

Grupparbete SQL

Backendutveckling och systemintegration.

Av Zein och Stefan Sundström

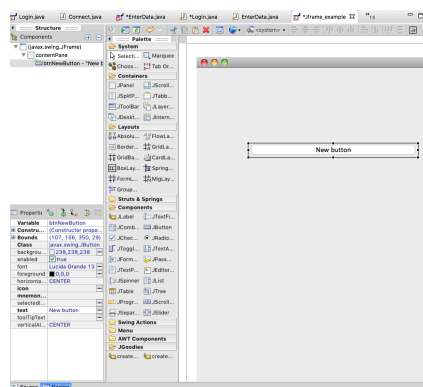
Beskrivning

Vi har valt att göra ett program som kan skriva och hämta information om hyresgäster och lägenheter från en databas. Användaren loggar in och kan sedan lista lediga lägenheter och lägga till nya hyresgäster. Vi använder oss av en server för uppkopplingen som måste startas innan programmet körs.

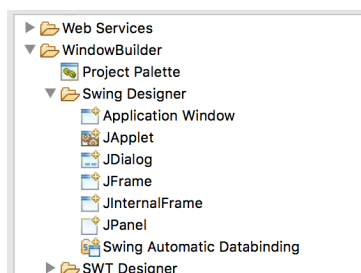
Bla bla bla..lite MVC tänk kanske

Gränssnitt

Gränssnittet är gjort i Swing med hjälp av WindowsBuilder - en GU-designer som kan köras i Eclipse. Den gör att man kan gå mellan kod och design-vy, man behöver på så sätt inte "chansa" när man sätter dimensioner och placering av objekten på skärmen.



När man installerat WindowsBuilder kan man, istället för att skapa en tom klass, utgå från en preset - JFrame i wizarden skapar en klass som ärver av JFrame. Själva huvudfönstret skapas och sedan kan man lägga till objekt som knappar och textfält.



"new/Other.." i Eclipse

Koden

Om man skriver manuellt så deklarerar man först det objekt man vill använda, sedan skapas en instans från klassen, dimensionerna sätts (x, y, bredd och höjd). Sedan körs metoden add från huvudfönstrets objekt för att det nya objektet ska läggas till. Vill man köra en händelse så lägger man till en ActionListener med Interfacet ActionListener som inparameter. En så kallad "anonym inre klass".

```
35 public class EnterData extends JFrame {
36
37     private JPanel mainframe;           lblPersonnummer = new JLabel("Personnummer");
38     private JTextField firstName;       lblPersonnummer.setBounds(45, 204, 99, 16);
39     private JTextField lastName;        mainframe.add(lblPersonnummer);
40     private JTextField personNumber;
41     private JTextField phoneNumber;     btnVisaLedigaLagenheter = new JButton("Visa Lediga Lägenheter");
42     private JTextField email;           btnVisaLedigaLagenheter.addActionListener(new ActionListener() {
43     private JTextField apartmentNumber; public void actionPerformed(ActionEvent e)
44     private JLabel lblPersonnummer;     {
45     private JLabel lblMobilnummer;      // Här skriver man vad som ska hända när man trycker på knappen
46     private JLabel lblEmail;           }
47     private JLabel lblLagenhetsnummer; });
48     private JButton btnSkrivUtgster;    btnVisaLedigaLagenheter.setBounds(762, 65, 184, 29);
49     private JTable table;              mainframe.add(btnVisaLedigaLagenheter);
50 }
```

JAR-filer

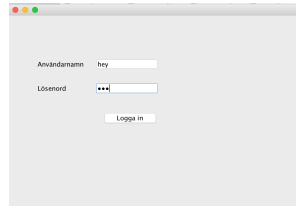
Vi använder oss av två jarfiler som importeras till Bulid Path'en i vår IDE.

RS2XML används för att på ett lätt sätt kunna skriva ut tabeller i Swing och mysql.connector.java är JDBC-drivern för MySQL-databasen vi jobbar mot.

Funktioner

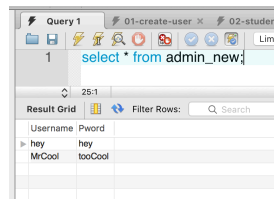
Logga in

För att logga in måste användaren ange ett användarnamn och ett lösenord.



A screenshot of a simple login window. It has a light gray background. At the top, there's a label 'Användarnamn' followed by a text input field containing the text 'hey'. Below that, there's a label 'Lösenord' followed by a password input field with three dots. At the bottom, there's a button labeled 'Logga in'.

Verifieringen görs genom att man jämför de två variabler användaren matar in mot motsvarande hos de administratörer som finns lagrade i databasens tabellen.



A screenshot of a database query result. The query is 'select * from admin_new;'. The result is displayed in a table with two columns: 'Username' and 'Pword'. There are two rows of data: one with 'hey' and 'hey', and another with 'MrCool' and 'tooCool'.

