RESUMEN: CONECTAR JAVA CON MYSQL

Añadir el Driver MySQL

Cada vez que iniciemos un nuevo proyecto:

1. Propiedades->Biblioteca (Libreries).
2. Seleccionar la opción: Añadir biblioteca (Add Library).
3. De la lista elegir: MySQL JDBC Driver.
4. Aparecerá dentro de bibliotecas como una nueva entrada.

Establecer una conexión:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;


public class Conexion {

   String BD ="viajes"; //nombre de la base de datos
   String URL_BD = "jdbc:mysql://localhost:3306/" + BD; //URL conexion BD
   String LOGIN = "root"; //usuario de la BD
   String PASS= ""; //Contraseña
   public Connection conexion;

   public Conexion(){}

   public Connection conexionMySQL(){
      conexion = null;
      try {
         Class.forName("con.mysql.jdbc.Driver").newInstance();
         conexion = DriverManager.getConnection(URL_BD, LOGIN, PASS);
      } catch (ClassNotFoundException | InstantiationException | IllegalAccessException ex) {
         System.err.println("Error driver JDBC. " + ex.toString());
         Logger.getLogger(Conexion.class.getName()).log(Level.SEVERE, null, ex);
      } catch (SQLException ex) {
         System.err.println("Error de conexión a la BD");
         Logger.getLogger(Conexion.class.getName()).log(Level.SEVERE, null, ex);
      }
      return conexion;
   }
}
```

Ya tenemos una clase llamada conexión con un método llamado: conexionMySQL (), que nos devuelve el identificativo de la conexión.

Desde cualquier otra clase podemos usarla así:

```
Conexion conexion = new Conexion();

Connection con = conexion.conexionMySQL();
```

Sentencias sql:

Permiten interactuar con bases de datos relacionales y es soportado por casi todos los administradores de bases de datos actuales.

Para realizar cualquier acción sobre la base de datos necesitamos un objeto Statement sobre la conexión.

Ejecutar una consulta – executeQuery():

```
// Preparamos la consulta sobre la base de datos

Statement statement = conexion.createStatement();

String select = "select * from categorias";

ResultSet rs = statement.executeQuery (select);
```

Método executeQuery() → sirve para realizar una consulta a base de datos.

- El parámetro que se pasa en un String en el que está la consulta en lenguaje SQL.
- No hace falta terminarlo con punto y coma.
- El resultado lo devuelve en el método ResultSet, que es una clase java parecida a una lista que contiene el resultado de la consulta. En lugar de contener todos los datos, los va consiguiendo de la BD según se van pidiendo.

  Para tener en cuenta: ResultSet puede resultar lento, pero evita que una consulta pesada dé muchos resultados tarde mucho porque llena la memoria del programa java.

Ver los resultados de la consulta - ResultSet:

El ResultSet contiene los registros leidos de la base de datos.

- El Statement.executeQuery(), tiene internamente un "puntero" apuntando justo delante (antes) del primer registro.
- El método *next()* del ResultSet hace que dicho puntero avance al siguiente registro, en este caso, al primero.
  - Si lo consigue, el método next() devuelve true.
  - Si no lo consigue (no hay siguiente registro que leer), devuelve false.

Métodos de ResultSet:

- next( );// accede al siguiente registro y devuelve false cuando no hay más registros
- first( ); // accede al primero
- previous( );// al anterior
- last( );// al último
- getInt("campo") ó getInt(indice columna); devuelve el contenido numérico entero de la columna
- getDouble("campo") ó getDouble(indice columna);// devuelve el contenido numérico double de la columna
- getString(("campo") ó getString (indice columna);// devuelve el contenido de la cadena de la columna
- getObject (índice columna);//devuelve un objeto en general de cualquier tipo.

Curiosidades para tener en cuenta:

Los índices de columna siempre comienzan por 1.

Se suelen visualizar los resultados de una consulta en un componente JTable

Es responsabilidad nuestra saber qué tipo de dato hay en cada columna. Si nos equivocamos y RecordSet es capaz de hacer la conversión, la hará por nosotros. Por ejemplo, en cualquiera de los campos anteriores podemos pedir un getString() y nos devolveran los números como String y la fecha como String.

También podemos usar getObject(), y el RecordSet nos devolverá el Object más adecuado para el tipo de campo que pedimos.

Cerrar la conexión con la BD:

```
// Cerramos la conexion a la base de datos.

conexion.close();
```

Las sentencias SQL más comunes:

Seleccionar datos:

```
select  (lista de campos a seleccionar) * (si son todos)
        from tablas
                where (condiciones de selección)
                        order by columna1, [ASC|DES], columna2, [ASC|DES],….
```

Modificar datos:

```
update tabla
        set columna1=expression, culomna2=expression,…..
                where (condición de búsqueda)
```

Insertar registros en una tabla:

```
insert into tabla [columna1, columna2,…] //si se omiten son todas
        values(expresión1, expresión2…)
```

Borrar registros de una tabla:

```
delete from tabla
        where (condición de búsqueda)
```