Guided task on Accessing relational databases from Java with MySQLWorkbench/XAMPP

Preface

The way to access the MySQL Workbench database from Java has changed, since the updated connector has a different way of registering the drivers, plus you have to put something in it to correctly configure the time zone (Or fix it from the server) It will be explained how to access a XAMPP server, but the MySQL Wokbench correction will be maintained so that the program is valid for both database servers.

Once the following task has been completed, both the code and a pdf document explaining the development of the practice will be sent to the task prepared for it in Moodle.

We have, for example,

Car Class

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ficherobd;

/**
 *
 * @author
 */
public class Coche
{
    private String matricula, marca, modelo, color;
    private int año;
    private double precio;

    public Coche(String matricula, String marca, String modelo, String color, int año, double precio) {
        this.matricula = matricula;
        this.marca = marca;
        this.modelo = modelo;
        this.color = color;
        this.año = año;
        this.precio = precio;
    }

    public String getMatricula() {
        return matricula;
    }

    public void setMatricula(String matricula) {
        this.matricula = matricula;
    }

    public String getMarca() {
        return marca;
    }

    public void setMarca(String marca) {
        this.marca = marca;
    }

    public String getModelo() {
        return modelo;
    }
```

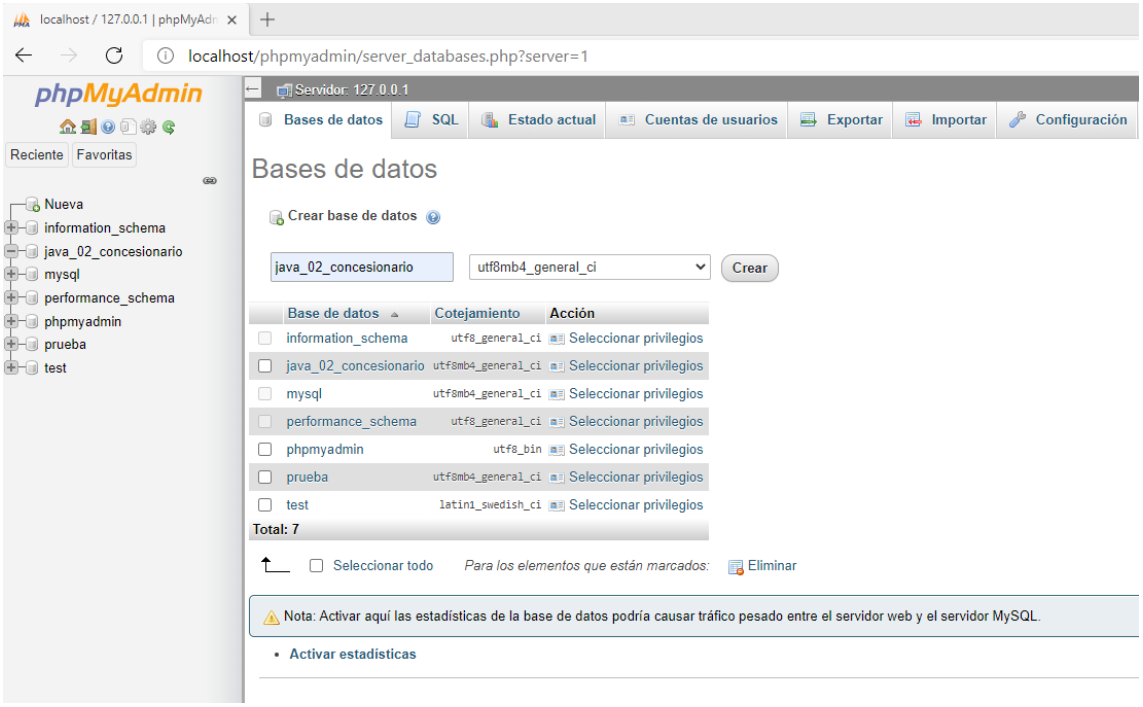
```
46
47      public void setModelo(String modelo) {
48          this.modelo = modelo;
49      }
50
51      public String getColor() {
52          return color;
53      }
54
55      public void setColor(String color) {
56          this.color = color;
57      }
58
59      public int getAño() {
60          return año;
61      }
62
63      public void setAño(int año) {
64          this.año = año;
65      }
66
67      public double getPrecio() {
68          return precio;
69      }
70
71      public void setPrecio(double precio) {
72          this.precio = precio;
73      }
74
75      @Override
76      public String toString() {
77          return "Coche{" + "matricula=" + matricula + ", marca=" + marca + ", modelo=" + modelo +
78                  ", color=" + color + ", a\u00f1o=" + año + ", precio=" + precio + '}';
79      }
80
81
82
83
84
85  }
86
```

This is a class with six attributes, their corresponding setters and getters and a toString method that overrides the default toString method, showing us all the information about the objects of the class that are instantiated.

DatabaseClass

It will be used to create a new database instance (it is understood that the root password is “” in the MySQL Workbench)

Logically, the database must be created on the database server, in this case XAMPP (In the screenshot it is already created, but it shows how it would be done)