Username	Pword
hey	hey
MrCool	tooCool

I klassen Login är körs en "händelselyssnare" på knappen "Logga in", vilket gör att metoden körs när användaren trycker på knappen. En socket skapas som kopplas till en Server på den egna datorn (local host port 2277).

Ett objekt av klassen OutputStreamWriter och PrintWriter används för att skicka iväg användarnamnet och lösenordet till Servern. Lösenordet valideras och svaret läses av med hjälp av BufferedReader-objektet "reader" och sparas som strängen "state". Om state är "1", dvs lösenord och användarnamn överensstämmer med en tabellrad i databasen, så öppnas startar programmet (huvudfönstret "EnterData" öppnas).

```
JButton btnLoggaIn = new JButton("Logga in");
btnLoggaIn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            Socket socket = new Socket("localhost", 2277);
            OutputStreamWriter out = new OutputStreamWriter(socket.getOutputStream());
            PrintWriter writer = new PrintWriter(out, true);

            InputStreamReader input = new InputStreamReader(socket.getInputStream());
            BufferedReader reader = new BufferedReader(input) {

                writer.println(username.getText());
                writer.println(passwordField.getPassword());

                String state = reader.readLine();

                if (state.equals("1")) {
                    Login.this.dispose();
                    EnterData enterDataView = new EnterData();
                    enterDataView.setVisible(true);
                } else {
                    JOptionPane.showMessageDialog(null, "användarnamnet och/eller lösenordet är fel ");
                }
            }

        } catch (Exception e2) {
            System.out.println();
            JOptionPane.showMessageDialog(null, "Det gick inte att koppla");
        }
    }
});
btnLoggaIn.setBounds(189, 195, 117, 29);
contentPane.add(btnLoggaIn);
```

Valideringen i detalj

Server.java

Användarnamnet och lösenordet samlas in i While loopen där villkoret säger att loopen kommer att snurra så länge lösenordet inte är 1. loginAdmin-metoden anropas som ligger i klassen Connection, användarnamn, lösenord och uppkoppling skickas in i metoden.

```
public static void main(String[] args) {
    try {
        ServerSocket server = new ServerSocket(2277);
        System.out.println("server is running");
        Socket socket;
        int state = Integer.MIN_VALUE;
        Connect connect = new Connect();
        Connection con = Connect.getConnection();
        while (state != 1) {
            socket = server.accept();
            // Write to users
            OutputStreamWriter out = new OutputStreamWriter(socket.getOutputStream());
            PrintWriter writer = new PrintWriter(out, true);

            // Read information from users
            InputStreamReader input = new InputStreamReader(socket.getInputStream());
            BufferedReader reader = new BufferedReader(input);

            // Här Servern tar emot datan från Login klassen
            String username = reader.readLine();
            String password = reader.readLine();
            state = connect.loginAdmin(username, password, con);
            writer.println(Integer.toString(state));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Connect.java

I loginAdmin skickas en query som säger "välj alla rader från tabellen admin som överensstämmer med användarnamn och lösenord". Ett PreparedStatement-objekt skapas från uppkopplingen med query'n som inparameter, från objektet så talar vi om vad "?" har för värde på diverse position. Sedan skickas detta till databasen som talar om för oss hur många rader som överensstämmer med vår fråga och denna sparas som en int i ett ResultSet. Om lösenordet stämmer returneras 1 från metoden.

```
public int loginAdmin(String username, String password, Connection con) {
    String query = "select * from admin_new where Username=? and Pword=?";
    int state;
    try {
        PreparedStatement pst = con.prepareStatement(query);
        pst.setString(1, username);
        pst.setString(2, password);
        ResultSet rs = pst.executeQuery();

        if (rs.next()) {
            return state = 1;
        } else {
            return state = 0;
        }
    } catch (SQLException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
    return state = -1;
}
```

Server.java

Värdet som metoden ovan returnerar lagras sedan som ett heltal (state) i serverklassen, omvandlas till en sträng och skickas ut via port 2277.

```
// Här Servern tar emot datan från Login klassen
String username = reader.readLine();
String password = reader.readLine();
state = connect.loginAdmin(username, password, con);
writer.println(Integer.toString(state));
```

Login.java

Metoden i knappens händelselyssnar-scoop läser inputen på port 2277 och inkommande data sparas som strängen state. Är lösenordet rätt, dvs 1, så öppnas programmet. Annars så meddelas användaren att lösenordet är fel och får försöka igen.

```
String state = reader.readLine();
if (state.equals("1")) {
    Login.this.dispose();
    EnterData enterDataView = new EnterData();
    enterDataView.setVisible(true);
} else {
    JOptionPane.showMessageDialog(null, "användarnamnet och/eller lösenordet är fel ");
}
```

