Design of user interfaces

Visualizador de la estructura de una base de datos MySQL

2018/19 course - Practice 9

David Bohmann
Petr Lukašík

November 7, 2018

Contents

1. Assigment

In some applications, it is necessary for the user to enter information that should not be seen by other people, such as access codes. In this case it is important that each character typed by the user becomes another character, usually an asterisk. Swing provides the JPasswordField class that performs this task automatically. The appearance is the same as a text field but the characters are replaced by a character (can be set) as it is written.

It is also usually necessary in some applications, to select from a list that is showing user one or more options. In this case Swing's JList class is the one that implements this functionality. Note that unlike other Swing classes available where users can select from a list of options, the JList class allows you to create, display and modify the selection scheme of options dynamically. In the case of the JComboBox class or the JCheckBox yet the possible options must be known at the time of program coding.

The objective of this practice is to make use of the previous classes, as well as the views in previous practices, to implement an application that allows us to show the structure of tables and fields of a MySQL database. Access to the database will use the corresponding driver as well as the classes included in the java.sql package. As in previous practices in the section 1.1 "Implementation notes" there will be examples of use of these classes.

The functionality that the application to develop in this practice will be the following:

- Have a text box and another key where the user enters the username and password that allows access to the database.
- Make the connection to the database using the username and password entered as well as report any errors that may occur.
- Show a list with the name of tables that make up the database.
- Allow through the use of toogle buttons where the user decides which selection mode (simple, by interval or by multiple intervals) to use to choose the tables.
- Allow the user to deselect the entire selection.
- Show in a list the fields of the selected tables, prefixing the name of the table separated by a point, that is, tabla.campo.

1.1 Implementation notes

The access of a Java application to a database is done through the JDBC API. This API makes use of a specific driver for each database (Oracle, MySQL, PostgreSQL, ...) that provides the connection with said database to perform the usual operations on it. Below is an example of access code to a MySQL database and the recovery of the tables and fields that make up each table.

Example of JDBC API use

```
...
import java.sql.*;
import java.util.logging.*;
...
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://
    SERVIDOR/BASE_DE_DATOS?useSSL=true",
         "USUARIO",
         "CLAVE");
DatabaseMetaData md = con.getMetaData();
String[] types = {"TABLE"};
ResultSet rs = md.getTables(null, null, "%", types);
while (rs.next()) {
     String nombreTabla = rs.getString("TABLE_NAME");
     System.out.println("Tabla: " + nombreTabla);
     ResultSet rs2 = md.getColumns(null, null, nombreTabla, null)
         ;
     while (rs2.next()) {
         String nombreCampo = rs2.getString("COLUMN_NAME");
         System.out.println("   Campo: " + nombreCampo);
     }
}
con.close();
...
```

For this practice the data of the connection are the following. The connection can only be done from a computer connected to the ULPGC network.

- Server: **mozart.dis.ulpgc.es**
- Database: **DIU_2018_19**
- User: **estudiante-DIU**
- Password: **DIU1819-aed56-noi**

