



# Information Systems

## Technologies and Architecture


Javier Tuya  
[Tr ENG] José García Fanjul ([jgfanjul@uniovi.es](mailto:jgfanjul@uniovi.es))  
Software Engineering Research Group  
<http://giis.uniovi.es>



Curso 2020-2021



1



## Contents

- Sample project
  - <https://in2test.lsi.uniovi.es/tools/samples-test-dev/>
- 1. JDBC connection to database (giis.demo.jdbc package)
  - 1. Connections and queries
  - 2. Handling exceptions and parameters
  - 3. Apache Commons DbUtils and Lombok
- 2. Application architecture (giis.demo.tkrun package)
  - 1. MVC pattern and Maven project structure
  - 2. Description of functionality shown as an example
  - 3. Description of the Model, View and Controller code
  - 4. Database configuration

J. Tuya, (2020-2021) Tecnologías y Arquitectura **2**

2

## Part 1: JDBC connection to database

### ■ Objective:

- Review of the mechanisms to access a database using jdbc
- Additional Apache Commons components that simplify access

### ■ Source code in giis.demo.jdbc package

J. Tuya, (2020-2021)

Tecnologías y Arquitectura

3

3

```
package giis.demo.jdbc;
import java.sql.*;

/* Ejemplos de acceso a una base de datos con conexion JDBC y bas
public class DemoJdbc {
    //informacion de conexion a la base de datos utilizada
    public static final String DRIVER="org.sqlite.JDBC";
    public static final String URL="jdbc:sqlite:DemoDB.db";
    //datos para SQLServer:
    //com.microsoft.sqlserver.jdbc.SQLServerDriver
    //jdbc:sqlserver://localhost:1433;DatabaseName=*****;user=*****;password=*****
    private Logger log=Logger.getLogger(DemoJdbc.class);

    * Demo basico de acceso a bases de datos, parte 1:
    public void demo1Basic() {
        try {
            //instalacion de la clase que contiene al driver (basta con hacerlo una vez)
            //el driver (en este caso sqldbc.jar) ha sido descargado y puesto en el classpath
            //Esto normalmente no es necesario desde a partir de JDBC 4.0 los drivers se autoregistran
            Class.forName(DRIVER);
            //Definicion de la cadena de conexion, especifica para cada gestor
            String connString=URL;

            //Ejecuta acciones de actualizacion: insertar datos en una tabla
            Connection cn=DriverManager.getConnection(connString);
            Statement stmt = cn.createStatement();
            try {
                stmt.executeUpdate("drop table if exists test");
            } catch (SQLException e) {
                //ignora excepcion, que se causara si la tabla no existe en la bd (p.e. al ejecutar la primera vez)
            }
            stmt.executeUpdate("create table test(id int not null, id2 int, text varchar(32))");
            stmt.executeUpdate("insert into test(id,id2,text) values(1,null,'abc')");
            stmt.executeUpdate("insert into test(id,id2,text) values(2,9999,'xyz')");
            stmt.close(); //no olvidar cerrar estas cosas
            cn.close();
        }
    }
}
```

Settings:  
-driver: particular for the DBMS used  
-url: specifies which is the DB (syntax dependent on the DBMS)

Class instantiation and connection creation

Statement is the basic object that must exist to manipulate data

SQL execution  
("if exists" in the drop is dependent on the DBMS, may not exist in others)

Do not forget to close the objects connection and statement

J. Tuya, (2020-2021)

Tecnologías y Arquitectura

4

4

## Connections and queries

```
//Consulta todas las filas a partir del resultado de una query SQL
cn=DriverManager.getConnection(connString);
stmt=cn.createStatement();
ResultSet rs=stmt.executeQuery("select id,id2,text from test order by id desc");
while (rs.next()) { //cada vez que se llama
    int id=rs.getInt(1); //obtencion de un valor con un tipo de dato especificado, indicando el numero de columna
    String id2=rs.getString(2); //obtencion de un valor como string, aunque sea entero
    if (rs.wasNull()) //comprobacion de valores nulos (respecto del ultimo get realizado)
        id2="NULO"; // si es nulo puedo hacer un tratamiento especial, en este caso poner un valor
    String text=rs.getString("text"); //obtencion de un valor indicando el nombre de la columna
    log.info("demo18a: "+id+" "+id2+" "+text);
}
rs.close();
stmt.close();
cn.close();
```