```java
package ficherobd;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

/**
 *
 * @author
 */
public class BaseDatos
{
    static Connection conn;
    static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    static final String USER = "root";
    static final String PASS = "Contraseña";
    static final String BD="java_02_concesionario";
    static final String DB_URL = "jdbc:mysql://localhost:3306/"+BD+"?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDate
    private static BaseDatos INSTANCE;

    /**
     * Patrón de diseño singleton
     */
    private BaseDatos()
    {
        try{
            Class.forName(JDBC_DRIVER).newInstance();
            conn=DriverManager.getConnection(DB_URL,USER,PASS);
            if(conn!=null){
                System.out.println("Conexión a la base de datos "+DB_URL+".....CORRECTA");
            }
        }
        catch(SQLException | InstantiationException | IllegalAccessException ex){
            System.err.println("Problemas al conectar"+ex.getMessage());
        }
        catch(ClassNotFoundException ex){
            System.err.println(ex.toString());
        }
    }

    public static BaseDatos getInstance()
    {
        if(INSTANCE == null)
            INSTANCE = new BaseDatos();
        return INSTANCE;
    }


    public Connection getConnection()
    {
        return conn;
    }



}
```

The attributes of the BaseData class (All of them are static)

An attribute of type connection, conn

A final attribute of type String with the Driver that we are going to use in the program, which is included in the following connector:

A final attribute of type String in which the user with whom the database will be accessed will be stored

A final attribute of type String in which the password corresponding to the user with whom it will be linked to the database will be stored.

A final attribute of type String in which the name of the database to which we want to connect will be stored.

A final attribute of type String in which the url address to access the database will be stored. Additionally, if you want to access a database in MysQLWorkbench from Netbeans, you must use

+"?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC";

After the database url.

NOTE: If the driver has not been configured correctly, using the line

static final String DB_URL = "jdbc:mysql://localhost:3306/"+BD+"?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC";

The program shows us the following on the screen:

(Can also be fixed by modifying the my.ini file in MySQL Workbench)

(In XAMPP it is not necessary, in principle, but it is better to put it in case we are going to change the DBMS in the future)

The last attribute is a private object of type BaseData (It is only declared, not instantiated)

The BaseData() constructor method allows us to create a new instance of the driver and achieve a connection using the lines:

Class.forName(JDBC_DRIVER).newInstance();

conn=DriverManager.getConnection(DB_URL,USER,PASS);

where conn is the connection type attribute

(Logically included within a try-catch to control possible exceptions)

The last two methods are:

getInstance, of type BaseData, public and static, which does is create an instance of BaseData if it does not already exist and matches the created instance to the attribute of type BaseData. Return that instance to be able to work with its methods

getConnection, of type Connection, public and static, which returns the connection created in the with attribute.

BBDDMethods.java

```java
package ficherobd;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author
 */
public class MetodosBBDD
{
    public static void crearTablaCoches()
    {
        Connection conexion = BaseDatos.getInstance().getConnection();
        final String sql = "CREATE TABLE IF NOT EXISTS coches "
                + "(matricula varchar(8),marca varchar(40),modelo varchar(40),"
                + "color varchar(40), año int not null,"
                + "precio decimal not null,"
                + "PRIMARY KEY (matricula))";
        try
        {
            Statement sentencia = conexion.createStatement();
            sentencia.executeUpdate(sql);

