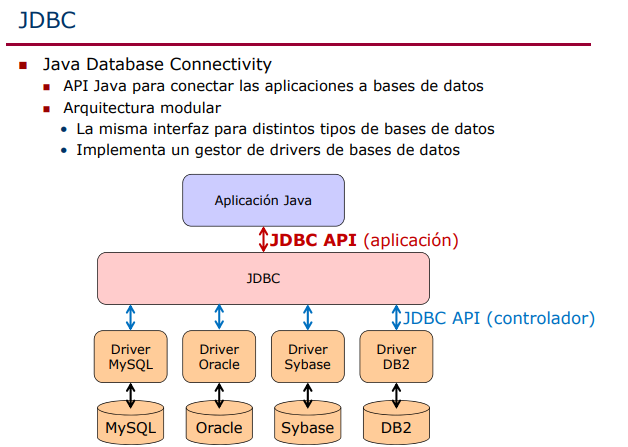
JDBC (Java Data Base Conectivity)

JDBC nos sirve para conectar nuestro programa Java con una base de datos, y necesitaremos un driver diferente para cada una de las bases de datos. Cada driver podremos descargárnoslo desde la página web de la base de datos correspondiente.



## Descarga e instalación el driver de MySQL

Para descargar el **driver** y añadirlo al proyecto, es necesario ir a la página de la BBDD con la que vamos a trabajar (en este caso MySQL), y descargar el JDBC de MySQL. Simplemente poniendo en un buscador “download jdbc mysql”, el primer enlace que nos aparecerá será la página para descargarlo. Puede descargarse un .zip o un .tar, lo que prefiramos (al descargar nos aparecerá una página para darnos de alta, pero tiene un enlace en la parte inferior para poder pasar la pantalla sin logarnos).

Una vez lo tengamos descargado, lo descomprimimos en cualquier carpeta de nuestro ordenador, y nos quedaremos únicamente con el fichero **-bin.jar** (ejemplo: mysql-connector-java-5.1.47-bin.jar).

Ese fichero será el que tengamos que incluir en cada proyecto que usemos en eclipse donde tengamos que acceder a la BBDD.

La manera de incluir ese jar en nuestro proyecto es la misma que con cualquier jar en eclipse, a través de las propiedades del proyecto, Java Build Path, librerías, añadir un jar externo.

Una vez añadido, veremos que nos aparece como el resto de librerías dentro de nuestro proyecto, y ya podremos usar la conexión a MySQL.

## Conexión con BBDD a través de JDBC

Para poder utilizar los objetos necesarios para nuestra conexión a BBDD, tendremos que importar las librerías donde se encuentran todos estos objetos, que son **java.sql**, y **javax.sql**.

**import** java.sql.\*;

**import** javax.sql.\*;

Todo tiene que estar registrado dentro de un Try-Catch para registrar cualquier Lo **primero** es registrar el driver que vamos a utilizar para ello usamos el código:

// Asociamos el driver de MySQL

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

Hecho esto el siguiente paso es establecer la conexión utilizando la clase **java.sql.DriverManager**, donde rellenamos los datos de url : “jdbc:mysql://localhost:3306/nombreBD” , usuario : “root” y password : “” (son el usuario y password por defecto para MySQL), quedando así:

//Creamos un objeto de tipo Connection para poder crear la conexión con el servidor

Connection conexion = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/BDEMPLEADOS","root","");

//Creamos un objeto Statement que recogerá esa conexión y nos permitirá ejecutar las sentencias SQL

Statement objetoStatement = conexion.createStatement();

## Consultas con JDBC

Para poder hacer una consulta, necesitaremos crear un objeto ResultSet para recoger el resultado de la consulta, y luego recorrer el mismo para obtener todo lo devuelto:

String sentencia = "SELECT \* FROM EMPLEADOS";

ResultSet resultado = objetoStatement.executeQuery(sentencia);

Nota: En BBDD pasa lo mismo que con cualquier gestión de recursos y memoria. Hay que tratar de hacer consultas que devuelvan sólo los datos que vamos a necesitar y no más, porque es un consumo de almacenamiento y recursos completamente innecesario (igual que pasa con los import de librerías).

Para recorrer el ResultSet, utilizaremos los métodos get… correspondientes a cada tipo de dato, y pasaremos como parámetro del método el nombre del campo (literal a como está en la tabla) o bien el número de columna del campo en la tabla:

//Recorremos nuestro ResultSet para recuperar todo lo devuelto por nuestra sentencia

**while** (resultado.next()) {

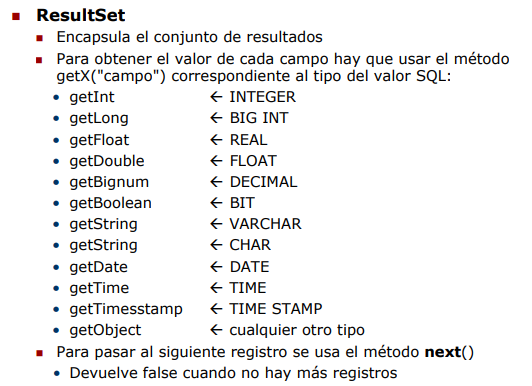
System.***out***.println(resultado.getInt("CODIGO") + " "

