Consultar en una Base de Datos una Lista

SQL -- Sentencia

drop database if exists mercado;

create database mercado;

use mercado;

drop table if exists proveedores;

create table proveedores (PROV\_ID integer NOT NULL, PROV\_NOMBRE varchar(40) NOT NULL, CALLE varchar(40) NOT NULL, CIUDAD varchar(20) NOT NULL, PAIS varchar(2) NOT NULL, CP varchar(5), PRIMARY KEY (PROV\_ID));

insert into proveedores values(49, 'PROVerior Coffee', '1 Party Place', 'Mendocino', 'CA', '95460'); insert into proveedores values(101, 'Acme, Inc.', '99 mercado CALLE', 'Groundsville', 'CA', '95199'); insert into proveedores values(150, 'The High Ground', '100 Coffee Lane', 'Meadows', 'CA', '93966');

create table CAFES (CAF\_NOMBRE varchar(32) NOT NULL, PROV\_ID int NOT NULL, PRECIO numeric(10,2) NOT NULL, VENTAS integer NOT NULL, TOTAL integer NOT NULL, PRIMARY KEY (CAF\_NOMBRE), FOREIGN KEY (PROV\_ID) REFERENCES PROVEEDORES(PROV\_ID));

insert into CAFES values('Colombian', 00101, 7.99, 0, 0); insert into CAFES values('French\_Roast', 00049, 8.99, 0, 0); insert into CAFES values('Espresso', 00150, 9.99, 0, 0); insert into CAFES values('Colombian\_Decaf', 00101, 8.99, 0, 0); insert into CAFES values('French\_Roast\_Decaf', 00049, 9.99, 0, 0);

CLASE CAFE

package tema3.transparencias;

import java.io.IOException; import java.sql.Connection; import java.sql.PreparedStatement; import java.sql.ResultSet; import java.sql.SQLException; import java.sql.Statement;

/\*\*

@descrition clase Café

\*/

public class Cafes {

// Consultas a realizar en BD  
private static final String SELECT\_CAFES\_QUERY = "select CAF\_NOMBRE, PROV\_ID, PRECIO, VENTAS, TOTAL from CAFES";  
// En una consulta parametrizada ponemos interrogaciones en los valores que  
// aún desconocemos  
private static final String UPDATE\_VENTAS\_CAFE = "update CAFES set VENTAS = ? where CAF\_NOMBRE = ?";  
  
/\*\*  
 \* Metodo que muestra por pantalla los datos de la tabla cafes  
 \*   
 \* @param con  
 \* @throws SQLException  
 \*/  
public void verTabla() {  
 /\* Conexión a la Base de Datos \*/  
 Connection con = null;  
 /\* Sentencia sql \*/  
 Statement stmt = null;  
 /\* Conjunto de Resultados a obtener de la sentencia sql \*/  
 ResultSet rs = null;  
 try {  
 con = new Utilidades1().getConnection();  
 // Creación de la sentencia  
 stmt = con.createStatement();  
 // Ejecución de la consulta y obtención de resultados en un  
 // ResultSet  
 rs = stmt.executeQuery(SELECT\_CAFES\_QUERY);  
  
 // Recuperación de los datos del ResultSet  
 while (rs.next()) {  
 String coffeeName = rs.getString("CAF\_NOMBRE");  
 int supplierID = rs.getInt("PROV\_ID");  
 float PRECIO = rs.getFloat("PRECIO");  
 int VENTAS = rs.getInt("VENTAS");  
 int total = rs.getInt("TOTAL");  
 System.out.println(coffeeName + ", " + supplierID + ", "  
 + PRECIO + ", " + VENTAS + ", " + total);  
 }  
  
 } catch (IOException e) {  
 // Error al leer propiedades  
 // En una aplicación real, escribo en el log y delego  
 System.err.println(e.getMessage());  
 } catch (SQLException sqle) {  
 // En una aplicación real, escribo en el log y delego  
 System.err.println(sqle.getMessage());  
 } finally {  
 try {  
 // Liberamos todos los recursos pase lo que pase  
 if (rs != null) {  
 rs.close();  
 }  
 if (stmt != null) {  
 stmt.close();  
 }  
 if (con != null) {  
 Utilidades1.closeConnection(con);  
 }  
 } catch (SQLException sqle) {  
 // En una aplicación real, escribo en el log, no delego porque  
 // es error al liberar recursos  
 }  
 }  
  