        }
        catch (SQLException ex)
        {
            System.out.println("Error al crear la tabla");
            System.err.println(ex.getMessage());
        }
    }
```

This class is composed entirely of static methods,

In the createCarTable() method, the line

Connection connection = Database.getInstance().getConnection();

It creates a BaseData instance for me and returns its connection to me, if everything goes well

Then a query is stored that allows the cars table if it does not exist and in the try, the query is executed using the commands:

Statement statement = connection.createStatement();

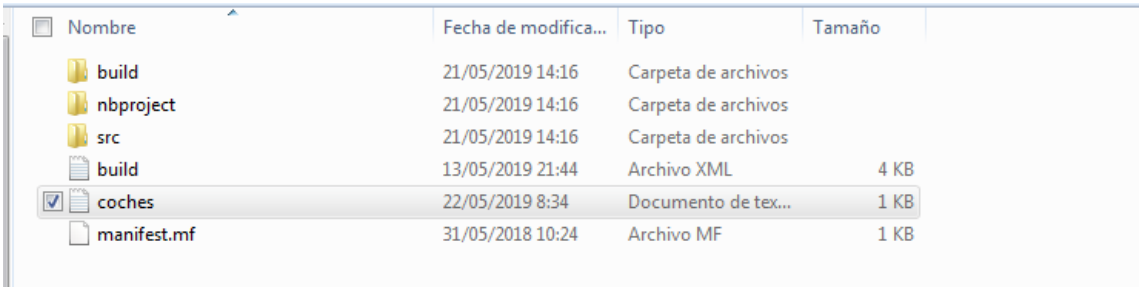
statement.executeUpdate(sql);

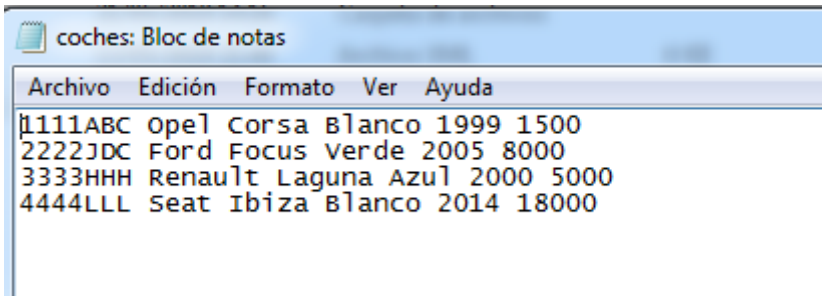
This creates the car table (As long as there are no errors such as those provided in the catch)

```
43
44      public static void cargarCoches()
45      {
46          BufferedReader br = null;
47          String cadena;
48          try
49          {
50              br = new BufferedReader(new FileReader(new File("coches.txt")));
51
52              while ( (cadena = br.readLine()) != null)
53              {
54                  String[] campos = cadena.split(" ");
55                  Coche coche = new Coche(
56                          campos[0], campos[1], campos[2], campos[3],
57                          Integer.parseInt(campos[4]), Double.parseDouble(campos[5]));
58
59                  insertarCoche(coche);
60              }
61          } catch (IOException exc) {
62              System.err.println(exc.getMessage());
63          }finally
64          {
65              try
66              {
67                  if(br!=null)
68                     br.close();
69              }
70              catch(IOException ioe){
71                  System.err.println(ioe.getMessage());
72              }
73          }
74      }
75
```

The loadCars() method requires that there be a cars.txt file to read from to create the Car objects and then to save them to the table by calling the insertCar() method. First, the document is read and separated using the Split by spaces command, each of the fields, then these fields are used to generate the Car object, which will be inserted using the insertCar method

Status of the Coches.txt file:






```java
public static List<Coche> getCoches()
{
    List<Coche> coches = new ArrayList();
    Connection conexion = BaseDatos.getInstance().getConnection();
    String sql = "select * from coches";

    try
    {
        Statement sentencia = conexion.createStatement();
        ResultSet rs = sentencia.executeQuery(sql);

        while (rs.next())
        {
            Coche c = new Coche(rs.getString(1), rs.getString(2),
                    rs.getString(3), rs.getString(4), rs.getInt(5), rs.getDouble(6));
            coches.add(c);
            System.out.println(c);
        }
    }
    catch (SQLException ex)
    {
        System.err.println("Error en el método getCoches");
        System.err.println(ex.getMessage());
    }
    return coches;
}
```

The getcoches() method allows you to query all the information in the MySQL table, creating an instance of the Car class with each of the rows in the table, which will be added to a list of Car objects called cars, which will be which the method returns. It should also be noted that the line System.out.println(c) will write the information of the object

```java
public static boolean existeCoche(String matricula)
{
    Connection conexion = BaseDatos.getInstance().getConnection();
    String sql = "select count(matricula) from coches where matricula = ?";

    try
    {
        PreparedStatement ps = conexion.prepareStatement(sql);
        ps.setString(1, matricula);
        ResultSet rs = ps.executeQuery();
        rs.next();
        if(rs.getInt(1)>0)
            return true;
    }
    catch (SQLException ex)
    {
        System.err.println("Error en el método existeCoche");
        System.err.println(ex.getMessage());
    }

    return false;
}
```

The existsCar() method informs us whether the car whose license plate we pass exists or not. A connection is created to the database and a string of characters is created in which the place that would correspond to the license

plate, we put a question closure (question mark). Then, using the following three lines, we prepare the query and execute it:

PreparedStatementps = connection.prepareStatement(sql);

ps.setString(1, enrollment);

ResultSetrs = ps.executeQuery();