+ resultado.getString("NOMBRE") + " "

+ resultado.getDate("FECALTA") + " "

+ resultado.getDouble("SUELDO"));

}



## Modificaciones, Inserciones, y Borrado con JDBC

En este caso no necesitaremos recoger ningún dato de vuelta, porque no devuelven nada en ninguno de los tres casos, por lo que sólo tendremos que ejecutar la sentencia, y las tres pueden ejecutarse con el mismo método:

String sentenciaInsert = "INSERT INTO EMPLEADOS (CODIGO, NOMBRE, SUELDO) VALUES (1, 'MANOLO', 1500) ";

String sentenciaUpdate = "UPDATE EMPLEADOS SET NOMBRE='PEPE' WHERE CODIGO=3";

String sentenciaDelete = "DELETE FROM EMPLEADOS WHERE CODIGO = 6";

//Para ejecutar el Insert

objetoStatement.executeUpdate(sentenciaInsert);

System.***out***.println("Registro insertado");

//Para ejecutar el Update

objetoStatement.executeUpdate(sentenciaUpdate);

System.***out***.println("Registro modificado");

//Para ejecutar el Delete

objetoStatement.executeUpdate(sentenciaDelete);

System.***out***.println("Registro borrado");

## Cerrar la conexión

Para finalizar, es recomendable cerrar la conexión a la BBDD:

//Cerramos las conexiones de la BBDD

**if** (resultado != **null**) { // liberar los ResultSet

resultado.close();

}

**if** (objetoStatement != **null**) { // liberar los Statement

objetoStatement.close();

}

**if** (conexion != **null**) { // liberar la conexión a la BD

conexion.close();

}

## Ejemplo de código completo

**import** java.sql.\*;

**import** javax.sql.\*;

**public** **class** PruebasBBDD {

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

**try**

{

// Asociamos el driver de MySQL

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

//Creamos un objeto de tipo Connection para poder crear la conexión con el servidor

Connection conexion = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/BDEMPLEADOS","root","");

//Creamos un objeto Statement que recogerá esa conexión y nos permitirá ejecutar las sentencias SQL

Statement objetoStatement = conexion.createStatement();

//Ejecutamos nuestra sentencia SQL

//Opción 1: Si es una consulta, necesitaremos un objeto de tipo ResultSet donde se almacenará el resultado de la consulta

String sentencia = "SELECT \* FROM EMPLEADOS";

ResultSet resultado = objetoStatement.executeQuery(sentencia);

//Recorremos nuestro ResultSet para recuperar todo lo devuelto por nuestra sentencia

**while** (resultado.next()) {

System.***out***.println(resultado.getInt("CODIGO") + " "

+ resultado.getString("NOMBRE") + " "

+ resultado.getDate("FECALTA") + " "

+ resultado.getDouble("SUELDO"));

}

//Opción 2: Si es una modificación, borrado o insertado, utilizamos otra manera de ejecutar la sentencia, porque no devuelve nada

// y por tanto no necesitaríamos un ResultSet

String sentenciaInsert = "INSERT INTO EMPLEADOS (CODIGO, NOMBRE, SUELDO) VALUES (1, 'MANOLO', 1500) ";

String sentenciaUpdate = "UPDATE EMPLEADOS SET NOMBRE='PEPE' WHERE CODIGO=3";

String sentenciaDelete = "DELETE FROM EMPLEADOS WHERE CODIGO = 6";

//Para ejecutar el Insert

objetoStatement.executeUpdate(sentenciaInsert);

System.***out***.println("Registro insertado");

//Para ejecutar el Update

objetoStatement.executeUpdate(sentenciaUpdate);

System.***out***.println("Registro modificado");

//Para ejecutar el Delete

objetoStatement.executeUpdate(sentenciaDelete);

System.***out***.println("Registro borrado");

//Cerramos las conexiones de la BBDD

**if** (resultado != **null**) { // liberar los ResultSet

resultado.close();

}

**if** (objetoStatement != **null**) { // liberar los Statement

objetoStatement.close();

}

**if** (conexion != **null**) { // liberar la conexión a la BD

conexion.close();

}

} **catch** (ClassNotFoundException ex) {

//Si ha fallado la asociación del driver

System.***out***.println("Error al registrar el driver de MySQL: " + ex);

} **catch** (SQLException sqle) {

//Si ha habido algún problema de SQL

System.***out***.println("Error SQL: " + sqle.getMessage() + sqle.getSQLState());

} **catch** (Exception e) {

//Si ha fallado cualquier otra cosa dentro de nuestro Try

System.***out***.println("Error de conexión: " + e);

}

}

}