}  
  
/\*\*  
 \* Método que actualiza las ventas de un café con un PreparedStatement  
 \*   
 \* @param cafe  
 \* @param ventas  
 \*/  
public void actualizarVentasCafe(String cafe, int ventas) {  
  
 /\* Conexión a la Base de Datos \*/  
 Connection con = null;  
 /\* Sentencia Preparada sql \*/  
 PreparedStatement stmt = null;  
   
 try {  
 con = new Utilidades1().getConnection();  
  
 // Creamos un objeto PreparedStatement pasándole nuestra consulta  
 // parametrizada  
 stmt = con.prepareStatement(UPDATE\_VENTAS\_CAFE);  
 // Le damos valoes a los parámetros de la consulta, indicando número  
 // de parámetro  
 // Y utilizando un método adecuado según el tipo de datos  
 stmt.setFloat(1, ventas);  
 stmt.setString(2, cafe);  
  
 // método executeUpdate para ejecutar inserciones, actualizaciones y  
 // borrado de datos  
 stmt.executeUpdate();  
   
  
 } catch (IOException e) {  
 // Error al leer propiedades  
 // En una aplicación real, escribo en el log y delego  
 System.err.println(e.getMessage());  
   
 } catch (SQLException sqle) {  
 // En una aplicación real, escribo en el log y delego  
 System.err.println(sqle.getMessage());  
   
 } finally {  
 try {  
 // Liberamos todos los recursos pase lo que pase  
  
 if (stmt != null) {  
 stmt.close();  
 }  
 if (con != null) {  
 Utilidades1.closeConnection(con);  
 }  
 } catch (SQLException sqle) {  
 // En una aplicación real, escribo en el log, no delego porque  
 // es error al liberar recursos  
 }  
 }  
}

}

Clase Conexion

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

public class Conexion { public String jdbc\_driver = "com.mysql.cj.jdbc.Driver"; public String user = "admin"; public String pass = "admin"; public String ip = "192.168.56.101"; public String port = "3306"; public String database\_name = "acceso"; public String params = "?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC&useSSL=false";

public Conexion(String jdbc\_driver, String user, String pass, String ip, String port, String database\_name, String params) {  
 this.jdbc\_driver = jdbc\_driver;  
 this.user = user;  
 this.pass = pass;  
 this.ip = ip;  
 this.port = port;  
 this.database\_name = database\_name;  
 this.params = params;  
}  
  
public Conexion(String user, String pass, String database\_name) {  
 this.user = user;  
 this.pass = pass;  
 this.database\_name = database\_name;  
}  
  
public Conexion(String database\_name) {  
 this.database\_name = database\_name;  
}  
  
public Conexion(){}  
  
public Connection getConnection(){  
 Connection conn = null;  
 PreparedStatement stmt = null;  
  
 String url = "jdbc:mysql://" + this.ip + ":" + this.port + "/" + this.database\_name + this.params;  
 try {  
 conn = DriverManager.getConnection(url,this.user,this.pass);  
  
 } catch (SQLException e) {  
 throw new RuntimeException(e);  
 }  
  
 return conn;  
}

}

CLASE CONNECT

package pruebas;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.List;

public class Connect {

// JDBC driver name and database URL

static final String JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";

static final String DB\_URL = "jdbc:mysql://localhost/xilften?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC";

// Database credentials

static final String USER = "root";

static final String PASS = "system";

public static void main(String[] args) {

Connection conn = null;

PreparedStatement stmt = null;

try{

//paso 2: Register JDBC driver a partir de JDK 6 ya no hace falta

//Class.forName(JDBC\_DRIVER);

//paso 3: Open a connection

System.out.println("Connecting to database...");

conn = DriverManager.getConnection(DB\_URL,USER,PASS);

