Login Manager Windows Form Sample

Contents

Introduction

This is Part B1 of Login Manager document. The whole document consists of following files:

- Part A – It is a reference to the Login Management Framework. It can be downloaded from http://www.limnor.com/support/LoginManagerPartA.PDF
- Part B1 – This file. It contains the first part of a Windows Form Application sample.
- Part B2 – It contains the second part of a Windows Form Application sample. It can be downloaded from http://www.limnor.com/support/LoginManagerPartB2.PDF
- Part C – It contains a PHP web application sample. It can be downloaded from http://www.limnor.com/support/LoginManagerPartC.PDF
- Part D – It contains an ASPX web application sample. It can be downloaded from http://www.limnor.com/support/LoginManagerPartD.PDF

The files for sample projects are long but with little text. Most of contents are screenshots showing step by step programming procedures of sample projects. Most steps look similar and repetitive. For beginners, please pay attention to the highlighting in the screenshots, especially the highlighting in the Expression Editor: the result of each operation depends on which parts of an expression are highlighted.

Login Management Framework

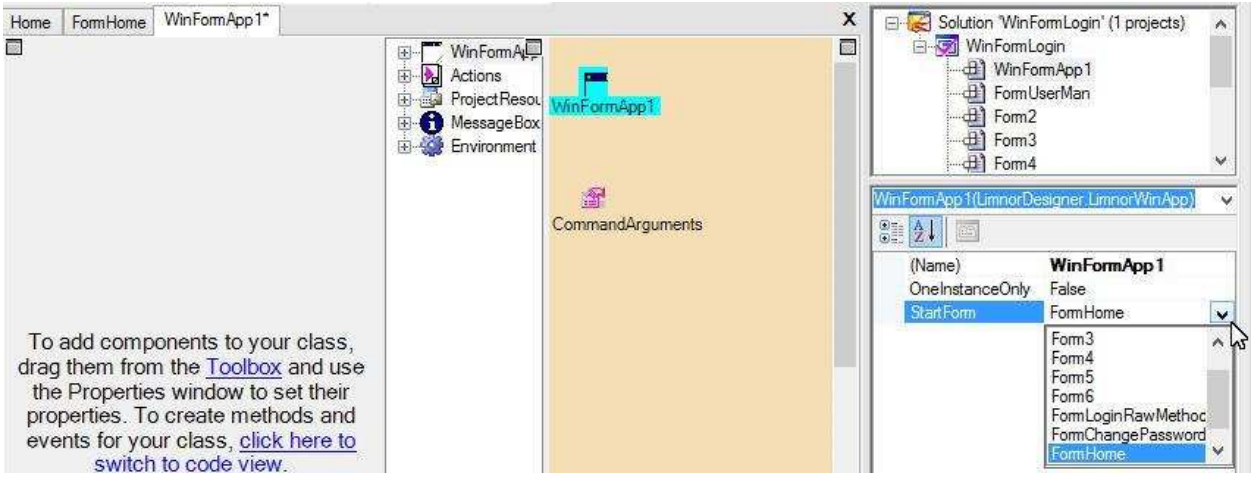
See part A from http://www.limnor.com/support/LoginManagerPartA.PDF

Windows Form Application Sample

We use a sample Windows Form project to show the use of the login management framework. Later we will use web samples.

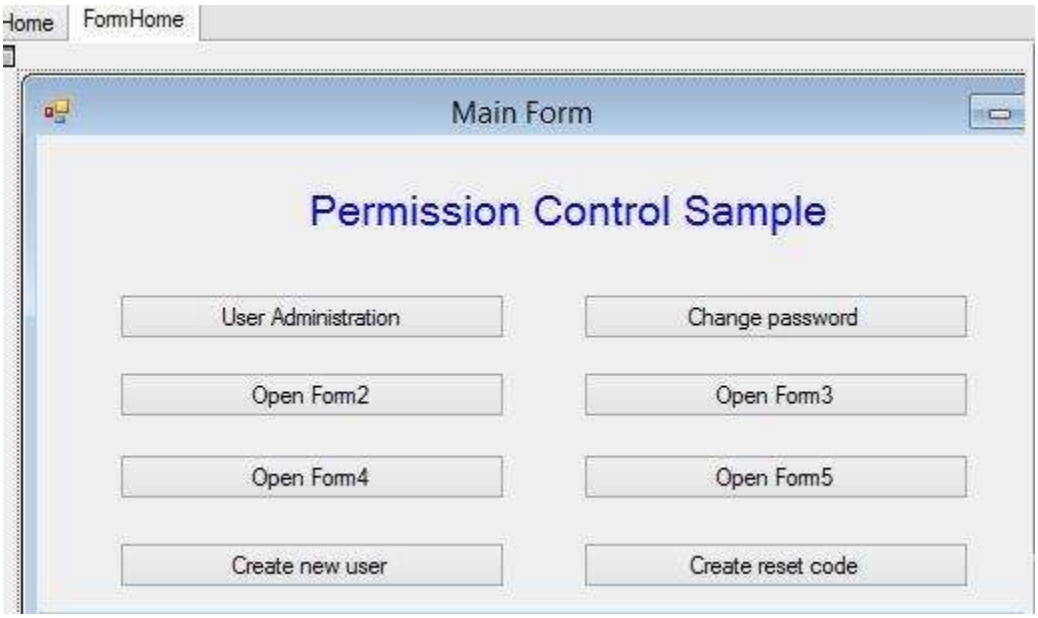
Start Form

The first form displayed in a Windows Form application is called the Start Form. The Application object has a StartForm property for specifying which form in a project to be used as the Start Form.

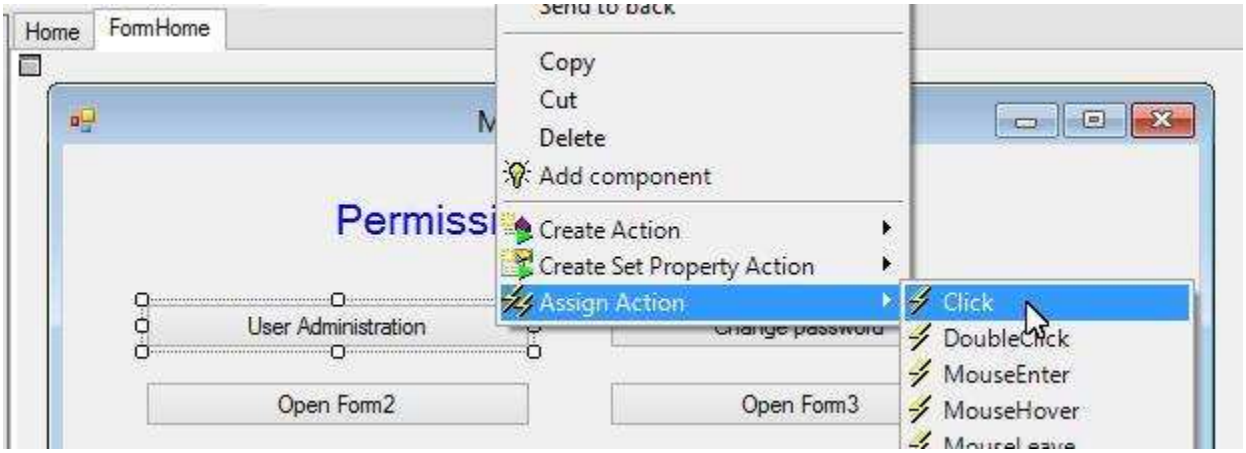


When the Start Form is closed the application shuts down. Thus, usually you should use a form which is always visible as the Start Form. For example, a MDI parent form is usually good for being the Start Form. Since a login form usually will be closed after log in, a login form usually is not good to be used as the Start Form.

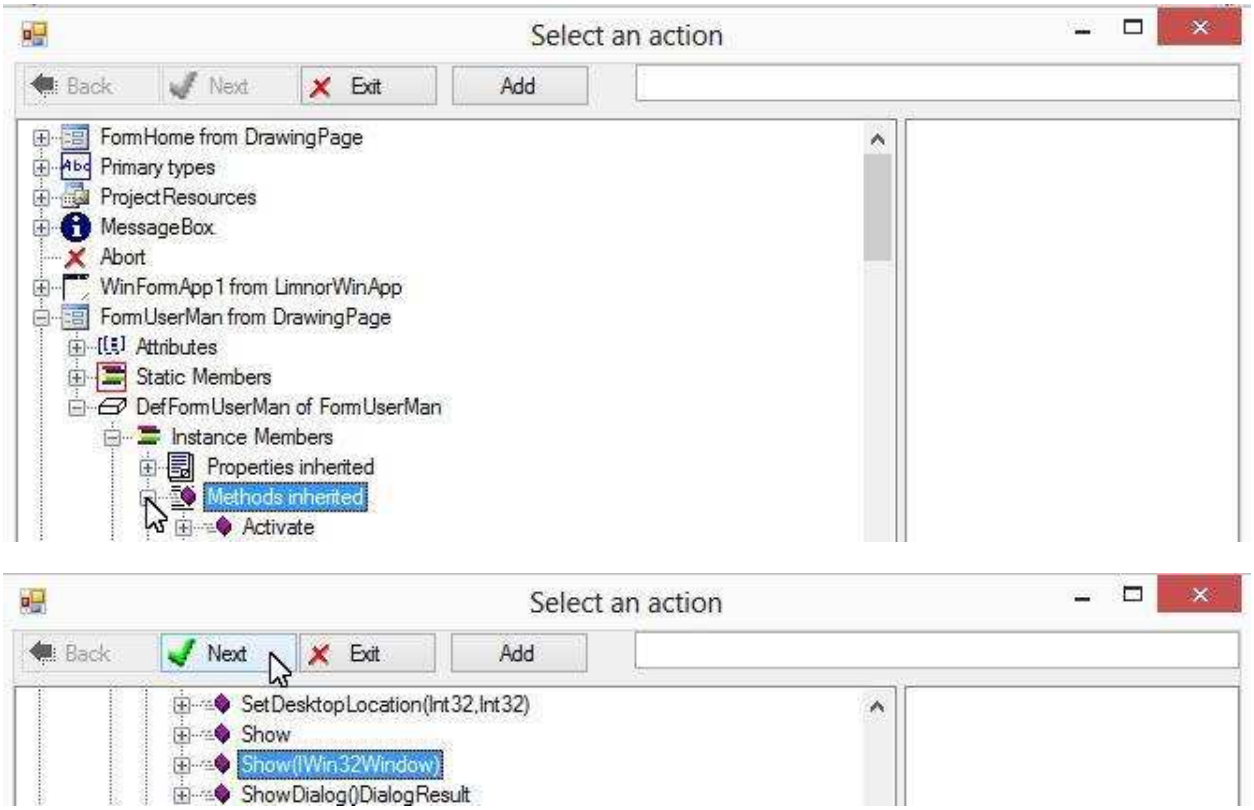
In this sample, we use such a Start Form as shown below:



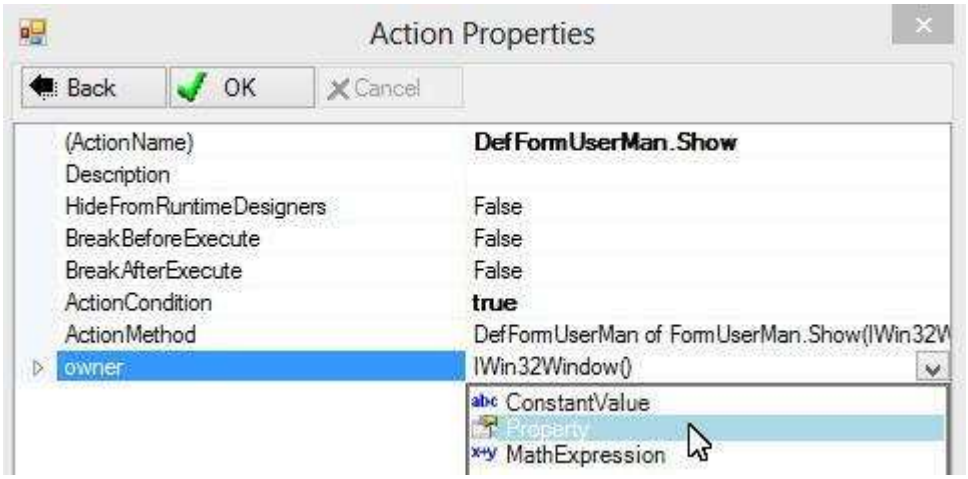
In this sample form, we have several buttons for opening other forms. The programming of these buttons is not affected by permission controls. Her we show the programming of the “User Administration” button. Right-click the button; choose “Assign Action”; choose “Click” event:



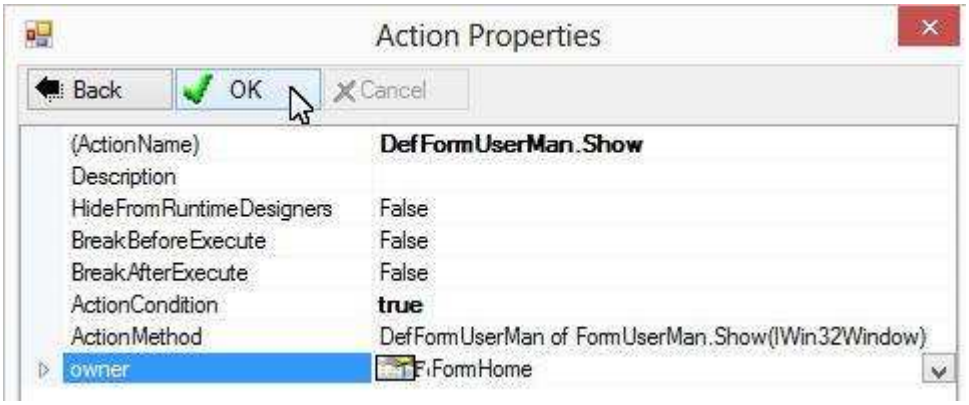
Select Show method of the default instance of the form FormUserMan:



