempre que entremos (utilicemos datos) debemos liberar dichos datos (SALIR).

Los objetos tienen un método llamado **close()** que debemos utilizar siempre que nuestras instrucciones han acabado.

Configurar el proyecto de Java para utilizar nuestro Driver de MySql.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Ya tenemos configurado el proyecto, es el momento de mostrar datos.

Comenzamos creando una clase llamada **Class01ReadEmpleados.java**

Texto

El contenido generado por IA puede ser incorrecto.

En esta librería está todo el modelo de acceso a datos JDBC

Imagen que contiene Icono

El contenido generado por IA puede ser incorrecto.

**CLASS01READEMPLEADOS.JAVA**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import java.sql.Statement;**

**public class Class01ReadEmpleados {**

**public static *void* main(*String*[] *args*) {**

**//1) REGISTRAR LA CLASE DE NUESTRO DRIVER**

**//JDBC DE MYSQL**

**try {**

**Class.forName("com.mysql.cj.jdbc.Driver");**

**//2) NECESITAMOS UNA CADENA DE CONEXION**

***String* connectionString =**

**"jdbc:mysql://localhost:3306/hospital";**

**//3) CREAR UNA CONEXION MEDIANTE DRIVERMANAGER**

***Connection* cn =**

**DriverManager.getConnection(connectionString**

**,"root", "mysql2025");**

**//4) CONSULTA SOBRE LA BBDD**

***String* sql = "select \* from EMP";**

**// 5) CREAMOS EL TIPO DE STATEMENT DEPENDIENDO**

**//DE LA CONSULTA. CONSULTA SIMPLE**

***Statement* st = cn.createStatement();**

**// 6) COMO ES CONSULTA SELECT, NECESITAMOS**

**//UN ResultSet Y EL METODO executeQuery()**

***ResultSet* rs = st.executeQuery(sql);**

**//7) RECORREMOS LOS REGISTROS MEDIANTE UN**

**//BUCLE WHILE**

**while (rs.next()){**

***String* apellido =**

**rs.getString("APELLIDO");**

**System.out.println("Apellido: " + apellido);**

**}**

**//8) LIBERAMOS LOS RECURSOS**

**rs.close();**

**cn.close();**

**} catch (*ClassNotFoundException* *e*) {**

**System.out.println("Class" + e);**

**} catch (*SQLException* *ex*){**

**System.out.println("Sql: " + ex);**

**}**

**}**

**}**