**Designing The RegisterStudentFrame**

****

**STEPS TO BE DONE IN** **RegisterStudentFrame**

In the **RegisterStudentFrame** we need to do following steps:

1. Display **username** on the top left

2. Allow the user to **logout**

3. Write code for the "**Register**" Button . When this button is clicked it should:

**a. Validate the inputs. If validation fails it should print the Error Message and return**

**b. Create a new object of User POJO and initialize it with *username*,*password* and *usertype*  information.**

**c. Add the User to the database by calling the method**

**addUser( ) of UserDAO**

**d. If record is successfully added it should display the message "Registration Successfully Done" and if a user by that name is already present it should display the message "User already present"**

**e. It should handle all the exceptions also**

4. Write code for the "**Back**" Button . When this button is clicked it should:

a**. Dispose the current frame and open the AdminOptionsFrame**

**THE TABLES USED IN RegisterStudentFrame**

**1.USERS**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Description** |
| **USERID** | **Varchar2(10)** | **Contains Userid of the admin or student** |
| **PASSWORD** | **Varchar2(10)** | **Contains password** |
| **USERTYPE** | **Varchar2(10)** | **Stores "Student" or "Admin"** |

**THE POJO CLASSES USED IN RegisterStudentFrame**

1. The **User** POJO

**THE DAO CLASSES USED IN RegisterStudentFrame**

1. The **UserDAO**

**HOW TO ADD STUDENT TO THE DATABASE:**

To register a student in the database we need to insert his/her record in the USER table.

To do this we will create a method called **addUser( )** in the **UserDAO**. The prototype of the method is:

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

This method will do 2 things:

**1. Check whether the userid given is present or not in the USERS table. If it is present the method addUser() should return false.**

**2. If userid is not present in the table then the method addUser() should add a new record of the given userid in the database and return true.**

Following is the code:

***public static boolean addUser(UserPojo user)throws SQLException{***

***String qry="Select \* from users where userid=?";***

***boolean status=true;***

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

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

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

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

***if(rs.next())***

***status=false;***

***else***

***{***

***qry="Insert into users values(?,?,?)";***

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

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

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

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

***ps.executeUpdate();***

***}***

***return status;***

***}***

**DESIGNING THE validateInputs( ) METHOD:**

This method should do the following:

**1. If any of the inputs is empty , it should return 0;**

**2. if userid or password is of less than 4 characters it should return -1**

**3. If password and retype password don't match**

**it should return -2**

**4.If everything is correct it should return 1**

Based on above points the prototype of **validateInputs( )** is:

***private int validateInputs()***

Following is the code:

***private int validateInputs() {***

***userid=txtUserid.getText();***

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

***char []repwd=txtRetypePassword.getPassword();***

***if(userid.isEmpty()|| pwd.length==0 || repwd.length==0 )***

***return 0;***

***else***

***{***

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

***String repassword=String.valueOf(repwd);***

***if(userid.length()<4 || pwd.length<4)***

***return -2;***

***if(password.equals(repassword)==false)***

***return -1;***

***else***

***return 1;***

***}***

***}***

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

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

***int response=validateInputs();***

***if(response==0)***

***{***

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

***return;***

***}***

***if(response==-1)***

***{***

***JOptionPane.showMessageDialog(null, "Password and Retype Password do not match!","Error!",JOptionPane.ERROR\_MESSAGE);***

***return;***

***}***

***if(response==-2)***

***{***

***JOptionPane.showMessageDialog(null, "Userid and Password should be atleast 4 characters long!","Error!",JOptionPane.ERROR\_MESSAGE);***

***return;***

***}***

***try***

***{***

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

***user.setUserId(userid);***

***user.setPassword(password);***

***user.setUserType("Student");***

***boolean status=UserDAO.addUser(user);***

***if(status)***

***{***

***JOptionPane.showMessageDialog(null, "Registration Done!","Userid: "+userid+"\nPassword:"+password,JOptionPane.INFORMATION\_MESSAGE);***

***clearAll();***

***}***

***else***

***JOptionPane.showMessageDialog(null, "Userid Already Present","Registration Denied!",JOptionPane.ERROR\_MESSAGE);***

***}***

***catch(SQLException ex)***

***{***

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

***ex.printStackTrace();***

***}***

***}***

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

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

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

***adminFrame.setVisible(true);***

***this.dispose();***

***}***