//paso 4: Execute a query

System.out.println("Creating statement...");

stmt = conn.prepareStatement("select \* from GENEROS");

//stmt.setFloat(1, (float) 1.1);

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

String nombreGenero = rs.getString("nombreGenero");

double idGen = rs.getFloat("idGenero");

String descripcionGenero = rs.getString("descripcionGenero");

//Genero gen = new Genero(idGen, nombreGenero, descripcionGenero);

System.out.print("ID: " + idGen);

System.out.print(", name: " + nombreGenero);

System.out.println(", desc: " + descripcionGenero);

}

//STEP 6: Clean-up environment

rs.close();

stmt.close();

conn.close();

}catch(SQLException se){

//Handle errors for JDBC

se.printStackTrace();

}catch(Exception e){

//Handle errors for Class.forName

e.printStackTrace();

}finally{

//finally block used to close resources

try{

if(stmt!=null)

stmt.close();

}catch(SQLException se2){

}// nothing we can do

try{

if(conn!=null)

conn.close();

}catch(SQLException se){

se.printStackTrace();

}//end finally try

}//end try

System.out.println("Goodbye!");

}//end main

}

Archivo XML

mysql-properties.xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">

<properties>

<entry key="dbms">mysql</entry>

<entry key="driver">com.mysql.jdbc.Driver</entry>

<entry key="database\_name">mercado</entry>

<entry key="user\_name">root</entry>

<entry key="password">system</entry>

<entry key="server\_name">localhost</entry>

<entry key="port\_number">3306</entry>

</properties>

CLASE PruebaUtilidades

package tema3.transparencias;

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

/\*\*

@descrition Clase que prueba la conexión a una BD utilizando la clase Driver Manager

\*/

public class PruebaUtilidades {

/\* TODO: Para que funcione hay que instalar el driver mysql  
 \* 1. Ejecutar el .msi que hay en la carpeta software  
 \* 2. El .jar del driver se ubica en C:\Program Files (x86)\MySQL\MySQL Connector J  
 \* 3. Ir a propiedades del proyecto --> Java Build Path --> libaries --> Add External Jar --> Incluir el .jar anterior  
 \* 4. Al ejecutar en Run as --> Run Configuration --> arguments --> Tema3\tema3\transparencias\mysql-properties.xml  
 \* 5. En mysql-properties.xml ir cambiando el nombre de la BD en los ejercicios según la que queramos usar  
 \*/  
 public static void main(String[] args) {  
 Utilidades myConexionDriverManager;  
 Connection myConnection = null;  
 if (args[0] == null) {  
 System.err.println("Properties file not specified at command line");  
 return;  
 } else {  
 try {  
 System.out.println("Reading properties file " + args[0]);  
 myConexionDriverManager = new Utilidades(args[0]);  
 } catch (Exception e) {  
 System.err.println("Problem reading properties file " + args[0]);  
 e.printStackTrace();  
 return;  
 }  
 }  
  
 try {  
 myConnection = myConexionDriverManager.getConnection();   
  
 } catch (SQLException e) {  
 e.printStackTrace(System.err);  
 } catch (Exception e) {  
 e.printStackTrace(System.err);  
 } finally {  
 Utilidades.closeConnection(myConnection);  
 }  
  
 }

}

CLASE Utilidades

package tema3.transparencias;

import java.io.FileNotFoundException; import java.io.IOException; import java.nio.file.Files; import java.nio.file.Paths; import java.sql.Connection; import java.sql.DriverManager; import java.sql.SQLException; import java.util.InvalidPropertiesFormatException; import java.util.Properties;

/\*\*

\*

@description Clase que establece una conexión a BD utilizando la clase DriverManager. lee los datos de u archivo de propiedades

\*/ public class Utilidades {

public String dbms;  
public String dbName;  
public String userName;  
public String password;  
public String urlString;  
  
private String driver;  
private String serverName;  
private int portNumber;  
private Properties prop;  
public Utilidades(String propertiesFileName)  
 throws FileNotFoundException, IOException,  
 InvalidPropertiesFormatException {  
 super();  
 this.setProperties(propertiesFileName);  
}  
  