```
127     private static void insertarCoche(Coche c)
128     {
129         if (!existeCoche(c.getMatricula()))
130         {
131             Connection conexion = BaseDatos.getInstance().getConnection();
132             String sql = "insert into coches values (?,?,?,?,?,?)";
133             try
134             {
135                 PreparedStatement ps = conexion.prepareStatement(sql);
136 
137                 ps.setString(1, c.getMatricula());
138                 ps.setString(2, c.getMarca());
139                 ps.setString(3, c.getModelo());
140                 ps.setString(4, c.getColor());
141                 ps.setInt(5, c.getAño());
142                 ps.setDouble(6, c.getPrecio());
143 
144                 ps.executeUpdate();
145             }
146             catch (SQLException exc)
147             {
148                 System.err.println(exc.getMessage());
149             }
150         }
151     }
152 
153 }
154 
```

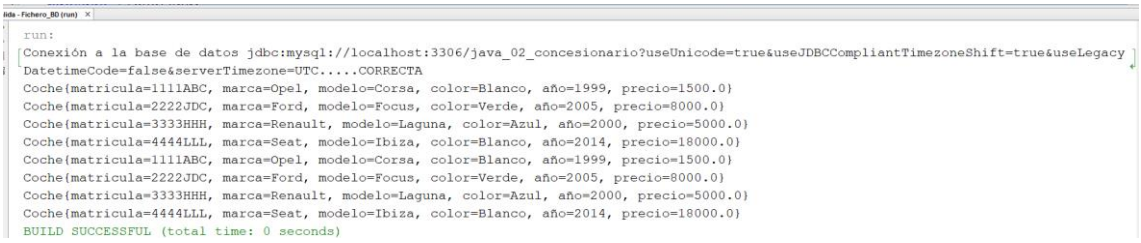
This method allows us to insert a Car into the table, if the car does not exist in the table (comparing it using c.getMatricula()) a connection and a character string that stores a query are created, then as in the previous method, we prepare and we execute.

The main class

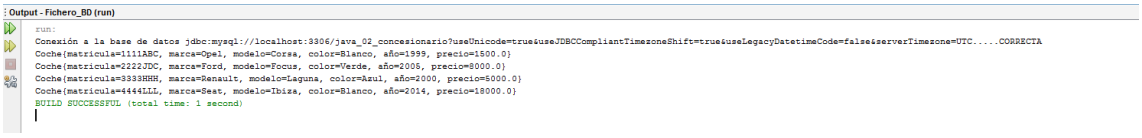
```java
package ficherobd;

/**
 *
 * @author
 */
public class Principal
{

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args)
    {
        MetodosBBDD.crearTablaCoches();
        MetodosBBDD.cargarCoches();
        for(Coche coche: MetodosBBDD.getCoches())
            System.out.println(coche);
    }

