**Usar PreparedStatement**

*PreparedStatement* nos brinda un sistema para precompilar la sentencia SQL y guardarla para ser ejecutada inmediatamente, sin necesidad de analizarla en cada caso.

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**import** java.sql.SQLException;

**public** **class** UsoPreparedStatement {

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

// Paso 1: Cargar el driver

**try** {

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

} **catch** (ClassNotFoundException e) {

System.***out***.println("No se ha encontrado el driver para MySQL");

**return**;

}

System.***out***.println("Se ha cargado el Driver de MySQL");

// Paso 2: Establecer conexión con la base de datos

String cadenaConexion = "jdbc:mysql://localhost:3306/TIENDA";

String user = "root";

String pass = "curso";

Connection con;

**try** {

con = DriverManager.*getConnection*(cadenaConexion, user, pass);

} **catch** (SQLException e) {

System.***out***.println("No se ha podido establecer la conexión");

System.***out***.println(e.getMessage());

**return**;

}

System.***out***.println("Se ha establecido la conexión con la BD");

// Paso 3: Interactuar con la BD

**try** {

String sql = "INSERT INTO CLIENTE (NIF, NOMBRE, DOMICILIO, TLF, CIUDAD) " + "VALUES (?, ?, ?, ?, ?)";

String nif = "55667788A";

String nombre = "DELGADO PEREZ CARLOS";

String domicilio = "C/ ALENZA, 7";

String tlf = "616667766";

String ciudad = "MADRID";

System.***out***.println("Se va a ejecutar la sentencia SQL:");

System.***out***.println(sql);

PreparedStatement sentencia;

sentencia = con.prepareStatement(sql);

sentencia.setString(1, nif);

sentencia.setString(2, nombre);

sentencia.setString(3, domicilio);

sentencia.setString(4, tlf);

sentencia.setString(5, ciudad);

**int** afectados = sentencia.executeUpdate();

System.***out***.println("Sentencia SQL ejecutada con éxito");

System.***out***.println("Registros afectados: " + afectados);

} **catch** (SQLException e) {

System.***out***.println("Error al añadir nuevo cliente");

System.***out***.println(e.getMessage());

}

// Paso 4: Cerrar la conexión

**try** {

con.close();

} **catch** (SQLException e) {

System.***out***.println("No se ha podido cerrar la BD");

System.***out***.println(e.getMessage());

**return**;

}

System.***out***.println("Se ha cerrado la base de datos");

}

}

Ahora vamos a analizar el código:

En primer lugar, PreparedStatement nos permite crear una sentencia SQL con parámetros que posteriormente podemos reemplazar:

String sql = "INSERT INTO CLIENTE (NIF, NOMBRE, DOMICILIO, TLF, CIUDAD) " +

"VALUES (?, ?, ?, ?, ?)";

Cada interrogación es un parámetro que habrá que sustituir por los datos de un cliente. De esta forma nos sirve la misma cadena de texto para todas las sentencias del mismo tipo que vayamos a ejecutar.

PreparedStatement sentencia;

sentencia = con.prepareStatement(sql);

Luego, hemos creado un objeto de tipo PreparedStatamentasociado a la conexión abierta y le hemos pasado como argumento al constructor la cadena SQL con los parámetros pendientes de reemplazar.

sentencia.setString(1, nif);

sentencia.setString(2, nombre);

sentencia.setString(3, domicilio);

sentencia.setString(4, tlf);

sentencia.setString(5, ciudad);

Por cada una de las interrogaciones (parámetros) dentro de la cadena SQL, tenemos que ejecutar un método *set...* con el siguiente formato:

obj.setxxx(ordenParametro, valorParametro);

Donde xxx hace referencia al tipo de datos (setInt(...), setString(...), setFloat(...), setDouble(...), etc.).

int afectados = sentencia.executeUpdate();

El último paso es ejecutar la sentencia con los métodos executeQuery() o excuteUpdate().