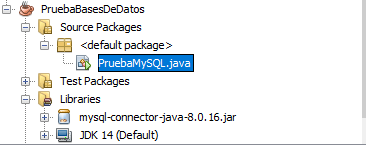


En el programa Java, debemos tener un conector:



Programa de acceso a la base de datos desde Java:

/\*\*

\*

\* @author adelapresa

\* Programa de prueba para conexión a una base de datos de MySQL, trabajando casi todo

\* desde el código fuente de Java

\* Presupone que el servidor de base de datos está arrancado, disponible,

\* en el puerto por defecto. (Por ejemplo, en el XAMPP que tenemos en el documento anterior)

\* El usuario y password de conexión con la base de datos debe cambiarse por el

\* adecuado a nuestras necesidades.

\* Al añadir el código que borra la base de datos, hace que no sea necesario haber definido

\* previamente la estructura de la misma, puesto que todo el control del programa está ahora en el lado del

\* lenguaje anfitrión. Sin embargo, sí es necesario que haya sido definida para que no nos dé problemas la primera

\* vez -puesto que no hay nada que borrar.

\*

\* Creamos una base de datos prueba

\* tiene una tabla acreedores con tres campos, de esta manera:

\* numacree(número del acreedor)

\* nombacree(nombre del acreedor)

\* dirección

\*

\* Relleno la tabla con datos y compruebo si puedo borrar uno de ellos

\*

\* Después se crea otra tabla en la base de datos, llamada clientes, con una estructura

\* bastante similar:

\* numcliente

\* nombcliente

\* dirección

\*/

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

/\*\*

\* Clase de prueba de conexión con una base de datos MySQL

\*/

public class PruebaMySQL {

/\*\*

\* Crea una instancia de la clase MySQL y realiza todo el código

\* de conexión, consulta y muestra de resultados.

\*/

public PruebaMySQL() throws SQLException

{

// Se mete todo en un try por los posibles errores de MySQL

try

{

// Se registra el Driver de MySQL

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

} catch (Exception ex) {

// handle the error

System.out.println(ex.getMessage());

}

// Se obtiene una conexión con la base de datos. Hay que

// cambiar el usuario "root" y la clave ""(Ni que decir tiene que en un entorno de //producción real, la contraseña no podría ser un campo vacío) por las

// adecuadas a la base de datos que estemos usando.

Connection conexion = DriverManager.getConnection (

"jdbc:mysql://localhost/prueba","root", "");

//Si quisiéramos acceder a una base de datos en MySQL Workbench, sería necesario modificar la anterior línea de la siguiente manera:

//"jdbc:mysql://localhost/prueba"+"?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC","root", "");

// Se crea un Statement, para realizar la consulta y se declara que pueda actualizarse

Statement s = conexion.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE,ResultSet.CONCUR\_UPDATABLE);

//Vamos a borrar la base de datos para que lo haga desde cero cada vez que empieza y no nos dé errores

//Que será el que luego insertaremos (Para que esto funcione, la base de datos debe haber sido creada previamente

//pero no nos hace falta hacer nada más)

s.executeUpdate("DROP DATABASE prueba");

s.executeUpdate("CREATE DATABASE prueba");

s.executeUpdate("USE prueba");

s.executeUpdate("CREATE TABLE acreedores(numacree INT, nombacree VARCHAR(25),dirección VARCHAR(25), PRIMARY KEY(numacree))");

//La siguiente línea borraría todos los elementos de la tabla acreedores si no la tuviésemos ya borrada al borrar la base de datos

//s.executeUpdate("DELETE FROM acreedores");

//Introduce datoS

s.executeUpdate("INSERT INTO acreedores " + "VALUES (1,'Recaredo', 'General Dávila 27')");

s.executeUpdate("INSERT INTO acreedores " + "VALUES (2,'Chindasvinto', 'General Dávila 17')");

s.executeUpdate("INSERT INTO acreedores " + "VALUES (3,'Leovigildo', 'General Dávila 7')");

// Se realiza la consulta. Los resultados se guardan en el

// ResultSet rs

ResultSet rs = s.executeQuery ("SELECT \* FROM acreedores");

// Se recorre el ResultSet, mostrando por pantalla los resultados.

while (rs.next())

{

System.out.println (rs.getInt ("numacree") + " " + rs.getString (2)+

" " + rs.getString(3));

//Vamos ahora a borrar el segundo de los registros insertados

//mediante el método ResultSet

if (rs.getString(2).equals("Chindasvinto")){

rs.deleteRow();//Esta orden me borra el registro cuyo segundo valor sea Chindasvinto

}

}

System.out.println("Ahora vamos a visualizar el listado de acreedores para comprobar que efectivamente se ha borrado");

ResultSet rs2=s.executeQuery("select numacree, nombacree, dirección from acreedores");

while (rs2.next())

{

System.out.println (rs2.getInt ("numacree") + " " + rs2.getString (2)+

" " + rs2.getString(3));}

//Creamos una tabla nueva

s.executeUpdate("CREATE TABLE clientes(numcliente INT, nombcliente VARCHAR(25),dirección VARCHAR(25))");

// Se cierra la conexión con la base de datos.

conexion.close();}

//Vamos a crear una tabla nueva clientes

/\*\*

\* Método principal, instancia una clase PruebaMySQL

\*

\* @param args the command line arguments

\*/

public static void main(String[] args) throws SQLException

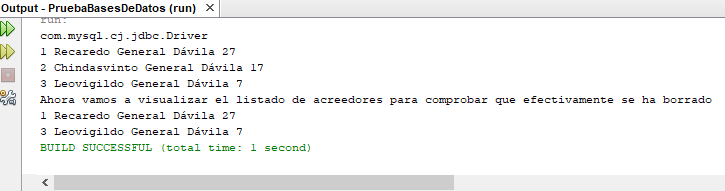
{

new PruebaMySQL();

}

}

La ejecución del anterioir programa produce la siguiente salida:



Podemos comprobar las modificaciones que hay en la propia base de datos –en phpmyadmin-