As always, create connection and statement

executeQuery to execute SQL that returns data in a ResultSet

Next is the iterator over a ResultSet.  
Each time it is executed, it advances the reading position.  
When it is false it indicates that there is no more data

Each field is obtained with a getX method  
getString always returns the value as string

Caution with reading null values from the DB.

J. Tuya, (2020-2021)

Tecnologías y Arquitectura

5

5

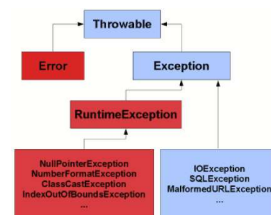
## Exception handling

```
public void demo2TryWithResources() {
    createTable();
    //En un mismo try se pueden poner diferentes sentencias
    try (Connection cn=DriverManager.getConnection(connString);
        Statement stmt=cn.createStatement();
        ResultSet rs=stmt.executeQuery("select id,id2,text from test order by id desc")) {
        while (rs.next()) { //cada vez que se llama rs.next() avanza el cursor a una fila
            int id=rs.getInt(1); //obtencion de un valor con un tipo de dato especificado, indicando el numero de columna
            String id2=rs.getString(2); //obtencion de un valor como string, aunque sea entero
            if (rs.wasNull()) //comprobacion de valores nulos (respecto del ultimo get realizado)
                id2="NULO"; // si es nulo puedo hacer un tratamiento especial, en este caso poner un valor
            String text=rs.getString("text"); //obtencion de un valor indicando el nombre de la columna
            log.info("demo2TryWithResources: "+id+" "+id2+" "+text);
        }
    } catch (SQLException e) {
        throw new UnexpectedException(e);
    }
}
```

You always have to close all the objects that you open,  
Which is sometimes difficult (we have Connection, Statement, ResultSet)  
The "try with resources" does all this in a single try  
Ensuring that whatever happens closes everything  
(no need to use finally)

Thus we only catch the exception once.  
Differentiate types of exceptions:

[https://www.javamex.com/tutorials/exceptions/exceptions\\_hierarchy.shtml](https://www.javamex.com/tutorials/exceptions/exceptions_hierarchy.shtml)



J. Tuya, (2020-2021)

Tecnologías y Arquitectura

6

6

## Parameters

To pass parameters use PreparedStatement (avoids code injection security problems)

```
public void demo3Parameters() {
    createTable();
    //En vez de crear un Statement y pasar el sql en executeQuery,
    //se crea un PreparedStatement con el sql, luego se le ponen los parametros y finalmente se ejecuta
    try (Connection cn=DriverManager.getConnection(URL);
        PreparedStatement pstmt=cn.prepareStatement("select id,id2,text from test where id=?")) {
        pstmt.setInt(1, 2); // pone valor 2 en el primer (y unico) parametro
        try (ResultSet rs=pstmt.executeQuery()) {
            while (rs.next()) {
                log.info("demo3Parameters: "+rs.getInt(1)+" "+rs.getInt(2)+" "+rs.getString(3));
            }
        }
    } catch (SQLException e) {
        throw new UnexpectedException(e);
    }
    //de forma similar se pueden ejecutar acciones de actualizacion sobre el PreparedStatement
}
```

The PreparedStatement is instantiated using ? as a placeholder for the values

Then the values are assigned with a setX statement for each parameter  
The first column in SQL always starts with 1

