

Re: Closing one window and opening another

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2006-08/msg02674.html>

- *From:* Ralf Seitner <Ralf.Seitner@xxxxxx>
 - *Date:* Wed, 30 Aug 2006 22:55:22 +0200
-

zilvar@xxxxxxxxx schrieb:

Using Netbeans 5.0, I'm trying to teach myself some Java more complicated than `System.out.println("Hello World.");`

I'd like some help figuring out the best way to build a login-style screen. It seems that I could create a main class and the two forms. My main could call `LoginScreen`, which listens for a button press, hides itself on command ... and this is where I get iffy. I doubt action listeners can return a value, so I think I need to implement a personal 'yo, successful login' event and wrap that around login in a try/catch block, which would catch the event and ask the 2nd window to show itself.

Is that a reasonable implementation of the idea, and if not, some pointers to whatever I'm missing would be much appreciated. :)

Thanks

Hi!

I don't know what you want to do with the try/catch-block.

The first thing which comes to my mind:

I think you can use a method, which performs the login, and returns a boolean (means: returns whether the login was successful)

You have a button, where an `ActionListener` listens on `ActionEvents`.

So... when an action is performed check if login was successful and if so, close the first window/dialog/frame and open the next one.

But maybe I totally misunderstood you...

This could look like the following ...

```
public boolean login(String userName, String passWord) {
    boolean loginSuccessful = false;
    // do something here to check if login is successful and
    // do some operations which have to be done if login
    // was successful – but only do them, if login is successful.
    // if successful set loginSuccessful to true.
```

Re: Closing one window and opening another

```
return loginSuccessful;
}
```

and then in your ActionListener... (you can realize your implementation of the ActionListener as an inner class, so you can access on methods defined in that class...)

```
public void actionPerformed(ActionEvent e) {
    boolean loginSuccessful;
    if (loginSuccessful) {
        loginScreen.setVisible(false);
        // probably even: loginScreen.dispose();
        JDialog dialogWhichFollowsOnSuccessfulLogin =
        createDialogWhichFollowsOnSuccessfulLogin()
        dialogWhichComesAfterLogin.setVisible(true);
    } else {
        // write a message and reset the password-field.
    }
}
private void createDialogWhichFollowsOnSuccessfulLogin {
    // create the dialog...
}
hth, bye, Ralf
.
```