Conexión desde Java a BD (MySql).

Lo primero que se necesita para conectarnos con una base de datos es un **Driver** o **Connector**. Ese Driver es la clase que, de alguna forma, sabe cómo hablar con la base de datos.

Según el gestor de base de datos que se utilice, se usara un driver u otro. Para MySql se usa **mysql-connector-java-5.0.5-bin.jar** que se guarda en el directorio \eclipse\jre\lib\ext.

Los pasos a seguir para trabajar con la base de datos desde Java son:

1º.- Importar el paquete java.sql.*

2º.- Se carga el driver para lo cual se pone Class.forName("Nombre del driver").
Para el caso del MySql, será:

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

3º.- Ahora ya se puede conectar a la base de datos invocando el método getConnection.
La sintaxis es:

```
Connection conn = DriverManager.getConnection(url,usr,pwd);
```

Donde **url** es una cadena compuesta por el protocolo + driver de la bbdd+ nombre y lugar donde se encuentra la base de datos, **usr** es el usuario para acceder a la bbdd y **pwd** es la clave de acceso.

Para el caso de MySql seria:

url → "jdbc:mysql://localhost/Nombre_de_la_Base_de _Datos"
usr →"root" y pwd →""

Todas estas instrucciones estarán recogidas dentro de un try-catch para que recoja la excepción en caso de no poder realizarse la conexión.

Ej;

```
try{
   Class.forName("com.mysql.jdbc.Driver");
   Connection conn;
   conn=DriverManager.getConnection("jdbc:mysql://localhost/bbdd","root","");
    ……..
}catch(ClassNotFoundException cnfe){
      System.out.println("Driver JBDC no encontrado");
      cnfe.printStackTrace();
}catch(SQLException sqle){
      System.out.println("Error al conectarse a la BD");
      sqle.printStackTrace();
}catch(Exception e){
       System.out.println("Error general");
      e.printStackTrace();
}
```

4º.- Una vez creada la conexión se crean las sentencias SQL, a través de los métodos execute(), executeUpdate() y executeQuery() de la interface Statement.

Statement st=conn.createStatement():

El método execute() se utiliza para la creación de tablas (Create table).

El método executeUpdate() se utiliza para las sentencias UPDATE, INSERT o DELETE, devuelve un número entero que indica la cantidad de registros afectados.

El método executeQuery() se utiliza para la SELECT, devuelve un conjunto de registros que se almacenan en un objeto ResultSet.

Ej:

```
st.execute("CREATE TABLE Libros (id  NUMBER(11), title VARCHAR2(64))”);
```

Ej:

```
int nr;
String cnsSQL;
cnsSQL=”INSERT TO autor VALUES(127,’Peter’,’Norton’)”;
nr=st.executeUpdate(cnsSQL);
cnsSQL=”UPDATE autor SET nombre=’Pedro’ WHERE nombre=’Peter’”;
nr=st. executeUpdate(cnsSQL);
```

Ej:

```
ResultSet rs=st.executeQuery(“SELECT * FROM autor);
```

5º.- ResultSet también tiene los métodos next() y getXXX().

next() recorre los registros devueltos.
getXXX () recupera los valores de las columnas por su nombre o posición.

Ej:

```
while(rs.next()){
     System.out.println(“Autor ”+rs.getString(1));
}
```

Cuando se lanza un método getXXX sobre un objeto ResultSet, el driver JDBC convierte el dato que se quiere recuperar a el tipo Java especificado y entonces devuelve un valor Java adecuado. La conversión de tipos se puede realizar gracias a la clase java.sql.Types. En esta clase se definen lo que se denominan tipos de datos JDBC, que se corresponde con los tipos de datos SQL.

SQL	Java
BIGINT	getLong()
BINARY	getBytes()
BIT	getBoolean()
CHAR	getString()
DATE	getDate()
DECIMAL	getBigDecimal()
DOUBLE	getDouble()
FLOAT	getDouble()
INTEGER	getInt()
LONGVARBINARY	getBytes()

SQL	Java
LONGVARCHAR	getString()
NUMERIC	getBigDecimal()
OTHER	getObject()
REAL	getFloat()
SMALLINT	getShort()
TIME	getTime()
TIMESTAMP	getTimestamp()
TINYINT	getByte()
VARBINARY	getBytes()
VARCHAR	getString()

6º.- Por último habrá que cerrar todas las conexiones con el método close()

Ej: st.close();
conn.close();

EJEMPLO

```
import java.sql.*;

public class Programa {
    public static void main(String args[]){
      try {
         Class.forName("com.mysql.jdbc.Driver");
         Connection conexion;
         conexion=DriverManager.getConnection ("jdbc:mysql://localhost/ejemplo”,”root”,””);
         Statement instruccion = conexion.createStatement();
         ResultSet  tabla;
         tabla= instruccion.executeQuery("SELECT cod , nombre FROM datos");
         System.out.println("Codigo\tNombre");
         while(tabla.next())
               System.out.println(tabla.getInt(1)+"\t"+tabla.getString(2));
         tabla.close();
         instruccion.close();
         conexion.close();
       }
      catch(ClassNotFoundException e) {
           System.out.println(e);
       }
      catch(SQLException e) {
           System.out.println(e);
       }
      catch(Exception e) {
           System.out.println(e);
       }
   }
}
```