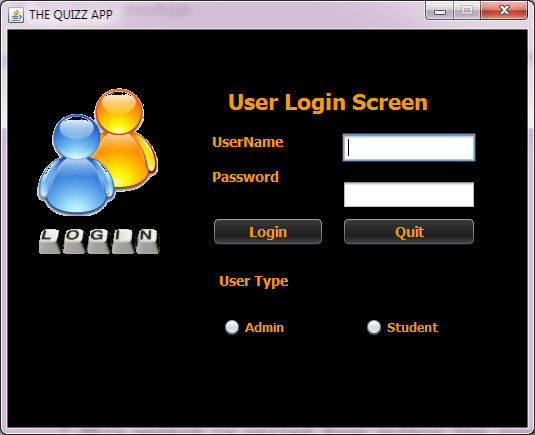
**The LoginFrame**



**THE TABLE USED IN LoginFrame**

**USERS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | | **Data Type** | **Description** |
| **USERID** | **Varchar2(10)** | | **Contains Userid** |
| **PASSWORD** | **Varchar2(10)** | | **Contains password** |
| **USERTYPE** | **Varchar2(10)** | | **Contains ‘Student’ or ‘Admin’ to indicate type of the user** |

**THE POJO CLASSES USED IN LoginFrame**

1. The **UserPojo** POJO

2. The **UserProfile** POJO

**CODE FOR "UserPojo" POJO**

***package onlineexam.pojo;***

***public class UserPojo {***

***private String userId;***

***private String password;***

***private String userType;***

***public UserPojo()***

***{***

***}***

***public UserPojo(String userId, String password, String userType) {***

***this.userId = userId;***

***this.password = password;***

***this.userType = userType;***

***}***

***public String getUserId() {***

***return userId;***

***}***

***public void setUserId(String userId) {***

***this.userId = userId;***

***}***

***public String getPassword() {***

***return password;***

***}***

***public void setPassword(String password) {***

***this.password = password;***

***}***

***public String getUserType() {***

***return userType;***

***}***

***public void setUserType(String userType) {***

***this.userType = userType;***

***}***

***}***

**CODE FOR "UserProfile" POJO**

***package onlineexam.pojo;***

***public class UserProfile {***

***private static String username;***

***private static String usertype;***

***public static String getUsertype() {***

***return usertype;***

***}***

***public static void setUsername(String username) {***

***UserProfile.username = username;***

***}***

***public static void setUsertype(String usertype) {***

***UserProfile.usertype = usertype;***

***}***

***public static String getUSername(){***

***return username;***

***}***

***}***

**THE DAO CLASSES USED IN LoginFrame**

1. The **UserDAO**

**HOW TO VERIFY USERID/PASSWORD**

To do this we need to create a method in **UserDAO** called **validateUser()** which will accept a **UserPojo** object as argument,search the **USER** TABLE for the given **UserID, Password** and **Type**  and return **true** or **false** , based upon whether the user is found or not.

Following is the prototype of this method:

**public static String validateUser(UserPojo user)throws SQLException**

Following are it's steps:

**a. It will accept an UserPojo POJO object as argument containing all the fields of data**

**b. It will get a Connection object from DBConnection class using the method getConnection( )**

**c. It will then frame a SELECT query with placeholders for the given USERID** ,**PASSWORD and USERTYPE**

**d. It will then create a PreparedStatement object passing it the SELECT query and use setters of the PreparedStatement to replace question marks with actual values of UserPojo object.**

**e. Then it will execute the query by calling the method executeQuery( ) of PreparedStatement and receive the result in a ResultSet object**

**f. Now , it will check whether USERNAME is found or not . If it is found then the method will return the true otherwise it will return false.**

**g. It will not handle any SQLException and will simply pass it on to it's caller**

2. Following is it's code:

***public static boolean validateUser(UserPojo user)throws SQLException***

***{***

***Connection conn=DBConnection.getConnection();***

***String qry="Select \* from Users where userid=? and password=? and usertype=?";***

***PreparedStatement ps=conn.prepareStatement(qry);***

***ps.setString(1,user.getUserId());***

***ps.setString(2,user.getPassword());***

***ps.setString(3, user.getUserType());***

***ResultSet rs=ps.executeQuery();***

***return rs.next();***

***}***

**WRITING THE CODE FOR BUTTON Button.png IN LoginFrame**

1. When the user will click the **Login** button then it will verify the login details and accordingly open the next screen