J. Tuya, (2020-2021)

Tecnologías y Arquitectura

7

7

## Apache Commons DbUtils

```
public void demo4DbUtils() {
    createTable();
    //El siguiente ejemplo define una consulta
    //como una lista de objetos
    //para leer un resultset y poner los resultados en los objetos.
    Connection conn=null;
    List<Entity> pojoList; //lista de objetos
    try {
        conn=DriverManager.getConnection(URL);
        //declara el handler que permitira obtener una lista de objetos de la clase indicada
        BeanListHandler<Entity> beanListHandler=new BeanListHandler<>(Entity.class);
        //Declara el runner que ejecutara la consulta
        QueryRunner runner=new QueryRunner();
        //ejecuta la consulta, el ultimo argumento es el parametro (lista variable si hay mas)
        String sql="select id,id2,text from test where id=?";
        pojoList=runner.query(conn, sql, beanListHandler, 2);
    } catch (SQLException e) {
        throw new UnexpectedException(e);
    } finally {
        DbUtils.closeQuietly(conn); //usar este metodo para cerrar la consulta
    }
}
```

Apache Commons DbUtils can be used to avoid manually stepping through resultsets and each of the columns

It is enough to create a handler in this case for POJO objects (beans).  
In this case, an object of class Entity

and a QueryRunner: the one that executes the query

When executing the runner with the indicated query, it will return a list of objects of the class used with the values obtained from the DB

Own method to close all created objects well

NOTE: It requires defining classes for the entities represented in each table with their corresponding getters and setters

J. Tuya, (2020-2021)

Tecnologías y Arquitectura

8

8

## Apache Con

**IMPORTANT:** When using these components, the Java capitalization convention **must be strictly adhered to**:

- Capitalize all the words that make up an identifier except the first letter of method and variable names.
- Do not use underlines

Also follow these criteria in names of tables and fields of the DB

If we do not define a specific object for the values to be obtained from the DB  
We can do it in a generic way to obtain all the values on a map

```
List<Map<String,Object>> mapList; //lista de maps que seran devueltos por la query
try {
    conn=DriverManager.getConnection(URL);
    String sql="select id,id2,text from test where id?";
    mapList=new QueryRunner().query(conn, sql, new MapListHandler(),1);
} catch (SQLException e) {
    throw new UnexpectedException(e);
} finally {
    DbUtils.closeQuietly(conn);
}
for (Map<String,Object> item : mapList)
    log.info("demo4DbUtils (Map): "+item.get("id")+" "+item.get("id2")+" "+item.get("text"));

//Existen otros tipos de handlers para manejar otros tipos de valores escalares, arrays o listas,
//y metodos de QueryRunner para sentencias sql de actualizacion
//Ver mas documentacion:
//http://commons.apache.org/proper/commons-dbutils/apidocs/index.html
//https://commons.apache.org/proper/commons-dbutils/examples.html
```

In this case we use MapListHandler instead of BeanListHandler.  
Notice the compact way to instantiate the objects and run everything  
In a single line.  
Compare with required code if DbUtils is not used

As the data is read in a map, this would  
be the way to manipulate them

J. Tuyá, (2020-2021)

Tecnologías y Arquitectura

9

9

## Lombok

This would be the way to define the POJO object.  
Include getters and setters for each attribute.  
Add constructor if instantiation is needed  
from program

Using lombok avoids creating manually  
getters and setters  
Simply indicate an annotation at the class level  
if a constructor is required with all the parameters,  
include AllArgsConstructor annotation

```
package giis.demo.jdbc;
public class Entity {
    private Integer id;
    private Integer id2;
    private String text;

    public Integer getId() { return this.id; }
    public Integer getId2() { return this.id2; }
    public String getText() { return this.text; }
    public void setId(Integer value) { this.id=value; }
    public void setId2(Integer value) { this.id2=value; }
    public void setText(String value) { this.text=value; }
}
```

```
package giis.demo.jdbc;
import lombok.*;
@Getter @Setter
public class Entity {
    private Integer id;
    private Integer id2;
}
```

**IMPORTANT:**  
To run from eclipse you need to  
install it in the environment.  
<https://projectlombok.org/setup/eclipse>

More info:  
<https://www.sitepoint.com/declutter-pojos-with-lombok-tutorial/>  
<http://www.baeldung.com/intro-to-project-lombok>

J. Tuyá, (2020-2021)

Tecnologías y Arquitectura

10

10