JDBC

ESTABLECER CONEXIÓN

1 - Importamos la libreria ***'mysql-connector'*** en ***'modules-dependencies'.***

2 - Cargamos el driver con el método ***'forName()'*** pasandole por parámetro ***'com.mysql.cj.jdbc.Driver'***.

3 - Creamos una nueva ***Connection*** con el método ***'getConnection()'*** y le pasámos por parámetro la ruta local ***'jdbc:mysql://localhost/nombreBD'*** , el user y la psw.

4 - ***IMPORTANTE:*** Siempre debemos ***liberar los recursos*** de la conexión.

SENTENCIAS SIMPLES

1 - Creamos un ***'Statement'*** a partir de la conexión con el método ***'createStatement()'.***

2 - Declaramos un ***'String'*** con la sentencia *SQL*.

3 - Almacenamos el resultado en un ***'ResultSet'*** (iterador), llamando al método ***'executeQuery(sql)'*** o ***'executeUpdate(sql)'*** del ***'Statement'*** creado anteriormente.

4 - Cada iteracción del resultado es un registro, para acceder a sus columnas usamos los métodos ***'.getInt()...'***, pasando el ***numColumna*** o el ***nomColumna*** de la tabla.

5 - Cuando hacemos uso de ***'executeUpdate(sql)'*** no nos devuelve un ***ResultSet*** sino un ***int*** con el numero de filas afectadas.

6 - ***IMPORTANTE:*** Siempre debemos ***liberar los recursos*** de la sentencia y resultado.

SENTENCIAS PREPARADAS

1 - Declaramos un ***'String'*** con la sentencia *SQL*, añadiendo ***'?'*** en los parámetros que introduciremos nosotros.

2 - Creamos una ***'PreparedStatement'*** a partir de la conexión con el método ***'prepareStatement(sql)'.***

3 - Introducimos los parámetros con los métodos ***'.setInt()...'***, pasando el ***numParametro*** y el ***valor***.

4 - Almacenamos los resultados de igual manera que en las sentencias simples.

5 - ***IMPORTANTE:*** Siempre debemos ***liberar los recursos*** de la sentencia y resultado.

CLASE RESULTSET

1 - Si no sabemos la estructura de la BD podemos obtener datos de ella con la clase ***ResultSet*** con los métodos ***'getTables()','getColumns()','getPrimaryKeys()'***

2 - ***EJEMPLO:*** para extraer datos de las tablas, creamos un '***DataBaseMetaData***' a partir de la conexión con el método ***'.getMetaData()'*** ,después creamos un String[] de los tipos que queremos obtener ***'TABLE', 'VIEW' ...***

3 - Después almacenamos el resultado en un ***ResultSet*** usando el método ***'.getTables(null,null,null, tipos)'*** para la BD.

4 - De cada registro, podremos obtener con el ***'.getString()'*** el ***'TABLE_CAT', 'TABLE_SCHEM', 'TABLE_NAME', 'TABLE_TYPE' ...***

5 - ***IMPORTANTE:*** Siempre debemos ***liberar los recursos*** del resultado.

EJECUCIÓN DE PROCEDIMIENTOS

1 - En primer lugar creamos una ***'Statement'*** como hemos visto anteriormente y ejecutamos la creación del procedimiento con el método ***'.execute(sql)'.***

2 - Creamos un ***String*** con el ***SQL*** introduciendo la sentencia entre corchetes ***'{call...}'.***

3 - Creamos un ***'CallableStatement'*** a la conexión usando el método ***'.prepareCall(sql)'.***

4 - Introducimos los parámetros de igual manera que con las ***PreparedStatement.***

5 - Usamos los métodos ***'.executeUpdate()'*** o ***'.executeQuery()'*** respectivamente de la ***'CallableStatement'***, sabemos que este último necesita un ***ResultSet.***

PARÁMETROS DE SALIDA EN PROCEDIMIENTOS

1 - En primer lugar debemos especificar el parametro ***OUT*** en el ***SQL*** de creación del procedimiento.

2 - Cuando registramos los argumentos del procedimiento, en el parámetro ***OUT*** debemos usar el método ***'.registerOutParameter(numP, Types.VARCHAR)'.***

3 - Una vez ejecutado el procedimiento de igual manera que antes, simplemente debemos hacer un sout ***'.getString(numPout)'*** para mostrar el resultado.

RESÚMEN SQL

DDL

```
AUTO_INCREMENT
ALTER tabla MODIFY columna
CHECK (campo >= ...)
PRIMARY KEY (...,...)
DEFAULT valor
FOREIGN KEY nom_fk (campo) REFERENCES tabla (campo) ON UPDATE CASCADE ON DELETE CASCADE
```

CONSULTAS

```
SELECT DISTINCT
BETWEEN v AND v
LIKE 'cadena%'
IN ('valor',...)
WHERE NOT
WHERE EXISTS ()
```

PROCEDIMIENTOS

```
DECLARE var INT;
SELECT INTO var

CASE
    WHEN condition1 THEN result1
    ELSE result
END;

SET var = valor;
CREATE OR REPLACE

IF condition THEN
        SELECT result;
    ELSE
        SELECT resultElse;
    END IF;
```