/\*\*  
 \* Asignación de propiedades de conexión de xml a atributos de clase  
 \* @param fileName  
 \* @throws FileNotFoundException  
 \* @throws IOException  
 \* @throws InvalidPropertiesFormatException  
 \*/  
private void setProperties(String fileName) throws IOException, InvalidPropertiesFormatException {  
 this.prop = new Properties();  
 prop.loadFromXML(Files.newInputStream(Paths.get(fileName)));  
  
 this.dbms = this.prop.getProperty("dbms");  
 this.driver = this.prop.getProperty("driver");  
 this.dbName = this.prop.getProperty("database\_name");  
 this.userName = this.prop.getProperty("user\_name");  
 this.password = this.prop.getProperty("password");  
 this.serverName = this.prop.getProperty("server\_name");  
 this.portNumber = Integer  
 .parseInt(this.prop.getProperty("port\_number"));  
  
 System.out.println("Set the following properties:");  
 System.out.println("dbms: " + dbms);  
 System.out.println("driver: " + driver);  
 System.out.println("dbName: " + dbName);  
 System.out.println("userName: " + userName);  
 System.out.println("serverName: " + serverName);  
 System.out.println("portNumber: " + portNumber);  
  
}  
  
/\*\*  
 \* Conexion a Base de Datos  
 \*   
 \* @return  
 \* @throws SQLException  
 \*/  
public Connection getConnection() throws SQLException {  
  
 Connection conn = null;  
 Properties connectionProps = new Properties();  
 connectionProps.put("user", this.userName);  
 connectionProps.put("password", this.password);  
  
 if (this.dbms.equals("mysql")) {  
 /\* Solicito a DriverManager una conexión con la base de datos \*/  
 /\*  
 \* Para identificar el controldador a usar se le proporciona una  
 \* URL, si no lo encuentra lanza SQLException  
 \*/  
 /\* formato de URL: jdbc:[host][:port]/[database] \*/  
 /\*  
 \* La URL varía según el gestor de BD,  
 \* jdbc:mysql://127.0.0.1:3306/libros,  
 \* jdbc:oracle:thin:@192.168.239.142:1521:libros  
 \*/  
 conn = DriverManager.getConnection("jdbc:" + this.dbms + "://"  
 + this.serverName + ":" + this.portNumber + "/" + this.dbName,  
 connectionProps);  
 } else if (this.dbms.equals("derby")) {  
 conn = DriverManager.getConnection("jdbc:" + this.dbms + ":"  
 + this.dbName + ";create=true", connectionProps);  
 }  
 System.out.println("Connectado a BD");  
 return conn;  
}  
  
/\*\*  
 \* Cierre de conexión a BD  
 \* @param connArg  
 \*/  
public static void closeConnection(Connection connArg) {  
 System.out.println("Releasing all open resources ...");  
 try {  
 if (connArg != null) {  
 connArg.close();  
 connArg = null;  
 }  
 } catch (SQLException sqle) {  
 System.err.println(sqle);  
 }  
}

}

CLASE Utilidades1

package tema3.transparencias;

import java.io.FileNotFoundException; import java.io.IOException; import java.nio.file.Files; import java.nio.file.Paths; import java.sql.Connection; import java.sql.DriverManager; import java.sql.SQLException; import java.util.InvalidPropertiesFormatException; import java.util.Properties;

/\*\*

\*

@description Clase que establece una conexión a BD utilizando la clase DriverManager. lee los datos de u archivo de propiedades

\*/ public class Utilidades1 {

public String dbms;  
public String dbName;  
public String userName;  
public String password;  
public String urlString;  
  
private String driver;  
private String serverName;  
private int portNumber;  
private Properties prop;  
  
private static final String PROPERTIES\_FILE="Tema3\\tema3\\transparencias\\mysql-properties.xml";  
public Utilidades1()  
 throws FileNotFoundException, IOException,  
 InvalidPropertiesFormatException {  
 super();  
 this.setProperties(PROPERTIES\_FILE);  
}  
  
