Practice 2.2: modify MySQL databases from applications

OBJECTIVE: insert, update, etc. a relational database from an application

PRACTICE: We will create DATA DEFINITION LANGUAGE(DDL) SENTENCES: CREATE, ALTER, DROP using method excuteUpdate() :

1. Create class ConnectTEST: to access DB test in MySql
2. Create method openConnection() and closeConnection():

**import** java.sql.\*;

**public** **class** ConnectTEST {

**private** Connection con;

**void** connect() {

**try** {

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

con=DriverManager.*getConnection*("jdbc:mysql://localhost/test","root","");

//if the connection fails, there is an exception here

System.***out***.println("CONNECTED TO DATABASE "+ "jdbc:mysql://localhost/test");

}**catch**(Exception e) {

e.printStackTrace();

}

}

**void** closeConnection() {

**try** {

con.close();

System.***out***.println("jdbc:mysql://localhost/test "+ "closed");

}**catch**(Exception e) {

e.printStackTrace();

}

}

}

1. See if the connection works: (xampp must be running): Create clase ConnectingToTest

**public** **class** UseConnectTest {

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

// **TODO** Auto-generated method stub

ConnectTEST ct=**new** ConnectTEST();

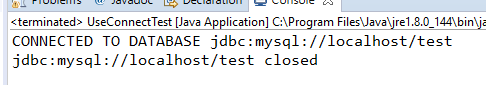
ct.connect();

ct.closeConnection();

}

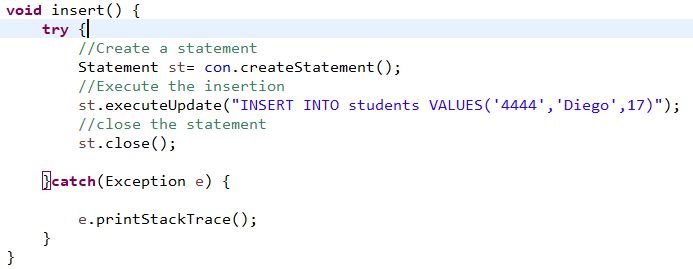
}

If it works, you must get something like this:

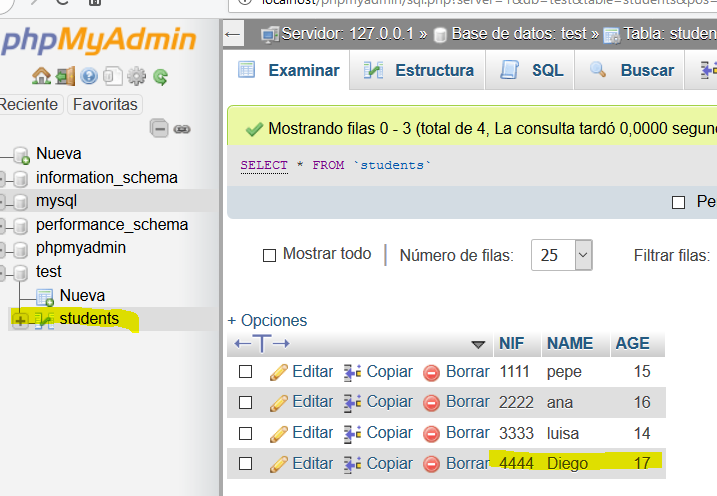


1. Now, we will write a method to INSERT values in the table STUDENTS:

If we write the code:



We get a new record in out database:



We will modify the code of insert() a little, asking the values to the user first:

**void** insert() {

Scanner sc=**new** Scanner(System.***in***);

String nif,name;

**int** age;

**try** {

System.***out***.println("Write the information of the new student for the database:");

System.***out***.println("NIF?");

nif=sc.nextLine();

System.***out***.println("Name?");

name=sc.nextLine();

System.***out***.println("age?");

age=Integer.*parseInt*(sc.nextLine());

//Create a statement

Statement st= con.createStatement();

//Execute the insertion

//st.executeUpdate("INSERT INTO students VALUES('4444','Diego',17)");

st.executeUpdate("INSERT INTO students VALUES('"+nif+"','"+name+"',"+age+")");

//String sql=String.format("INSERT INTO students VALUES('%s','%s',%d)",nif,name,age);

//st.executeUpdate(sql);

//close the statement

st.close();

}**catch**(Exception e) {

e.printStackTrace();

}

}

Try the method:

**public** **class** UseConnectTest {

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

// **TODO** Auto-generated method stub

ConnectTEST ct=**new** ConnectTEST();

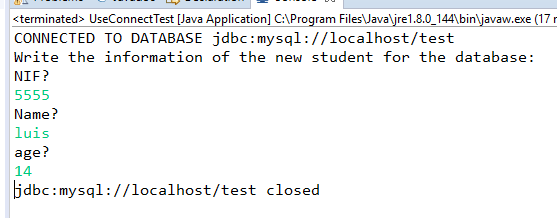
ct.connect();

ct.insert();

ct.closeConnection();

}

}





1. Now, let’s update some value, for example luisa’s age:

**void** update() {

**try** {

System.***out***.println("Updating....");

//Create a statement

Statement st= con.createStatement();

//Execute the updating

st.executeUpdate("UPDATE students SET AGE=20 WHERE students.NIF='3333'");

//close the statement

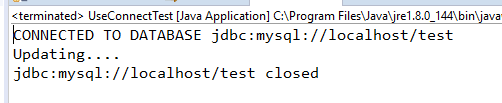
st.close();

}**catch**(Exception e) {

e.printStackTrace();

}

}





A more general method to update the age of a given nif can be like this:

**void** update(String nif,**int** age) {

String sql;

**try** {

System.***out***.println("Updating....");

//Create a statement

Statement st= con.createStatement();

//Execute the updating

//st.executeUpdate("UPDATE students SET AGE=20 WHERE students.NIF='3333'");

sql=String.*format*("UPDATE students SET AGE=%d WHERE students.NIF='%s'",age,nif);

st.executeUpdate(sql);

//close the statement

st.close();

}**catch**(Exception e) {

e.printStackTrace();

}

}

If we try it, we can see how the row is modified:

**public** **class** UseConnectTest {

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

ConnectTEST ct=**new** ConnectTEST();

ct.connect();

//ct.insert();

ct.update("4444",25);

ct.closeConnection();

}

}