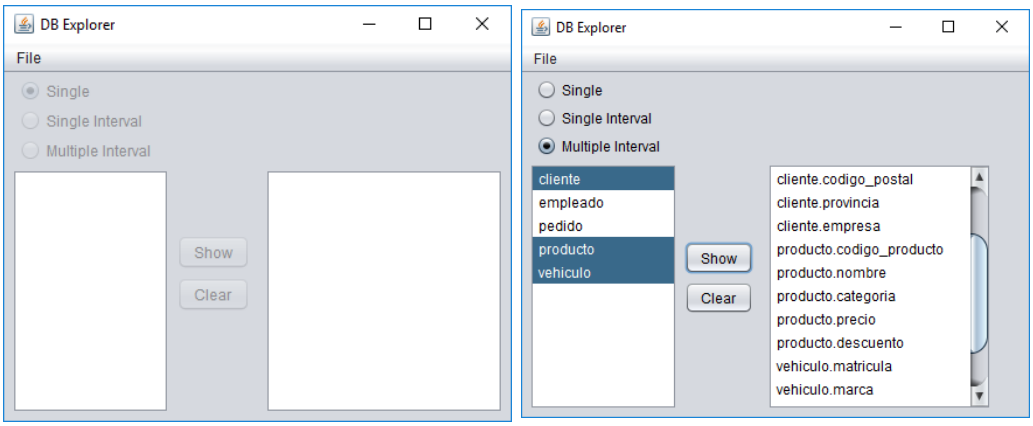
2. Implementation

We knew that this is going to be harder nut to crack. Since we wanted to program this application on our own laptops we have been forced to make some workaround. Using the local computer with CentOS and MySQL packages installed we created database dump:

```
1 mysqldump --single-transaction -u estudiante-DIU -h mozart.dis.
    ulpgc.es -p DIU_2018_19 > dump.sql
```

From the dump we created our local database to work with. Yet in final application we use default connecting into school database.

The application is mainly created with a MenuBar for options in the top area and lists on the left and right side of the application. There are also disabled settings for DB manipulation in form of radiobuttons and buttons between main areas (shown in pic 2.1a). MenuBar contains items with login and closing application. User needs to log in into school database using login form (pic 2.2) and credentials shown in the 1.1 Implementation notes section. After that two possibilites exist. Dialog forms differs if user logs in (pic 2.3a) or he fails to (pic 2.3b) and he has to try again. If he successfully connect into database the database tables will show in the left list and button settings will enable. User can afterwards choose the selection mode with the radio buttons and use that selection in the left list. If user wants to show the list of fields of the selected tables he has to click button "Show". Fields will show on the right list with name defined in the 1. Assigment chapter. Clicking on the second button "Clear" will clear fields from the right list and any selection made. The fully working application can be seen on picture 2.1b.



(a) Newly started application (b) Application in work

Figure 2.1: Database explorer application

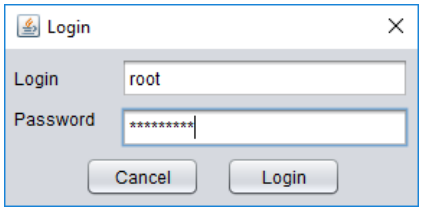
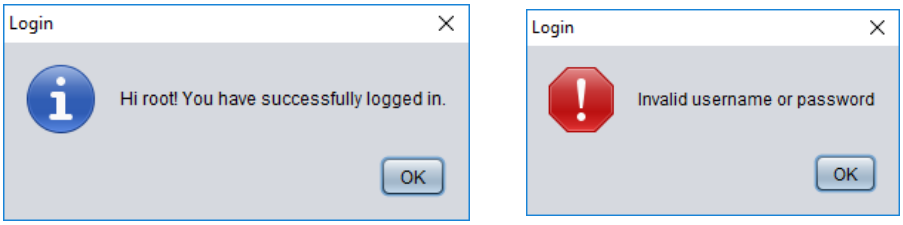


Figure 2.2: Login form



(a) Detail on login success (b) Detail on login failure

Figure 2.3: Different login states

3. Shneiderman and Plaisant principles

- **Consistency**
  Form is looking more unpleasant for user as there are no borders or highlighting. But the application is similiar to previous applications. Upper area is filled with settings and center/bottom area with output.
- **Universal usability**
  There are implemented shortcuts for connection and closing application (Menu-Items).
- **Informative feedback**
  When user connects to database he is informed about his success and table names immediately show in the list. Dialog boxes are shown when connecting to database.
- **Design dialogue to yield closure**
  After selecting tables and clicking buttons "Show" he immediately sees field result in the right list. Clear button works the same.
- **Simple error handling.**
  Login failure is shown by dialog box which tells the user about his wrong credentials.
- **Permit easy reversal of actions**
  Every shown field can be reversed to previous state by choosing the showing of right tables or clearing the right list.
- **Support internal locus of control**
  User is in control of connecting to given database. He can use any database user he has access to. He can show any fields from tables.
- **Reduce short-term memory load**
  For such a small application our application is fast enough for human needs.