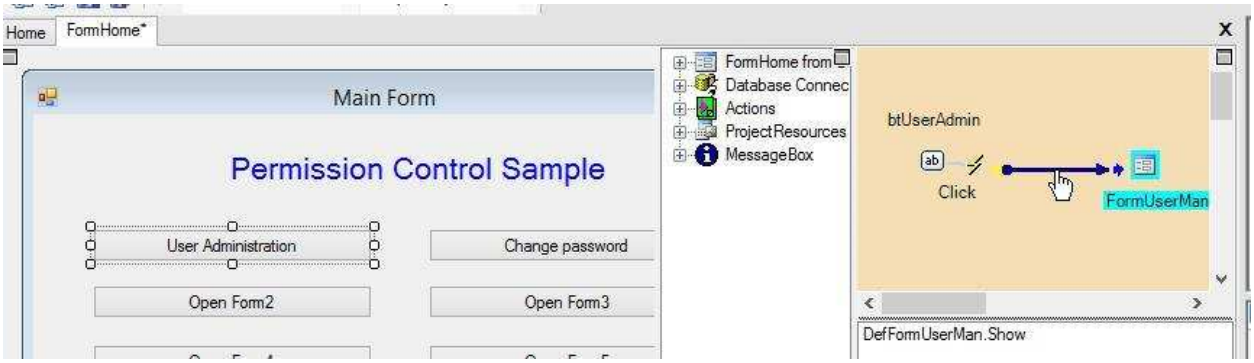
Select the current form as the owner for showing the user administration form:



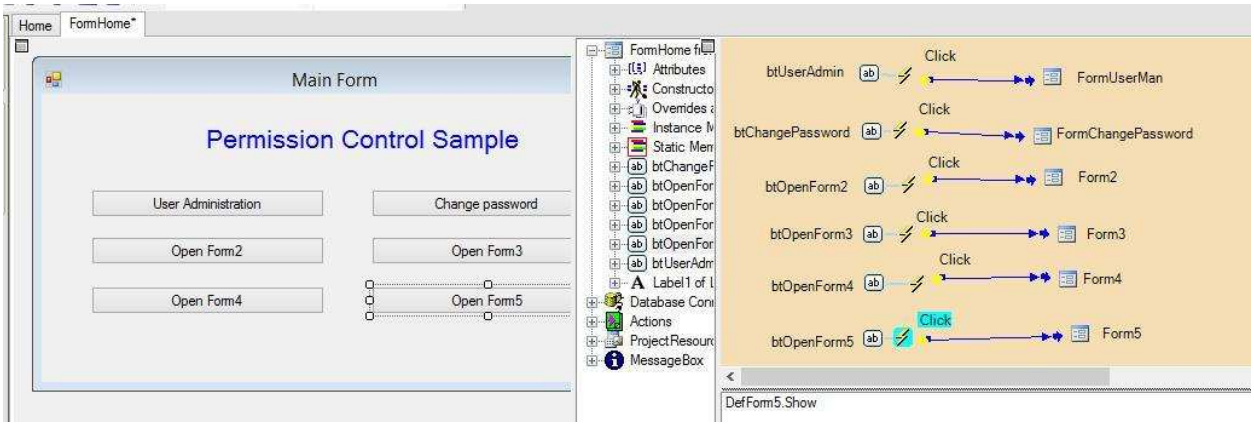
Click OK to finish creating this action:



The action is created and assigned to the button:



In the similar way, we may program all the buttons, each button opens one form:



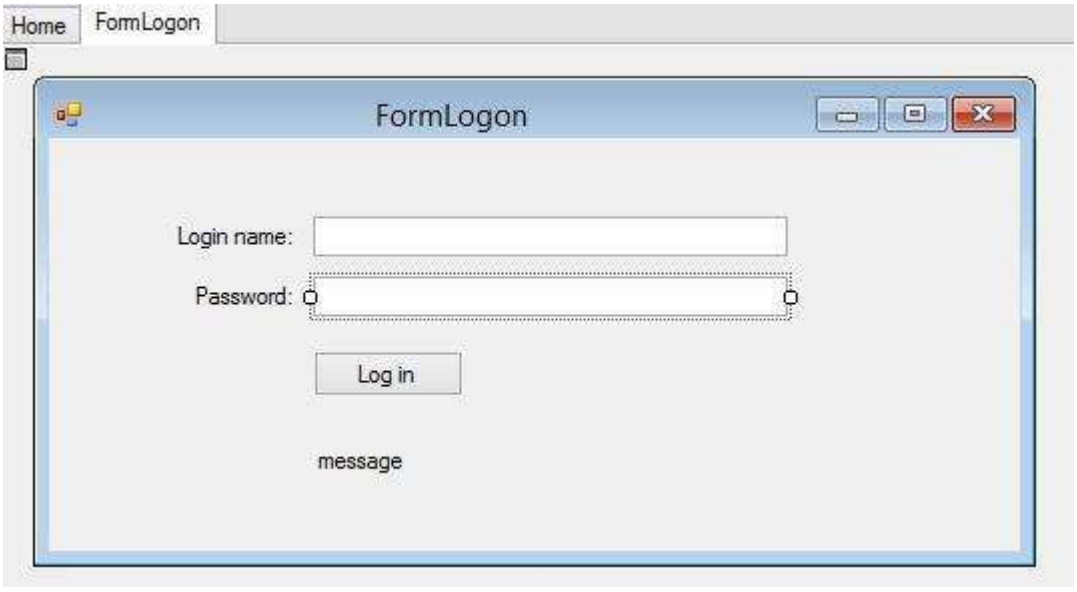
The programming in this section has nothing to do with login management and permission controls. It is normal Windows Form programming. For more information on Window Form programming, see http://www.limnor.com/support/UseDefaultFormInstance.pdf.

Login Form

To add permission controls to an application, we must develop a form to allow user log in.

UI

It uses a text box for user login name and a text box for password. It uses a button for submitting the login name and password. It uses a label to show error message.



User table

A database table is used to record all users. The user table should have at least a login name field and a password field. For this sample project, we use a table named useraccount. It has following fields:

Field name	Field type	Description
UserID	Integer	an auto-number

UserAlias	text	the login name
UserPassword	text	password hash
Salt	text	Salt for password hash
UserLevel	integer	permission level
ResetCode	text	Password reset code hash
ResetCodeExpire	Date time	Password reset code expiration time
FirstName	text	These fields are not used in login management
LastName	text	
Email	text	
Phone	text	

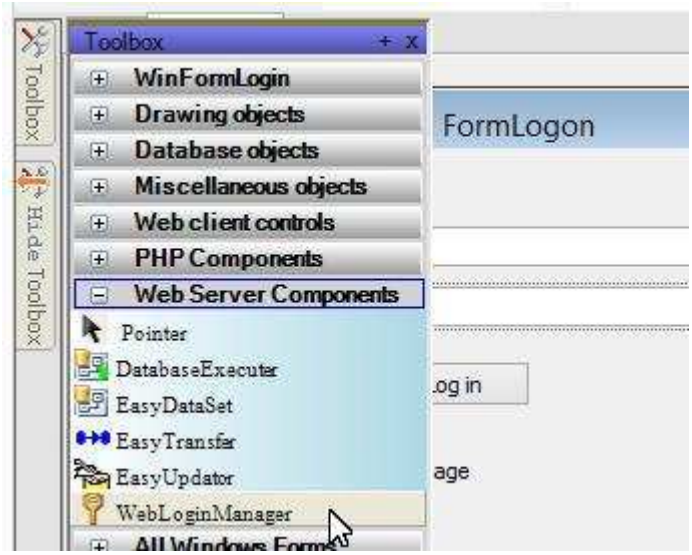
The SQL script for creating the above table is as following:

```
CREATE TABLE useraccount (
	UserID INT NOT NULL AUTO_INCREMENT,
	UserAlias VARCHAR(20) NOT NULL,
	UserPassword VARCHAR(128) NULL,
	Salt VARCHAR(128) NULL,
	UserLevel INT NULL,
	ResetCode VARCHAR(128) NULL,
	ResetCodeExpire DATETIME NULL,
	FirstName VARCHAR(30) NULL,
	LastName VARCHAR(30) NULL,
	Email VARCHAR(200) NULL,
	Phone VARCHAR(20) NULL,
	PRIMARY KEY (UserID)
);
CREATE UNIQUE INDEX UserAlias ON useraccount (UserAlias);
```

- Note the field sizes for UserPassword, Salt and ResetCode. These fields should be large enough for the hash algorithm to be used. For this sample, we are going to use SHA256.
- UserAlias and UserPassword fields are enough for the Login Management Framework to work.
- Salt field is recommended for more secure password storage.
- UserLevel field is for more permission controls.
- ResetCode and ResetCodeExpire fields are required for password reset functionality.
- UserID field is not used in login and permission control. The Login Management Framework may make the value of this field available for the convenience of your other programming.
- Other fields are not used by the Login Management Framework.

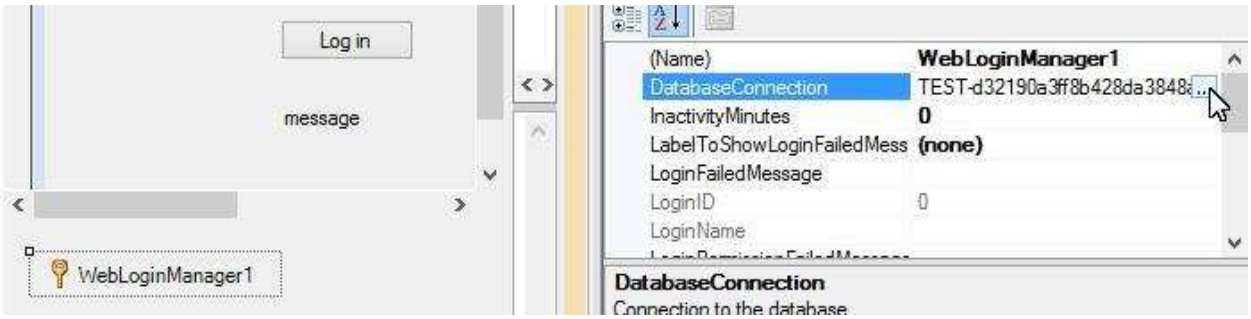
Add Login Manager Component

Drop WebLoginManager to the form.

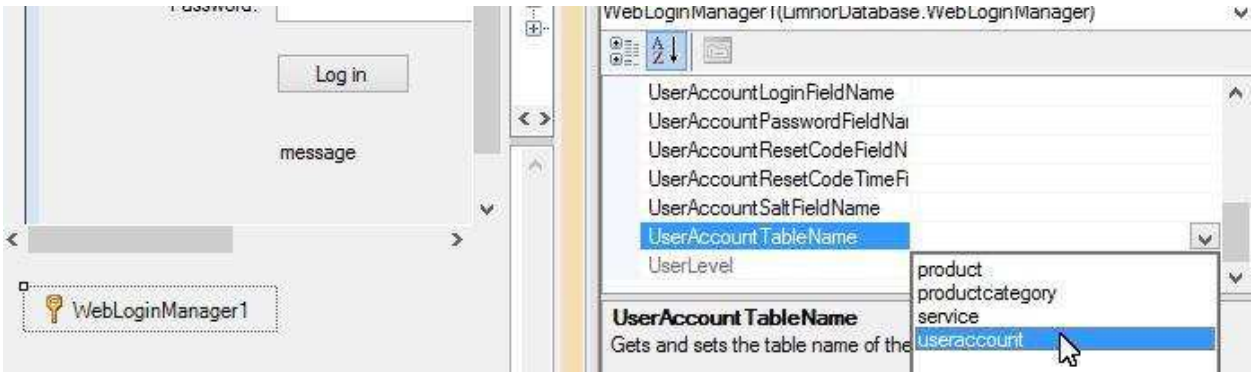


Set properties of Login Manager

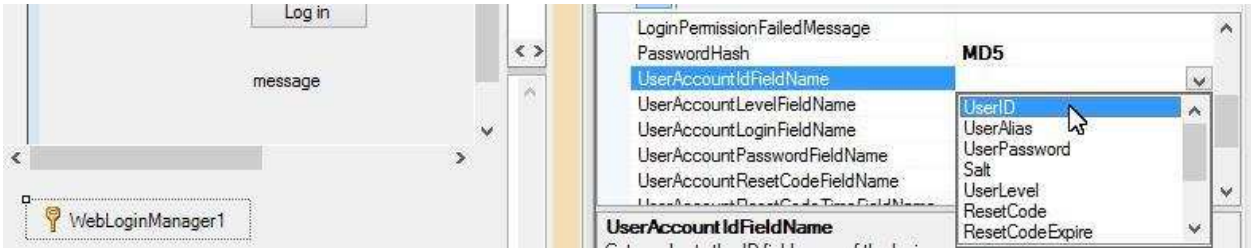
Set database connection:



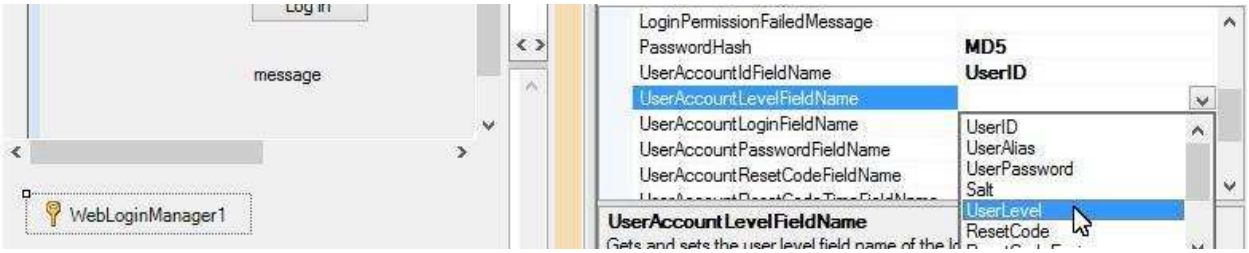
Specify user table:



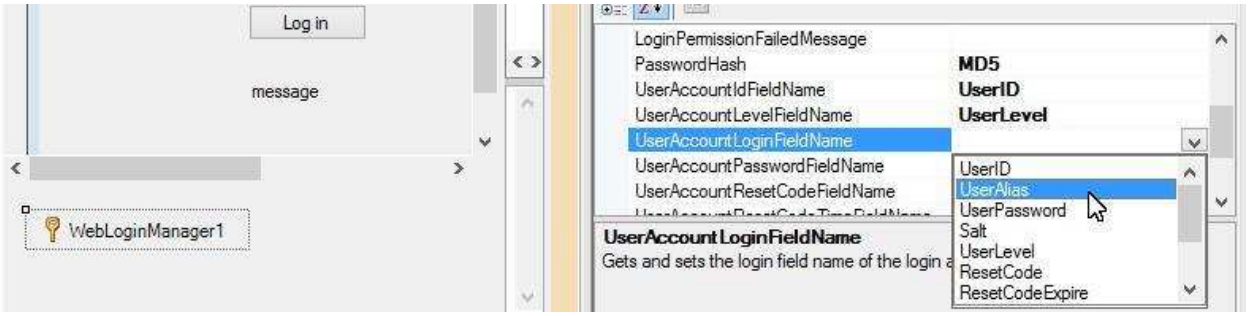
Specify user ID field (this is optional):



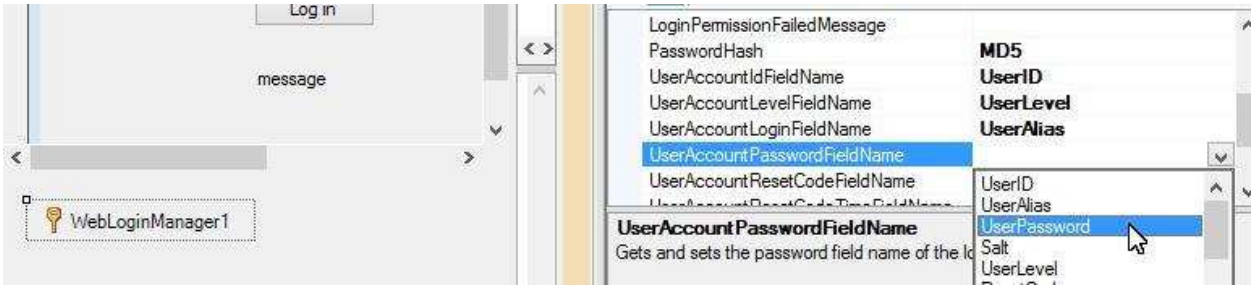
Specify user level field (this is optional):



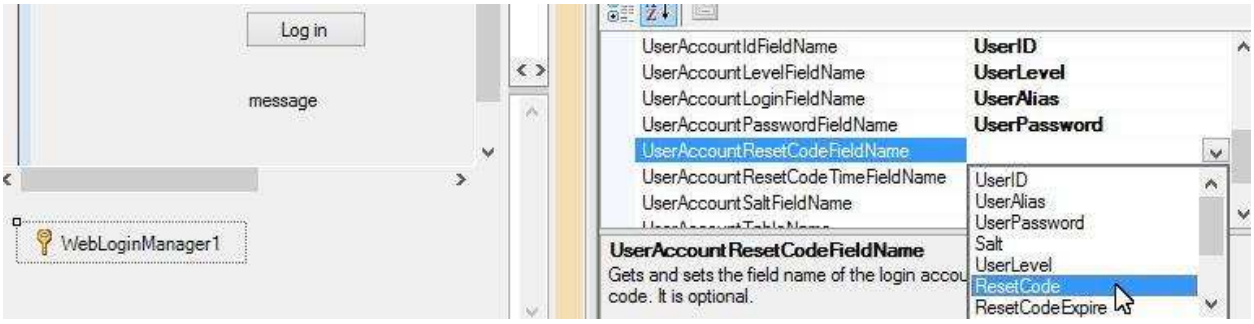
Specify user login name field (this is required):



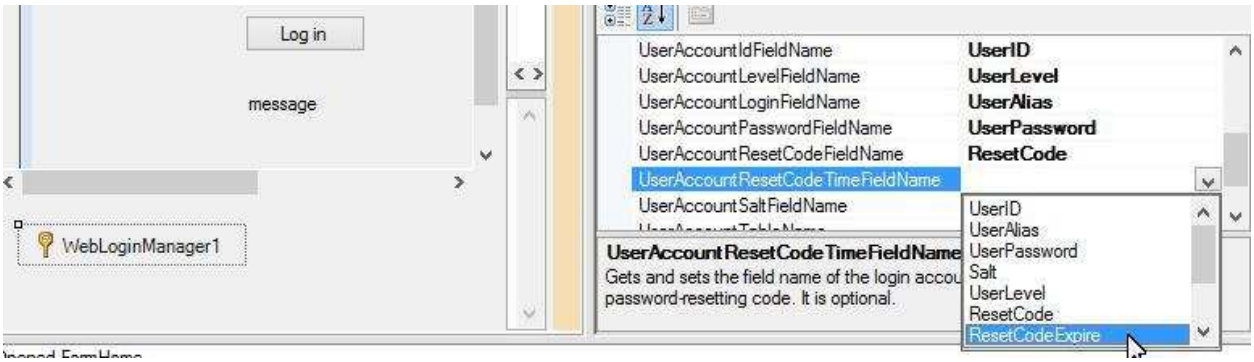
Specify user password field (this is required):



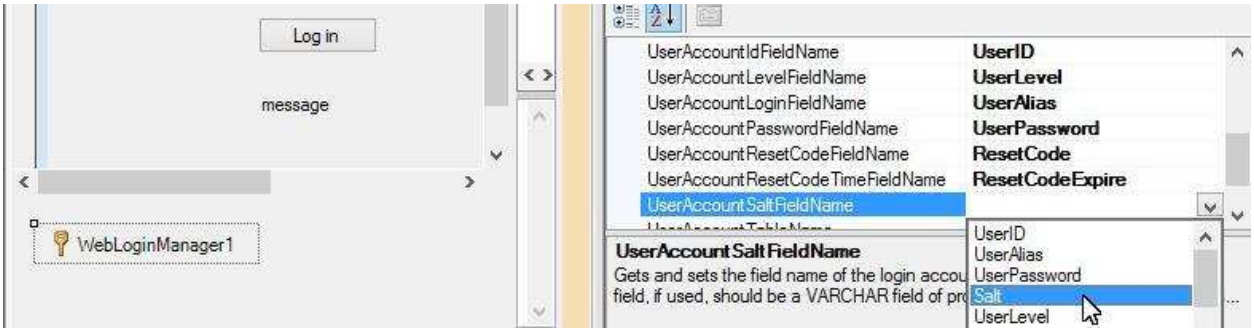
Specify reset code field (this is optional):



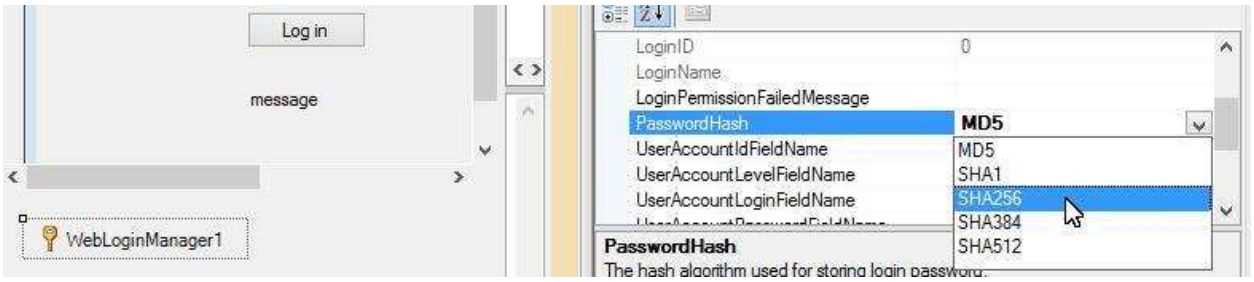
Specify reset code expiration field (this is optional):



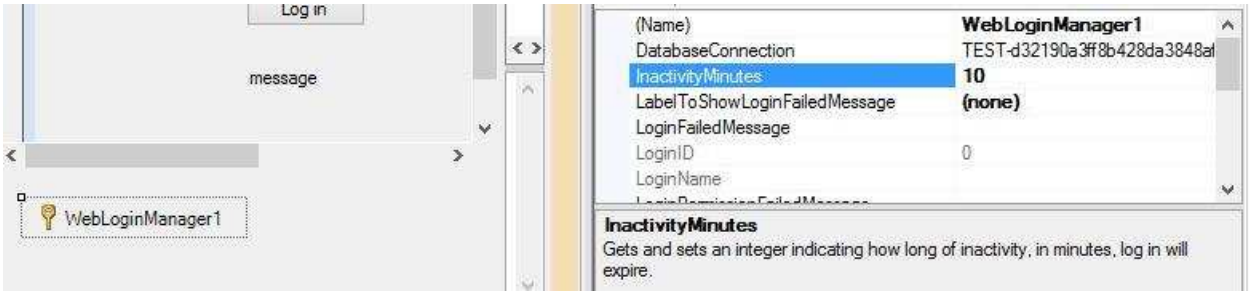
Specify salt field (this is optional):



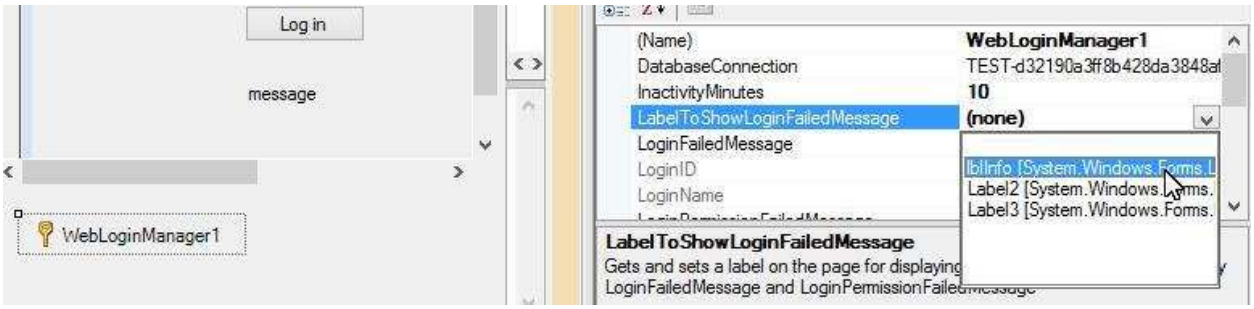
Specify password hash algorithm. For this sample, we use SHA256:



Specify login expiration time. We use 10 minutes. If there is not a mouse or keyboard activity in 10 minutes then the login will expire and all protected forms will be closed.

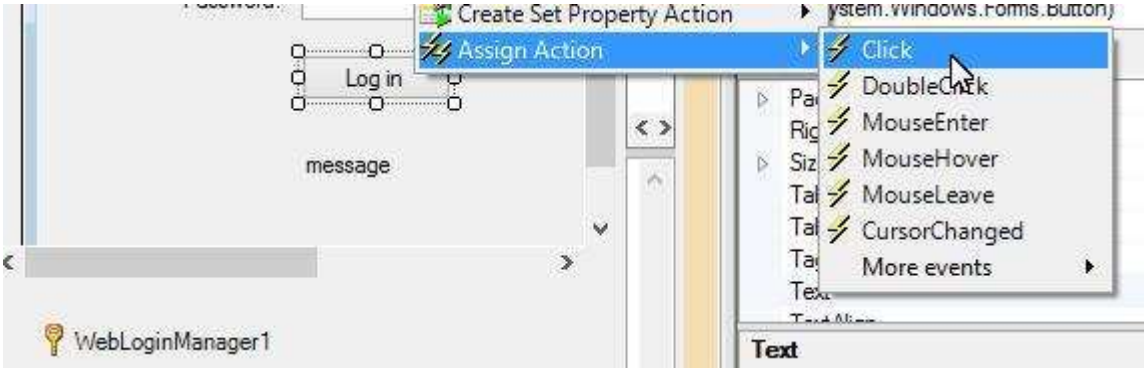


Specify error message display label:

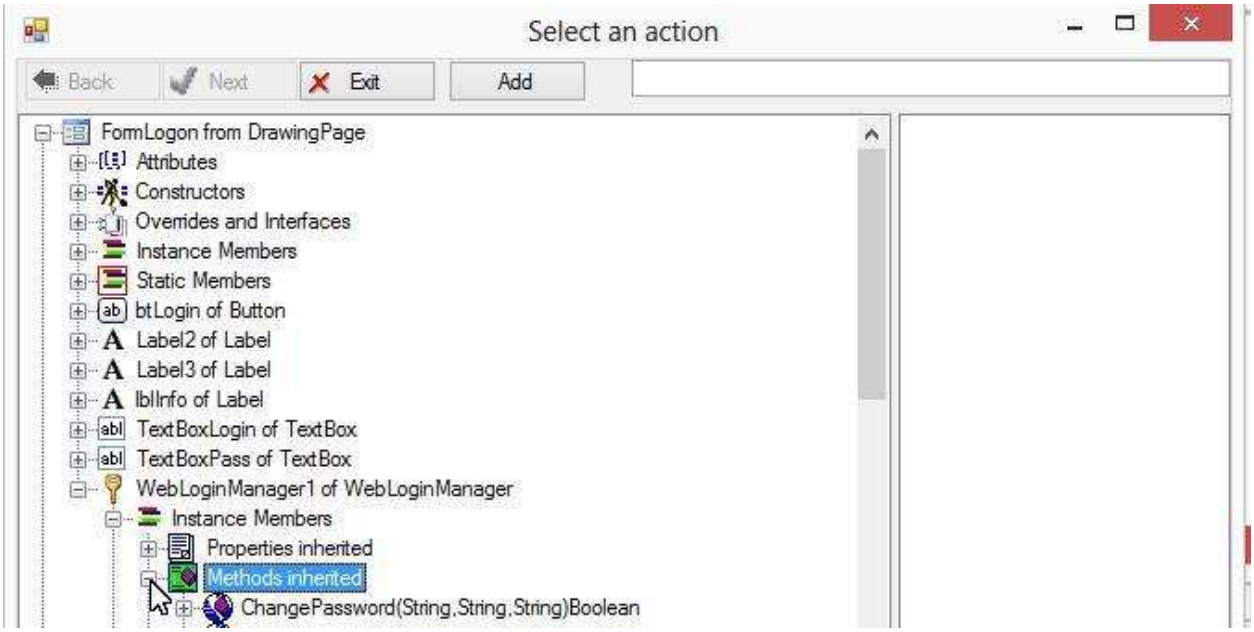


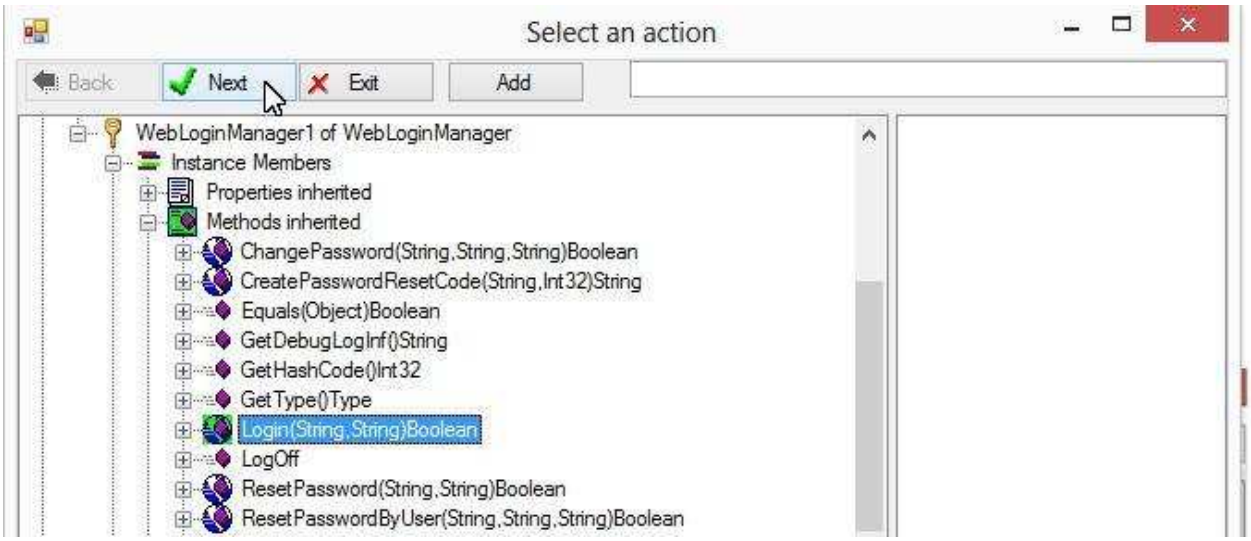
Execute login action

The button is used to execute a login action. Right-click the button; choose “Assign Action”; choose “Click” event:

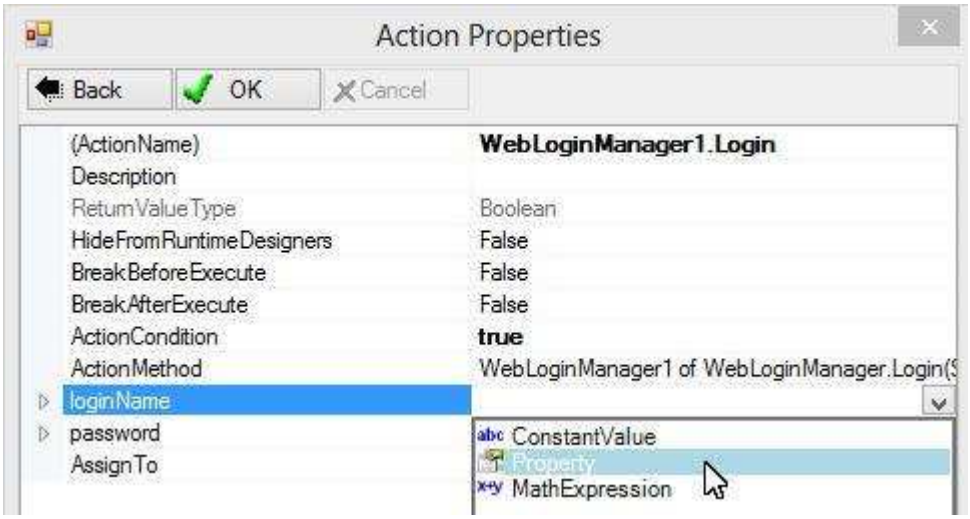


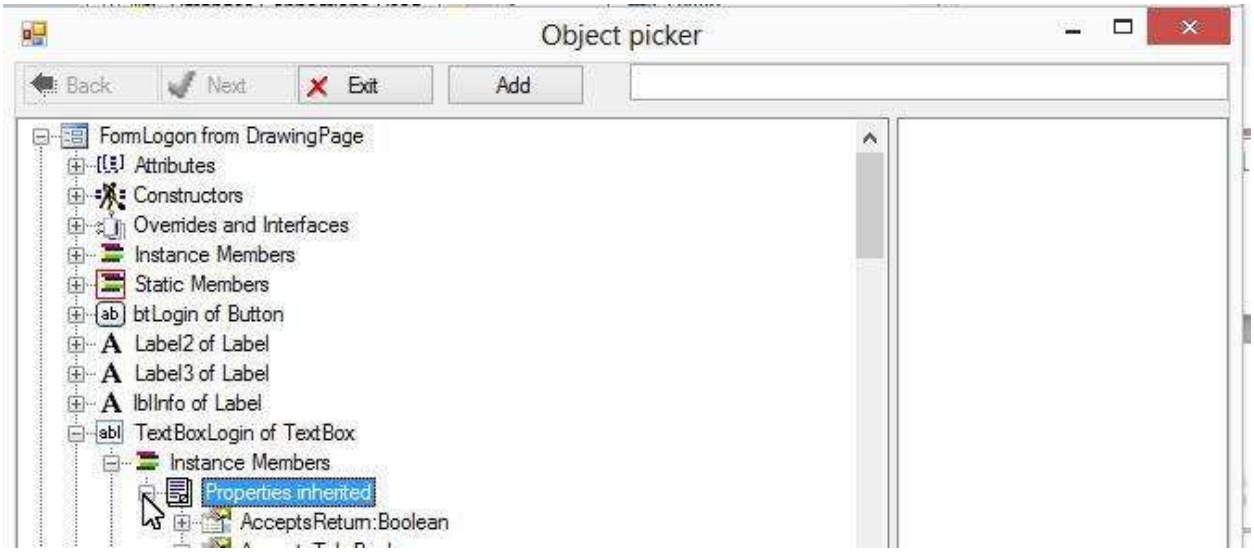
Choose the Login method of the Login Manager:



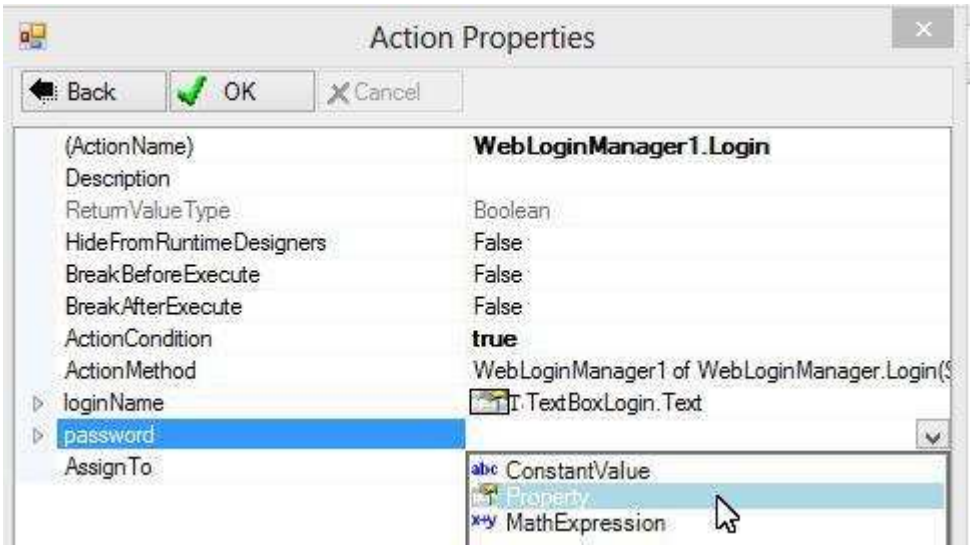


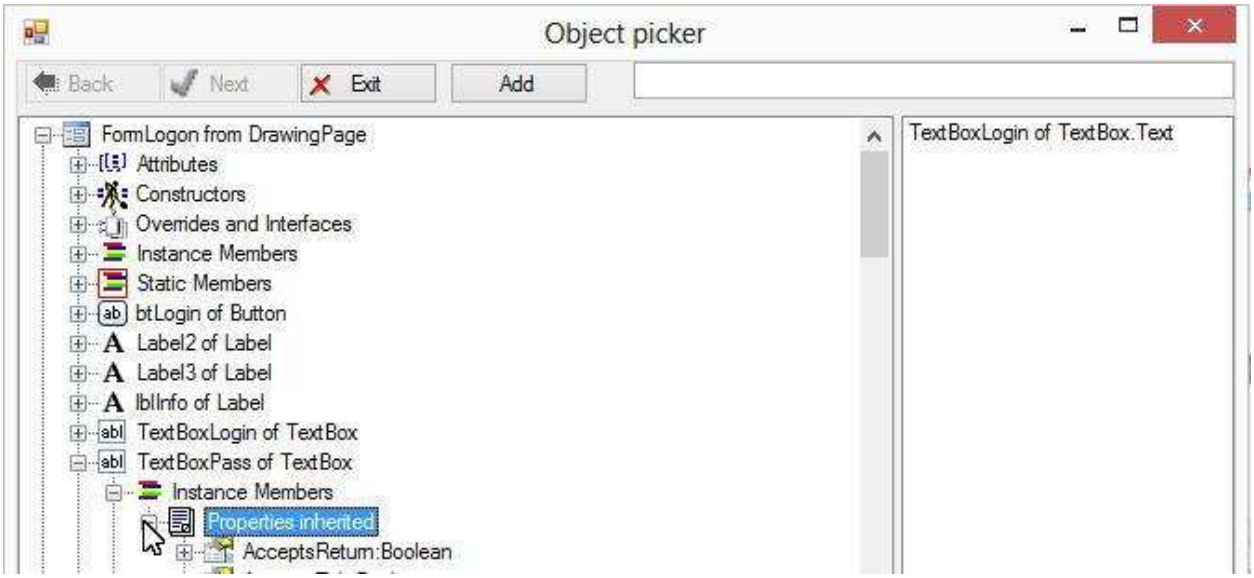
Select the Text property of login name text box for the loginName:



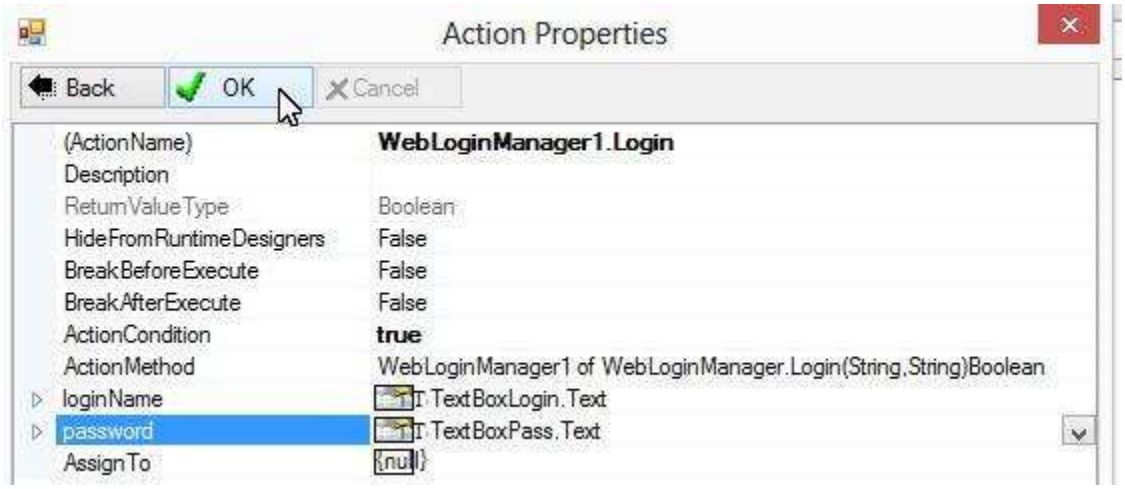


Select the Text property of the password text box for “password” parameter:

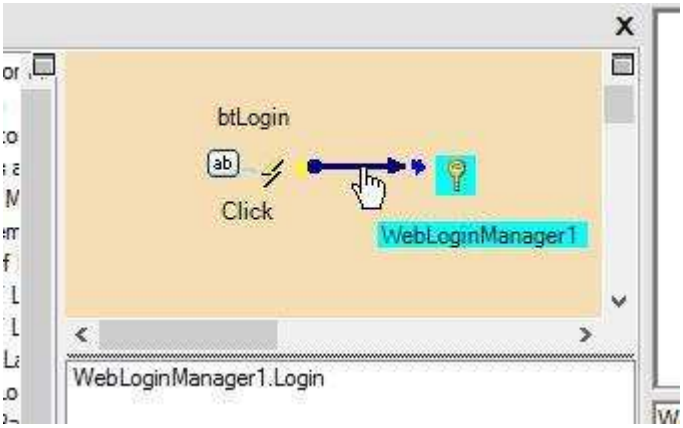




Click OK to finish creating this action:



The action is created and assigned to the button:

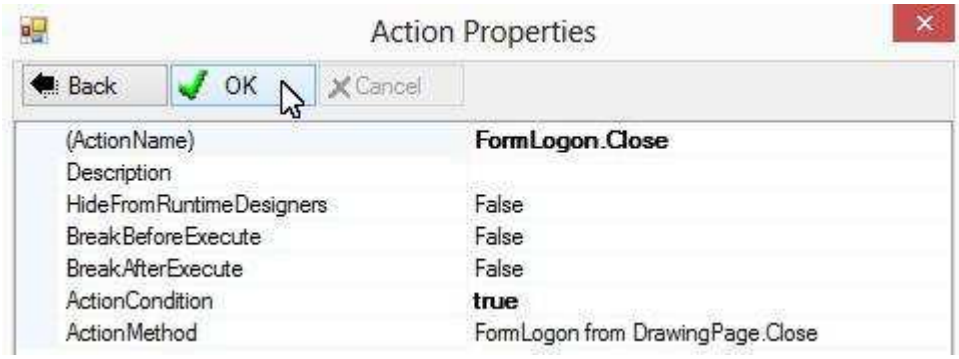


Close login form

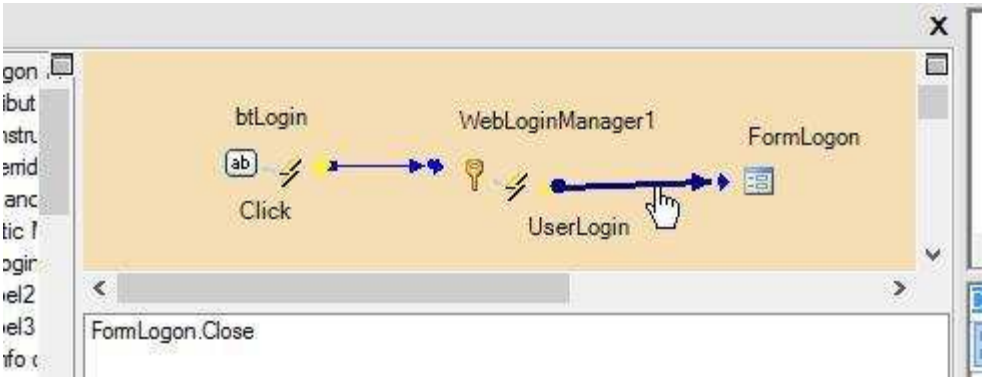
In this sample project, once the log on is successful, we want to close the login form. We may do it at the time the event UserLogin occurred. Right-click the Login Manager component; choose “Assign Action”; choose “UserLogin” event:

Choose Close method of the current form:

Click OK to finish creating this action:



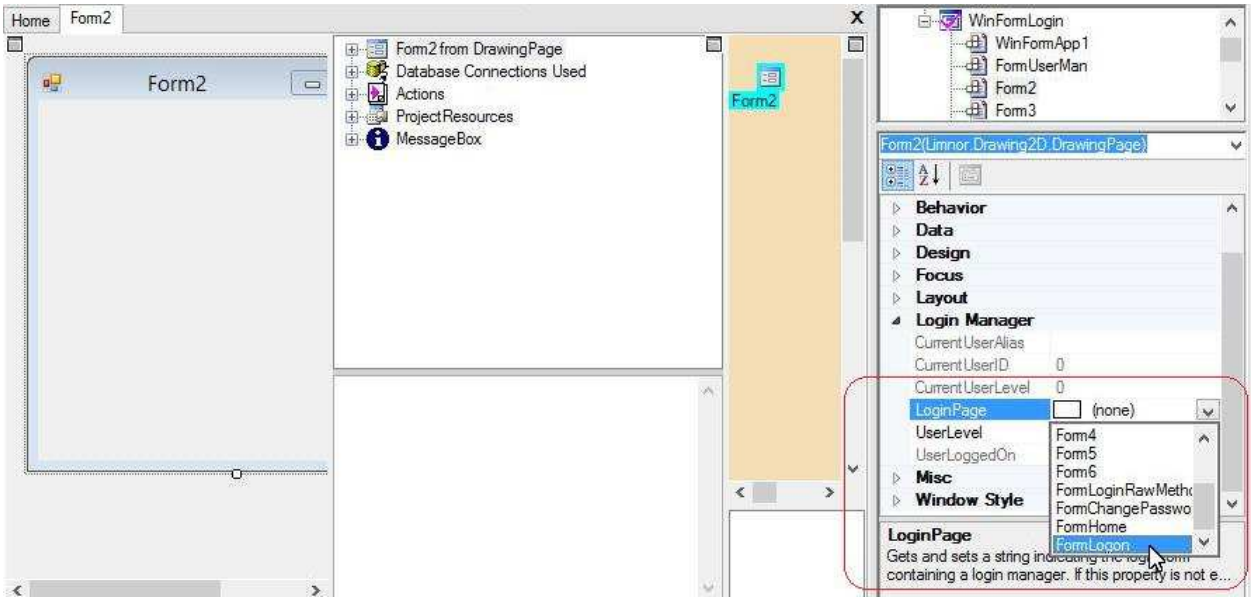
The action is created and assigned to the event:



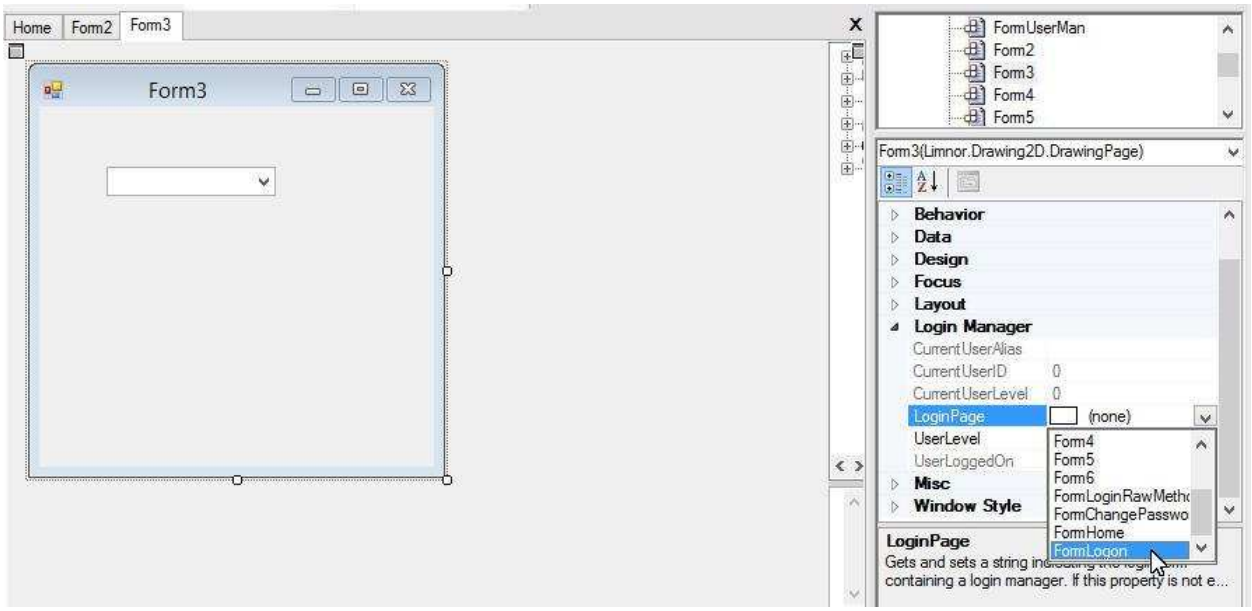
Note that this step of handling UserLogin event to close the login form is only needed for a Windows Form application. For a web application, this handling is not needed. For a web application, after logging on, target web page is automatically loaded.

Protect Forms

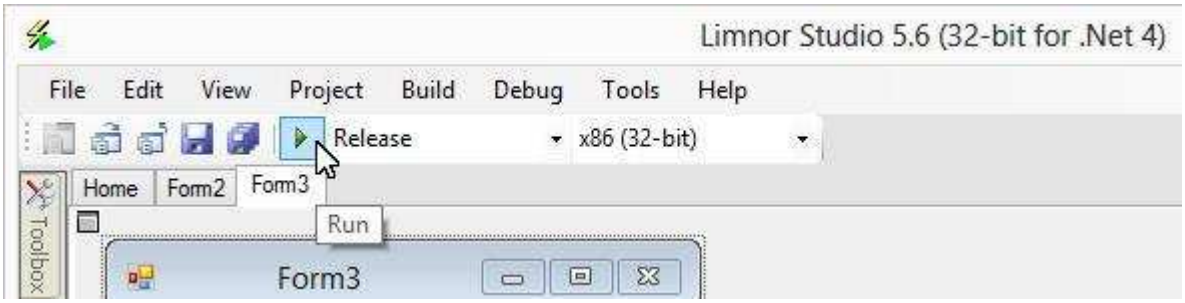
Once we have a login form, we may protect forms. For a form we want to protect, set its LoginPage property to the login form:



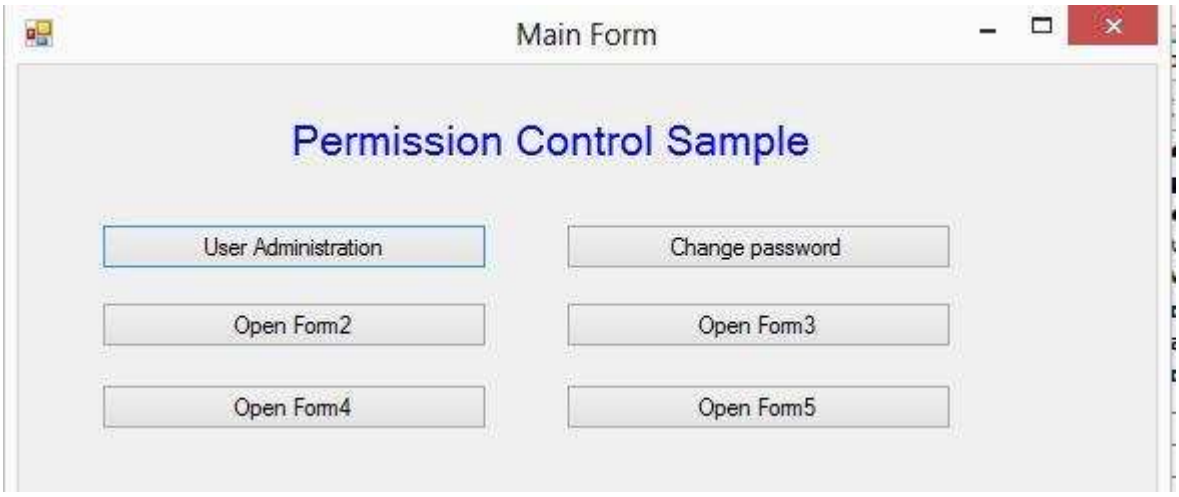
We may do it for all forms to be protected:



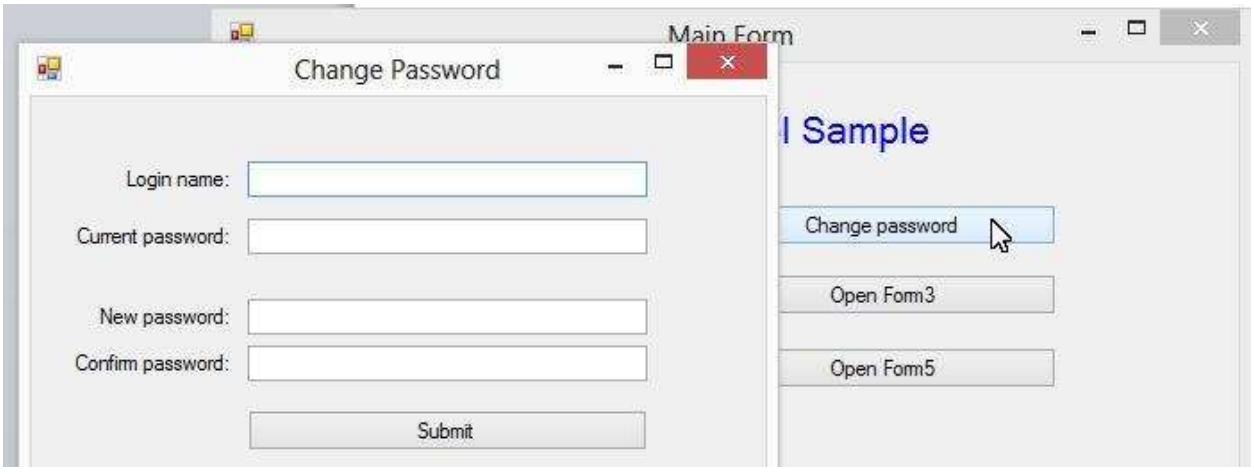
We are done adding permission control to the application. Let's see its effects. Run the project:



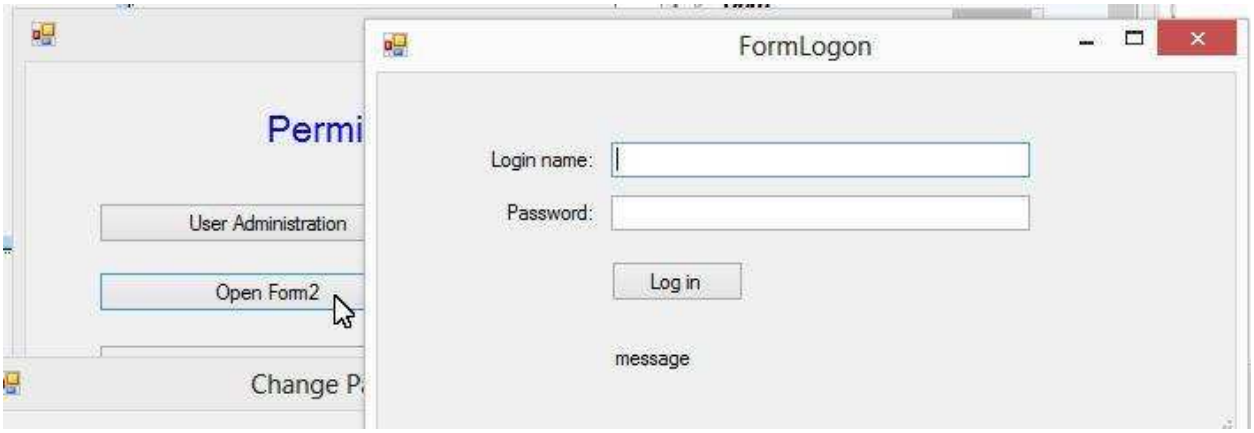
The Start Form appears:



Click “Change password”, the form for changing password appears because this form is not protected:



Click “Open Form2”, the login form appears as a dialogue because Form 2 is protected:



Close the login form without logging on, an error message appears:

We do not see Form 2 appears, because it is closed. Note that showing login dialogue and closing the protected form are done automatically by the Login Management Framework, saving you the trouble of programming these activities.