}
```

The table is created, the cars are loaded and the list provided by the getCars() method is read, extracting their information through the console.

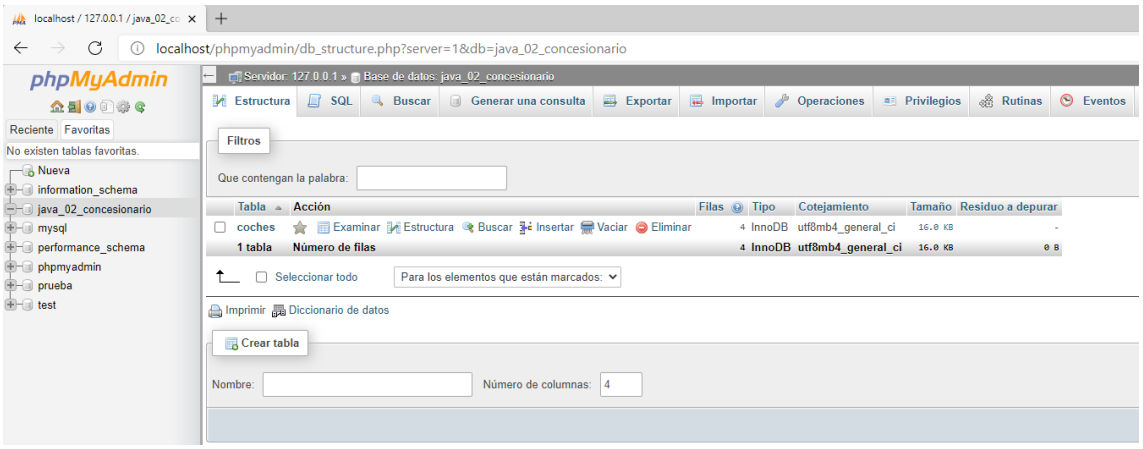
Running this program will produce the following output:

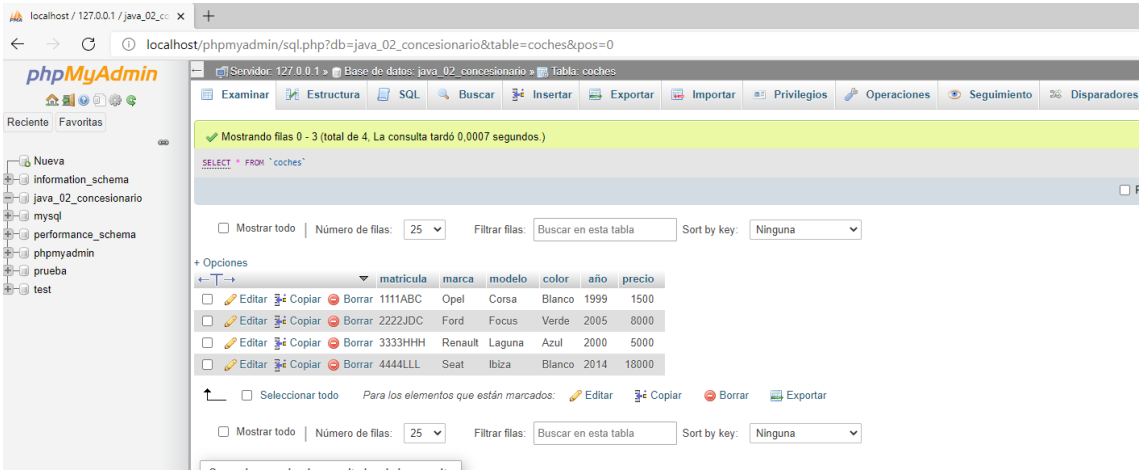


If we remove the line from getCars() in which the cars are written, we have:



We can also examine how the database is on the server.


localhost / 127.0.0.1 / java_02_cc
localhost/phpmyadmin/db_structure.php?server=1&db=java_02_concesionario
phpMyAdmin
Reciente
Favoritas
No existen tablas favoritas.
Nueva
information_schema
java_02_concesionario
mysql
performance_schema
phpmyadmin
prueba
test
Servidor: 127.0.0.1 » Base de datos: java_02_concesionario
Estructura
SQL
Buscar
Generar una consulta
Exportar
Importar
Operaciones
Privilegios
Rutinas
Eventos
Filtros
Que contengan la palabra:
Tabla
Acción
Filas
Tipo
Cotejamiento
Tamaño
Residuo a depurar
coches
Examinar
Estructura
Buscar
Insertar
Vaciar
Eliminar
4
InnoDB
utf8mb4_general_ci
16.0 KB
1 tabla
Número de filas
4
InnoDB
utf8mb4_general_ci
16.0 KB
0 B
Seleccionar todo
Para los elementos que están marcados:
Imprimir
Diccionario de datos
Crear tabla
Nombre:
Número de columnas:
4


localhost / 127.0.0.1 / java_02_cc
localhost/phpmyadmin/sql.php?db=java_02_concesionario&table=coches&pos=0
phpMyAdmin
Reciente
Favoritas
Nueva
information_schema
java_02_concesionario
mysql
performance_schema
phpmyadmin
prueba
test
Servidor: 127.0.0.1 » Base de datos: java_02_concesionario » Tabla: coches
Examinar
Estructura
SQL
Buscar
Insertar
Exportar
Importar
Privilegios
Operaciones
Seguimiento
Disparadores
Mostrando filas 0 - 3 (total de 4, La consulta tardó 0,0007 segundos.)
SELECT * FROM `coches`
Mostrar todo
Número de filas:
25
Filtrar filas:
Buscar en esta tabla
Sort by key:
Ninguna
+ Opciones
matricula
marca
modelo
color
año
precio
Editar Copiar Borrar 1111ABC Opel Corsa Blanco 1999 1500
Editar Copiar Borrar 2222JDC Ford Focus Verde 2005 8000
Editar Copiar Borrar 3333HHH Renault Laguna Azul 2000 5000
Editar Copiar Borrar 4444LLL Seat Ibiza Blanco 2014 18000
Seleccionar todo
Para los elementos que están marcados:
Editar
Copiar
Borrar
Exportar
Mostrar todo
Número de filas:
25
Filtrar filas:
Buscar en esta tabla
Sort by key:
Ninguna