2. Following are it's important points:

**a. It will first validate whether all the data has been properly filled all or not.**

**b. If not , then it will generate an error message and return.**

**c. Then it will verify whether "Admin" or "Student" option has been selected or not.**

**d. If not , then it will generate an error message and return.**

**e. Otherwise , it will create an UserPojo object , fill all the values in it and pass it to the method validateUser( ) of the UserDAO.**

**f. If the method validateUser( ) returned false then it will display an error message.**

**g. Otherwise , it will also store username and usertype as static fields in another class called UserProfile. This class will be used throughout the app to display the username on every frame.**

**h.Then if the user is Admin it will open the AdminOptionsFrame and if the user is Student , it will open the StudentOptionsFrame**

**i. It will also handle any SQLException that will be thrown by the method validateUser( )**

**public class LoginFrame extends javax.swing.JFrame {**

**private String username;**

**private String password;**

**public LoginFrame() {**

**initComponents();**

**this.setLocationRelativeTo(null);**

**}**

**private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {**

**boolean isValidInput=validateInputs();**

**if(isValidInput==false)**

**{**

**JOptionPane.showMessageDialog(null, "Username or Password cannot be left blank!","Error!",JOptionPane.ERROR\_MESSAGE);**

**return;**

**}**

**String userType=getUserType();**

**if(userType==null)**

**{**

**JOptionPane.showMessageDialog(null, "Please Choose User Type","Error!",JOptionPane.ERROR\_MESSAGE);**

**return;**

**}**

**try**

**{**

**UserPojo user = new UserPojo();**

**user.setUserId(username);**

**user.setPassword(password);**

**user.setUserType(userType);**

**boolean isValidUser=UserDAO.validateUser(user);**

**if(isValidUser)**

**{**

**JOptionPane.showMessageDialog(null, "Login Accepted!","Welcome "+username,JOptionPane.INFORMATION\_MESSAGE);**

**UserProfile.setUsername(username);**

**UserProfile.setUsertype(userType);**

**if(user.getUserType().equalsIgnoreCase("admin"))**

**{**

**AdminOptionsFrame adminFrame=new AdminOptionsFrame();**

**adminFrame.setVisible(true);**

**}**

**else**

**{**

**StudentOptionsFrame studentFrame=new StudentOptionsFrame();**

**studentFrame.setVisible(true);**

**}**

**this.dispose();**

**}**

**else**

**JOptionPane.showMessageDialog(null, "Invalid UserId/Password","Login Denied!",JOptionPane.ERROR\_MESSAGE);**

**}**

**catch(SQLException sq)**

**{**

**JOptionPane.showMessageDialog(null, "Error while connecting to DB!","Exception!",JOptionPane.ERROR\_MESSAGE);**

**sq.printStackTrace();**

**}**

**}**

***private boolean validateInputs() {***

***username=txtUsername.getText();***

***char []pwd=txtPassword.getPassword();***

***if(username.isEmpty()|| pwd.length==0 )***

***return false;***

***else***

***{***

***password=String.valueOf(pwd);***

***return true;***

***}***

***}***

***private String getUserType() {***

***if(jrAdmin.isSelected())***

***return jrAdmin.getText();***

***else if(jrStudent.isSelected())***

***return jrStudent.getText();***

***else***

***return null;***

***}***

***private void btnQuitActionPerformed(java.awt.event.ActionEvent evt) {***

***int ans;***

***ans=JOptionPane.showConfirmDialog(null, "Are u sure ?","Quitting!",JOptionPane.YES\_NO\_OPTION,JOptionPane.QUESTION\_MESSAGE);***

***if(ans==JOptionPane.YES\_OPTION)***

***{***

***DBConnection.closeConnection();***

***System.exit(0);***

***}***

***}***

**}**