/\*\*  
 \* Asignación de propiedades de conexión de xml a atributos de clase  
 \* @param fileName  
 \* @throws FileNotFoundException  
 \* @throws IOException  
 \* @throws InvalidPropertiesFormatException  
 \*/  
private void setProperties(String fileName) throws IOException, InvalidPropertiesFormatException {  
 this.prop = new Properties();  
 prop.loadFromXML(Files.newInputStream(Paths.get(fileName)));  
  
 this.dbms = this.prop.getProperty("dbms");  
 this.driver = this.prop.getProperty("driver");  
 this.dbName = this.prop.getProperty("database\_name");  
 this.userName = this.prop.getProperty("user\_name");  
 this.password = this.prop.getProperty("password");  
 this.serverName = this.prop.getProperty("server\_name");  
 this.portNumber = Integer  
 .parseInt(this.prop.getProperty("port\_number"));  
  
 System.out.println("Set the following properties:");  
 System.out.println("dbms: " + dbms);  
 System.out.println("driver: " + driver);  
 System.out.println("dbName: " + dbName);  
 System.out.println("userName: " + userName);  
 System.out.println("serverName: " + serverName);  
 System.out.println("portNumber: " + portNumber);  
  
}  
  
/\*\*  
 \* Conexion a Base de Datos  
 \*   
 \* @return  
 \* @throws SQLException  
 \*/  
public Connection getConnection() throws SQLException {  
  
 Connection conn = null;  
 Properties connectionProps = new Properties();  
 connectionProps.put("user", this.userName);  
 connectionProps.put("password", this.password);  
  
 if (this.dbms.equals("mysql")) {  
 /\* Solicito a DriverManager una conexión con la base de datos \*/  
 /\*  
 \* Para identificar el controldador a usar se le proporciona una  
 \* URL, si no lo encuentra lanza SQLException  
 \*/  
 /\* formato de URL: jdbc:[host][:port]/[database] \*/  
 /\*  
 \* La URL varía según el gestor de BD,  
 \* jdbc:mysql://127.0.0.1:3306/libros,  
 \* jdbc:oracle:thin:@192.168.239.142:1521:libros  
 \*/  
 conn = DriverManager.getConnection("jdbc:" + this.dbms + "://"  
 + this.serverName + ":" + this.portNumber + "/" + this.dbName,  
 connectionProps);  
 } else if (this.dbms.equals("derby")) {  
 conn = DriverManager.getConnection("jdbc:" + this.dbms + ":"  
 + this.dbName + ";create=true", connectionProps);  
 }  
 System.out.println("Connectado a BD");  
 return conn;  
}  
  
/\*\*  
 \* Cierre de conexión a BD  
 \* @param connArg  
 \*/  
public static void closeConnection(Connection connArg) {  
 System.out.println("Releasing all open resources ...");  
 try {  
 if (connArg != null) {  
 connArg.close();  
 connArg = null;  
 }  
 } catch (SQLException sqle) {  
 System.err.println(sqle);  
 }  
}  
  
/\*\*  
 \* Metodo para imprimir la información de una Excepción SQL y poder depurar errores fácilmente  
 \* @param ex  
 \*/  
public static void printSQLException(SQLException e) {  
   
 while (e != null) {  
 if (e instanceof SQLException) {  
 //Estado ANSI  
 e.printStackTrace(System.err);  
 System.err.println("SQLState: "  
 + ((SQLException) e).getSQLState());  
 //Códio de error propio de cada gestor de BD  
 System.err.println("Error Code: "  
 + ((SQLException) e).getErrorCode());  
 //Mensaje textual  
 System.err.println("Message: " + e.getMessage());  
  
 //Objetos desencadenantes de la excepción  
 Throwable t = e.getCause();  
 while (t != null) {  
 System.out.println("Cause: " + t);  
 t = t.getCause();  
 }  
 //Cualquier otra excepción encadenada  
 e = e.getNextException();   
   
 }  
 }  
}

}