Note that we may also protect the Start Form. If we do that then the login form will appear at the time the application is started. If a log in is not done then the application will shut down, because closing the Start Form will shut down the application.

Password Management

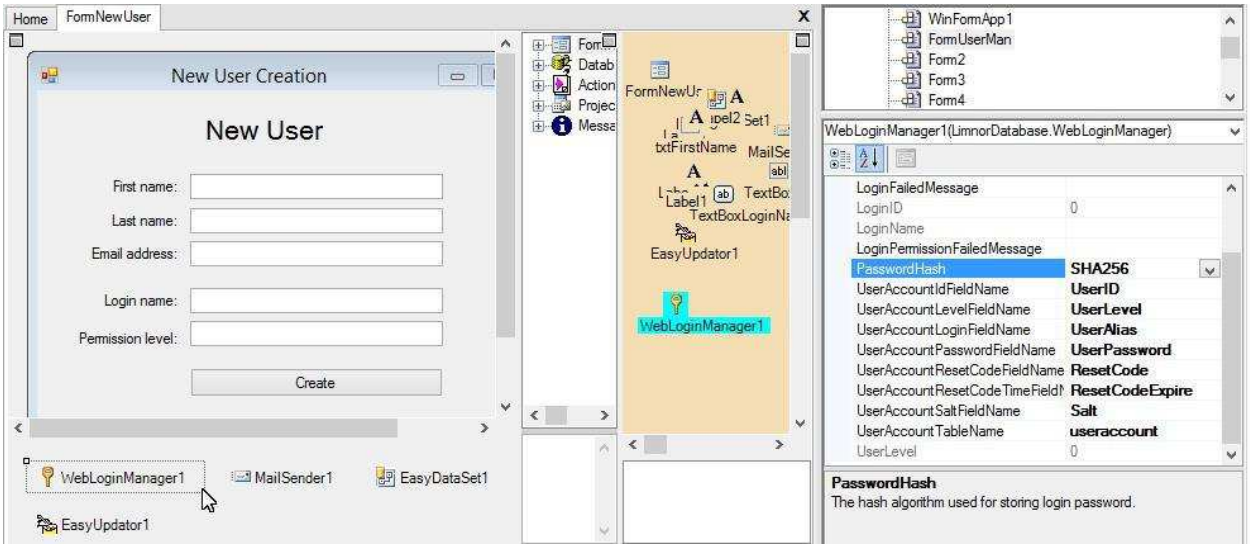
For developing data entry forms, see http://www.limnor.com/support/Limnor%20Studio%20-%20User%20Guide%20-%20Part%20VI.pdf. Chapter 5 “Data Binding” describes data entry using EasyDataSet. Chapter 8 “Data Entry Using EasyGrid” describes data entry using EasyDataGrid.

For creating and modifying a user table, one special issue is to securely handle password storage. You do not want to save plain password text in the database. The issue arises when a user wants to change his/her password; when a user forgets his/her password; when a new user record is created and a new password is to be assigned to the new record. The Login Manager Component makes it easy to address the issue properly.

Add new user

For this sample project, we send initial randomly created password to a new user via email. The operator of the program will not see the new password. On receiving the new password, the new user is supposed to change the password. If your application allows a new user to create his/her account, like some web applications do, then the password is entered by the user, not randomly created.

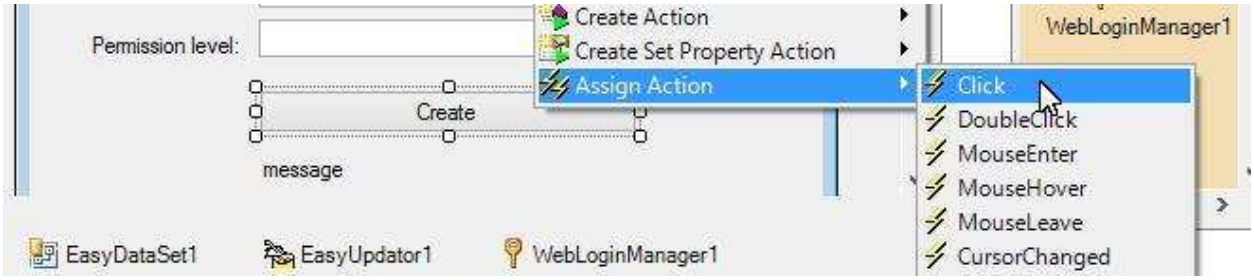
Note that the properties of the Login Manager Component must be set the same as we did for the Login Manager component in the login form.



We want to make such programming: the operator clicks the “Create” button; verify login name and email address are not empty; EasyDataSet is used to query the database to see if the login name is in use; EasyUpdator is used to create a new user record; WebLoginManager is used to set a random password to the new user record; MailSender is used to send the new random password to the new user. These programming tasks are straight forward. Let’s do it one by one.

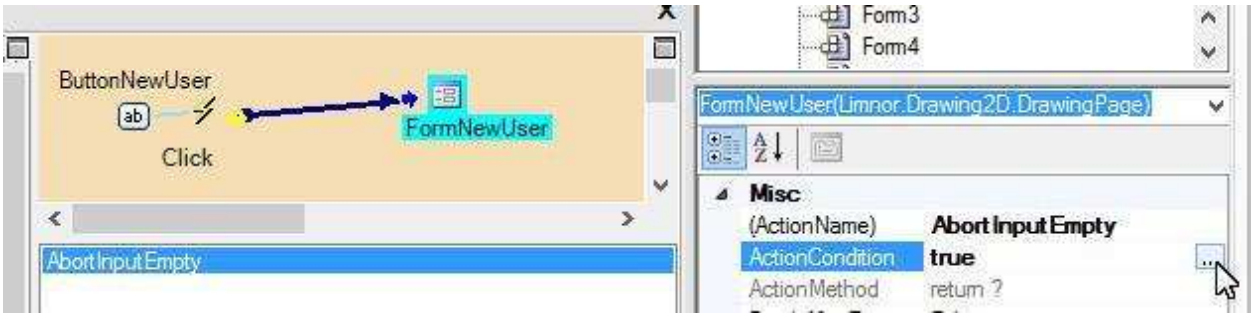
Input verification

Each programming starts by right-clicking the button; choose “Assign Action”; choose “Click” event.

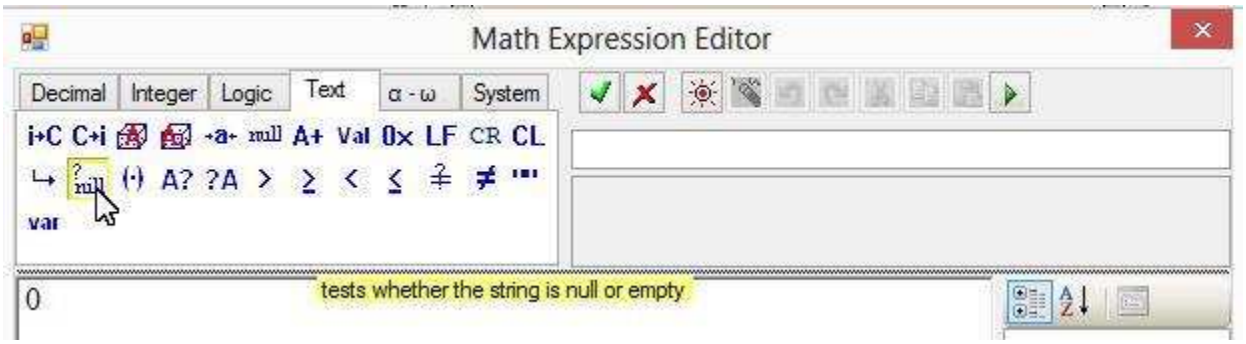


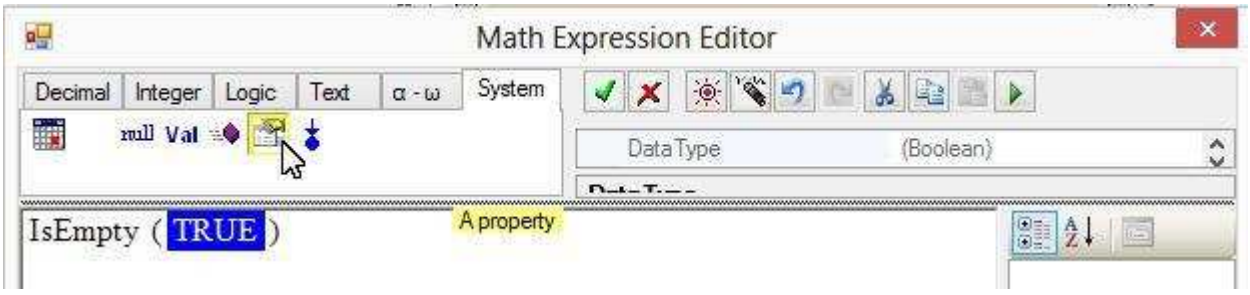
Select the Abort action:

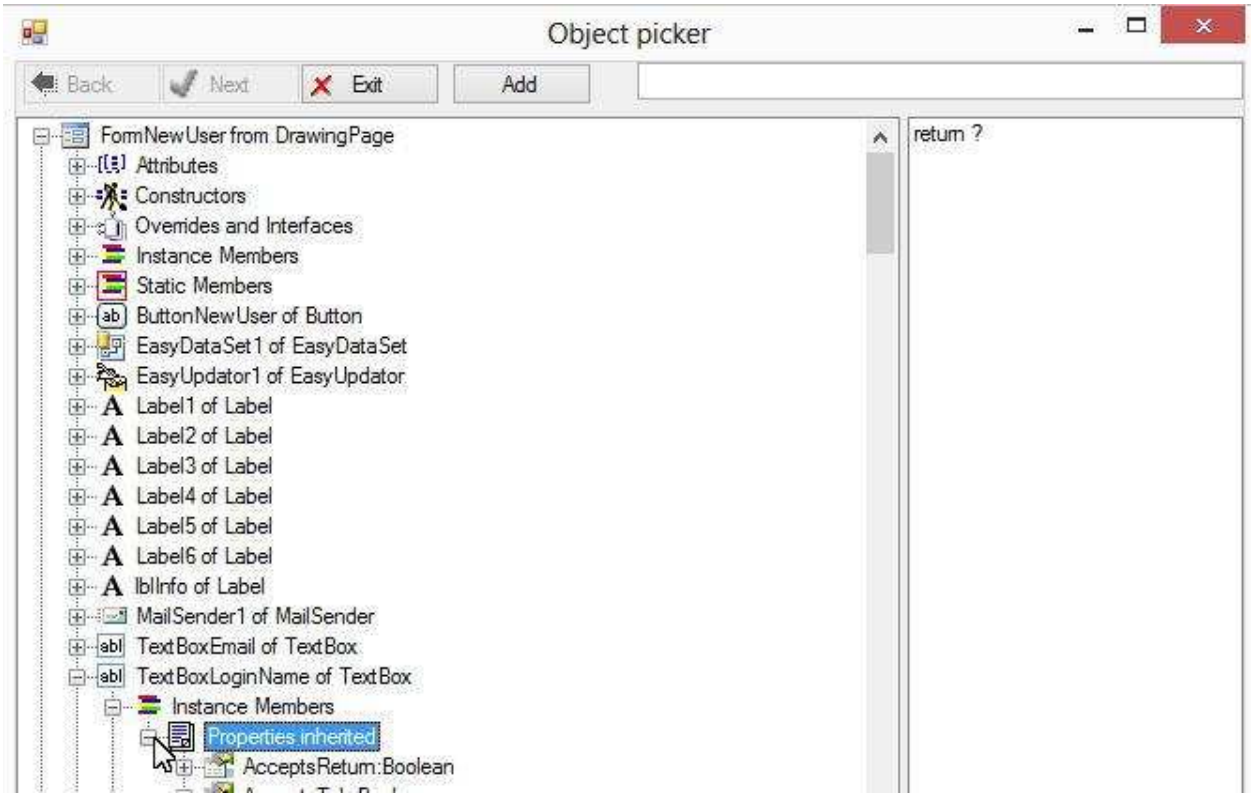
The abort action is assigned to the button. Rename it and set its ActionCondition to verify inputs:



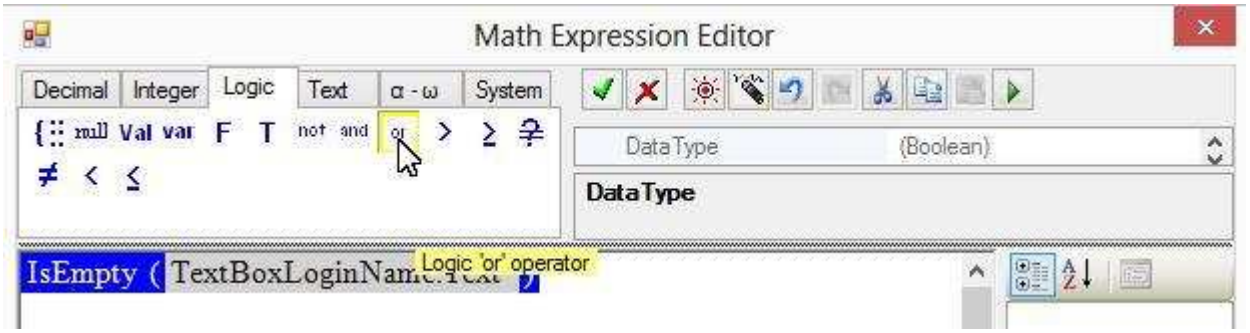
The conditions are that the login name or email address is empty:







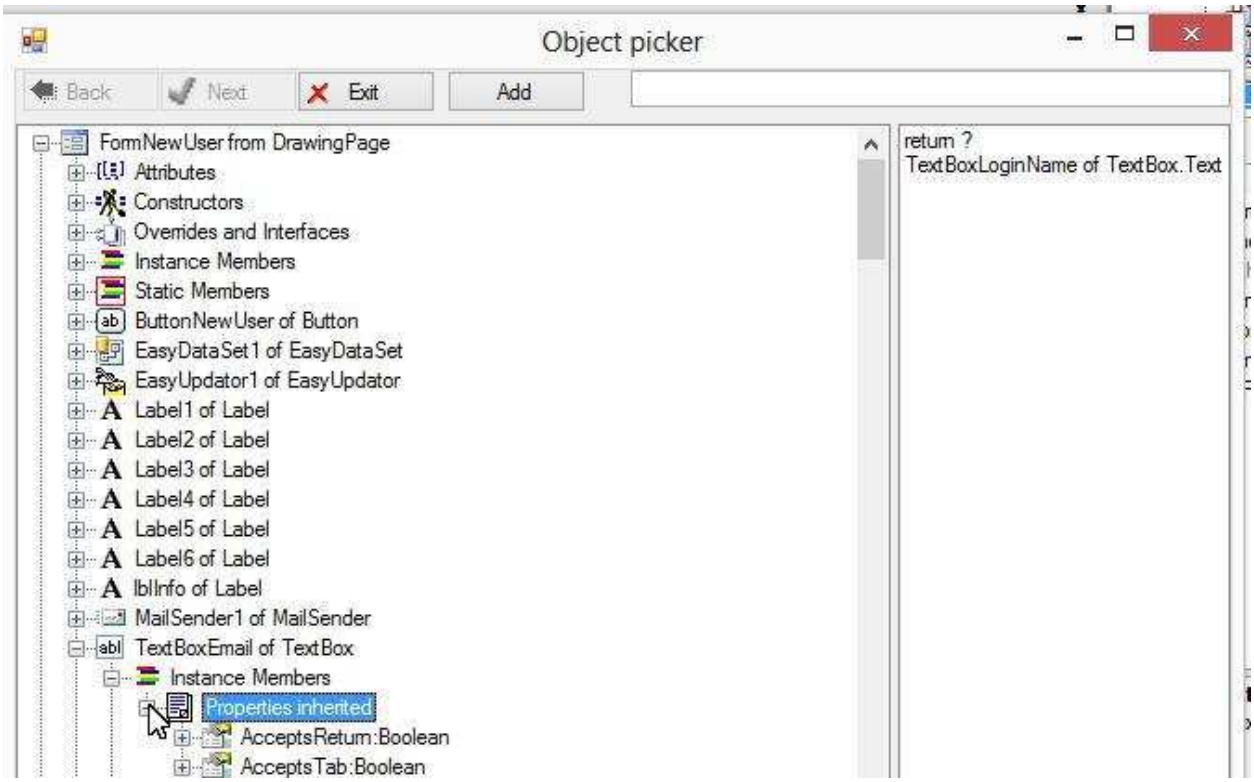
Click “IsEmpty” and then click “or”:

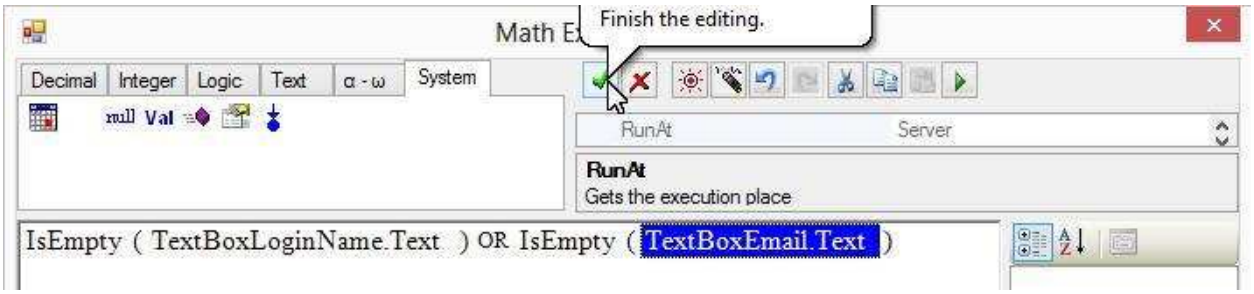


Click “x” and then click “?null”:

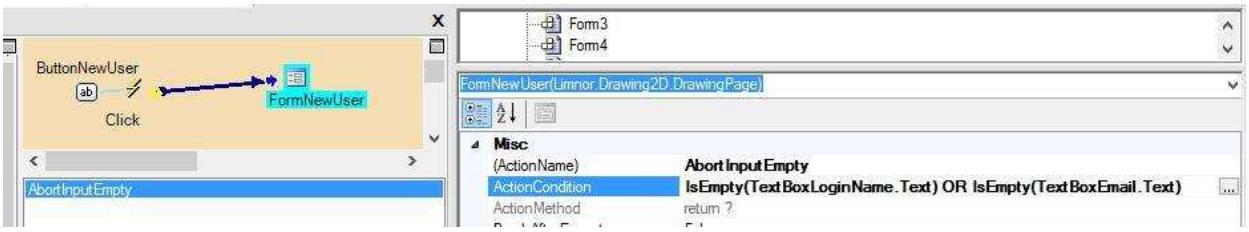
Click “x” inside “IsEmpty”, then click the property icon:

Select Text of the email address text box:



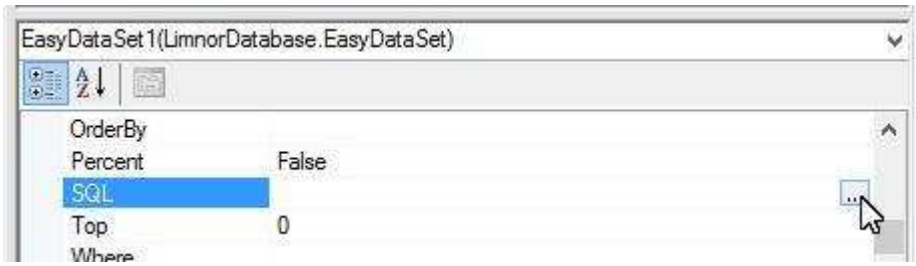


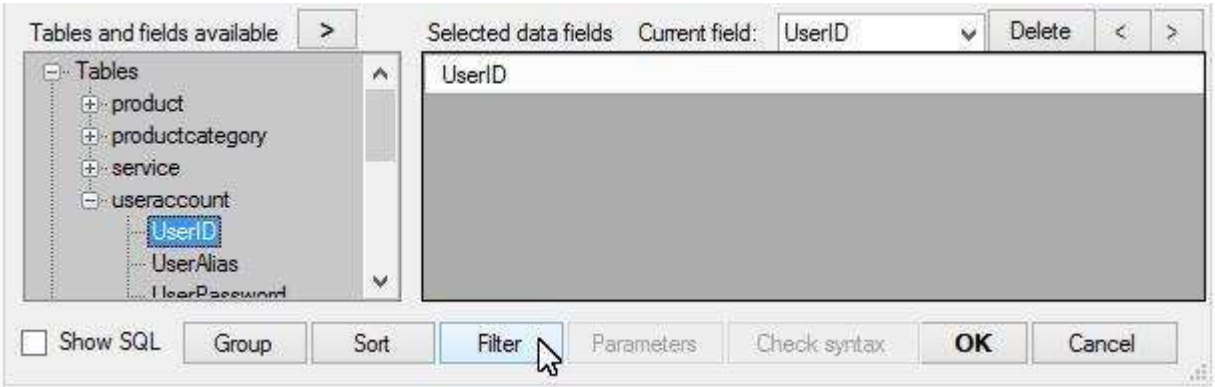
The input verification is done:

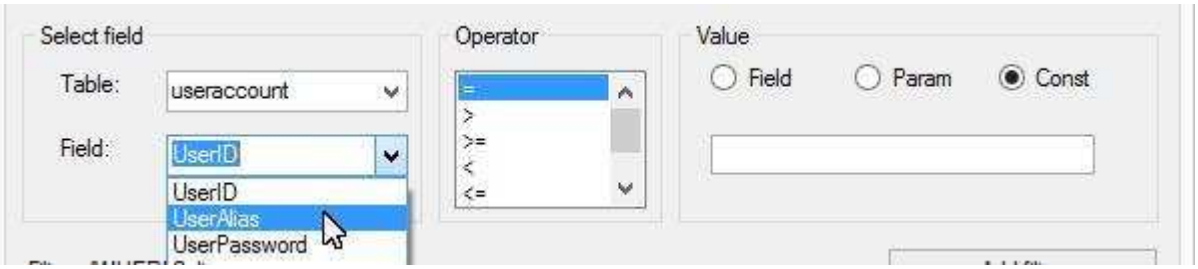


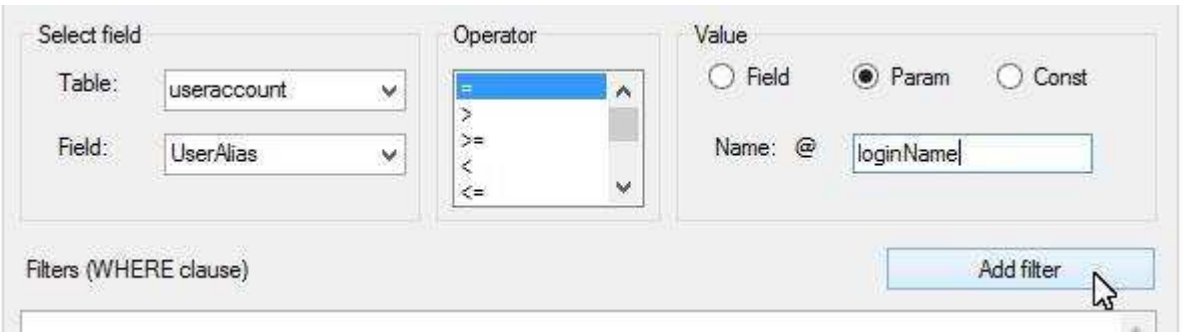
Login name usage checking

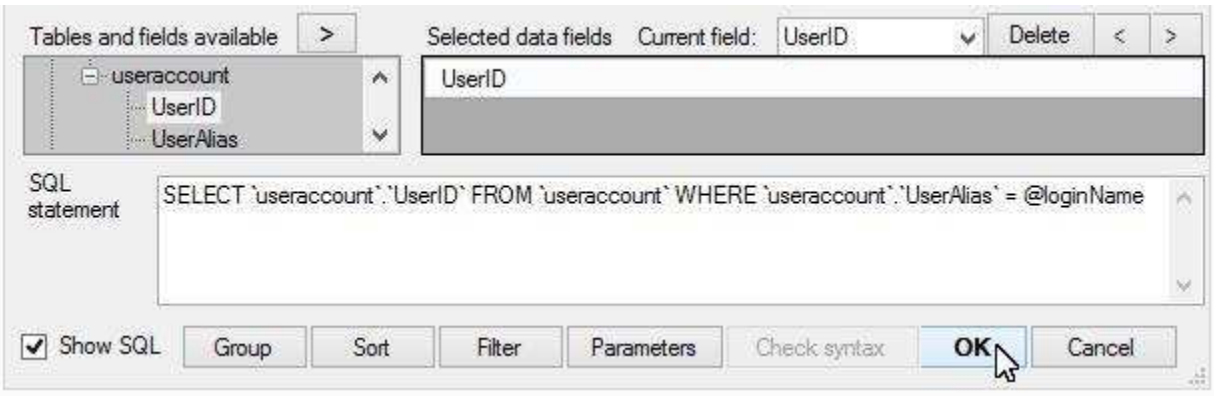
The SQL property of the EasyDataSet is set to search UserAlias field:



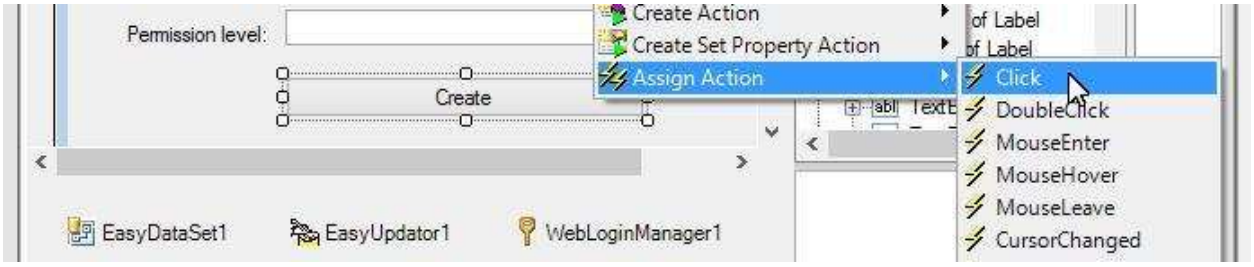


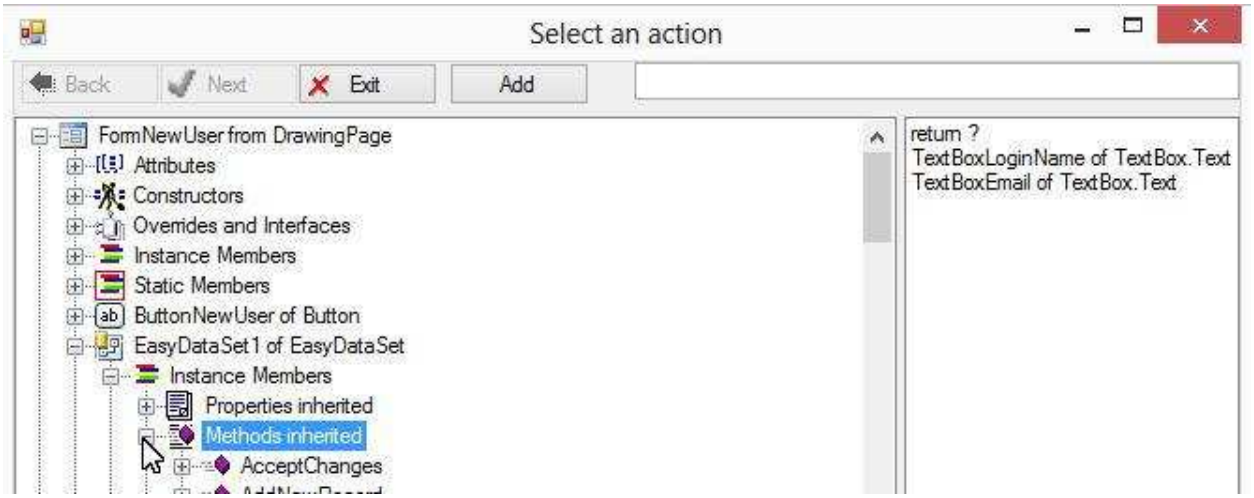




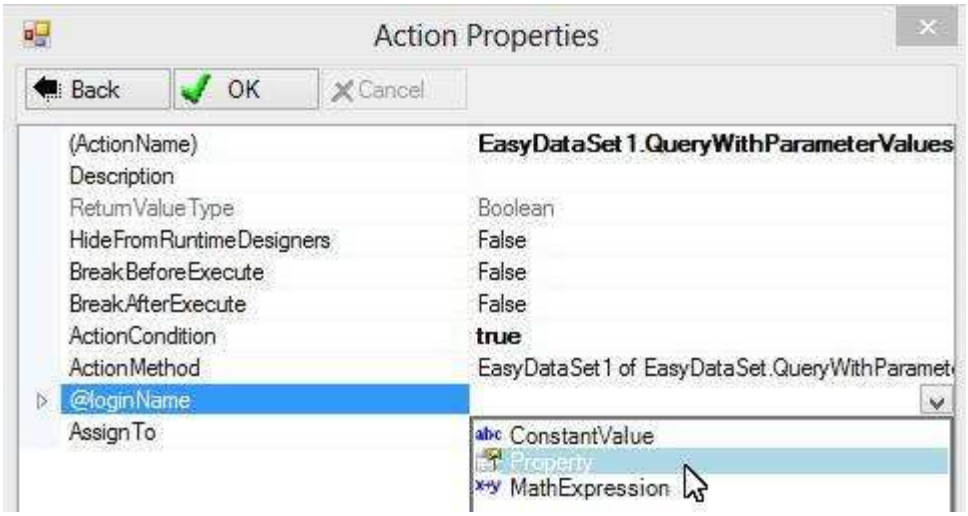


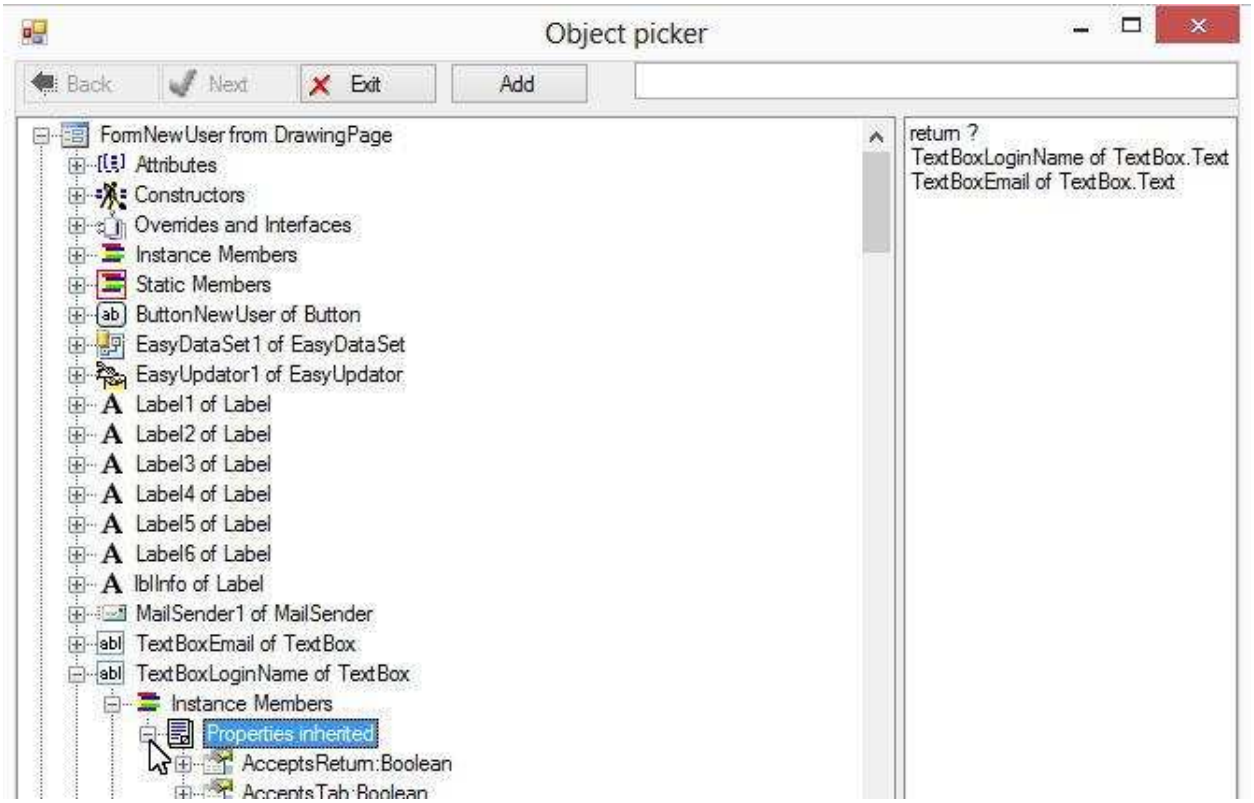
Now we may use the EasyDataSet to verify that the login name is not in use:

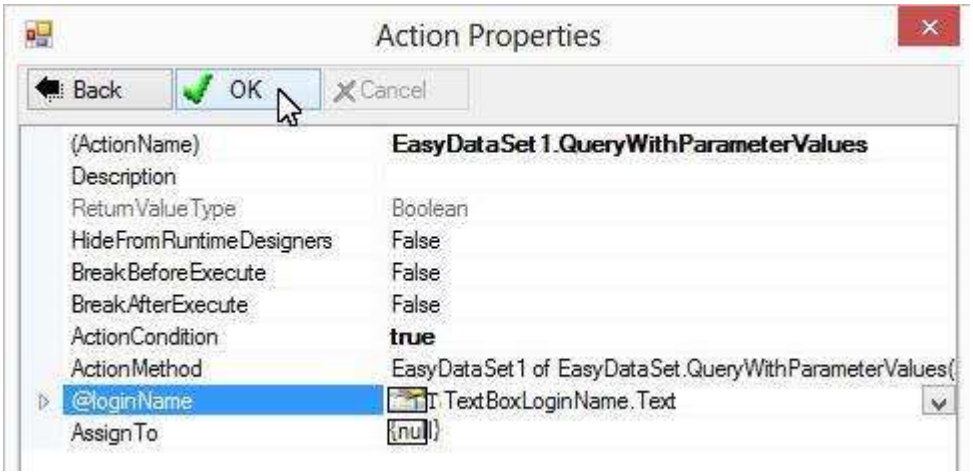

Permission level:
Create
Create Action
Create Set Property Action
Assign Action
Click
DoubleClick
MouseEnter
MouseHover
MouseLeave
CursorChanged
EasyDataSet1
EasyUpdator1
WebLoginManager1


Select an action
Back
Next
Exit
Add
FormNewUser from DrawingPage
Attributes
Constructors
Overrides and Interfaces
Instance Members
Static Members
ButtonNewUser of Button
EasyDataSet1 of EasyDataSet
Instance Members
Properties inherited
Methods inherited
AcceptChanges
return ?
TextBoxLoginName of TextBox.Text
TextBoxEmail of TextBox.Text

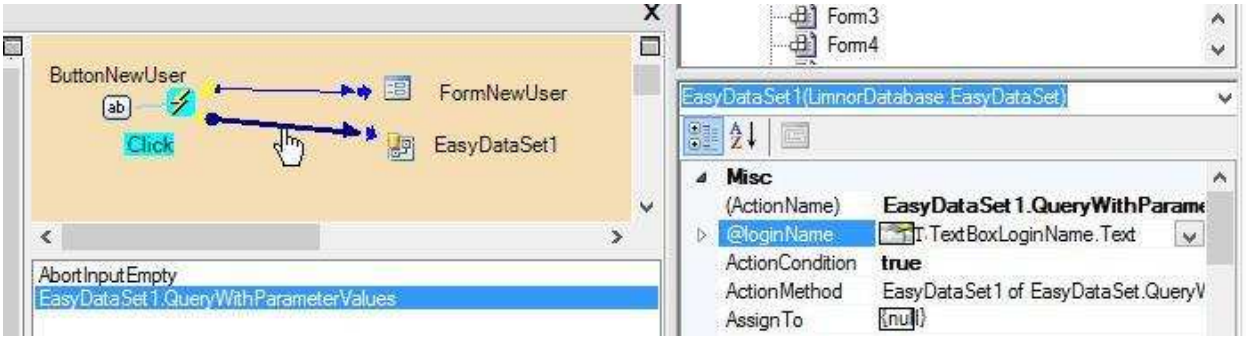

Select an action
Back
Next
Exit
Add
OnWebPageLoadedAfterEventHandlerCreations(StringCollection)
Query()Boolean
QueryWithParameterValues(String)Boolean
QueryWithWhere(String)Boolean
QueryWithWhereDynamic(String)Boolean
return ?
TextBoxLoginName of TextBox.Text
TextBoxEmail of TextBox.Text


Action Properties
Back
OK
Cancel
(ActionName)
EasyDataSet1.QueryWithParameterValues
Description
ReturnValueType
Boolean
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
EasyDataSet1 of EasyDataSet.QueryWithParamet
@loginName
AssignTo
ConstantValue
Property
MathExpression

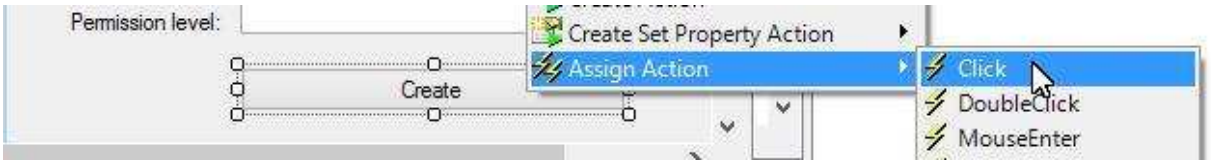




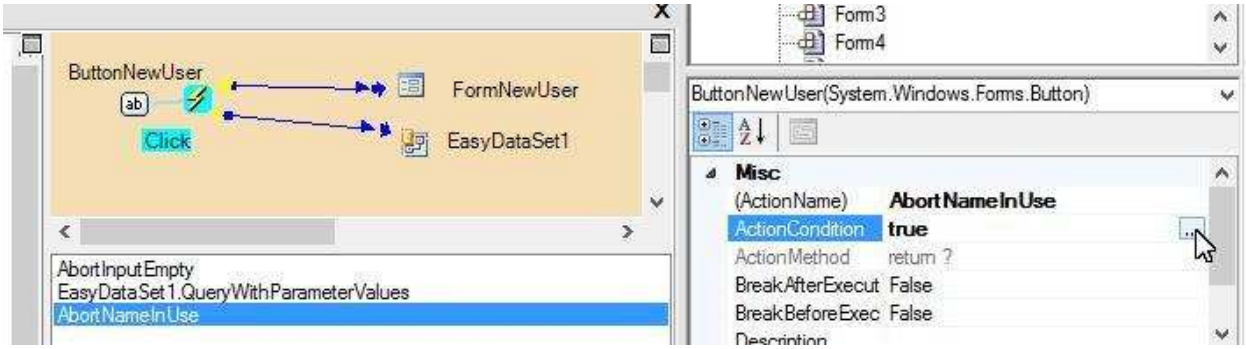
The action is created and assigned to the button:



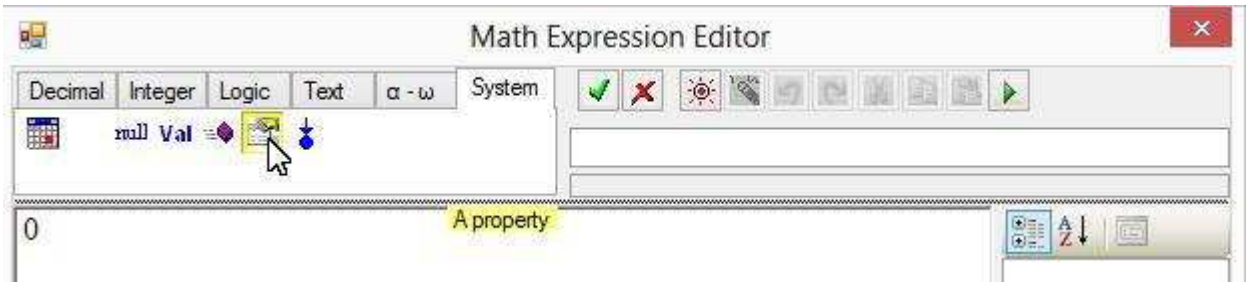
Assign an abort action after searching the database:

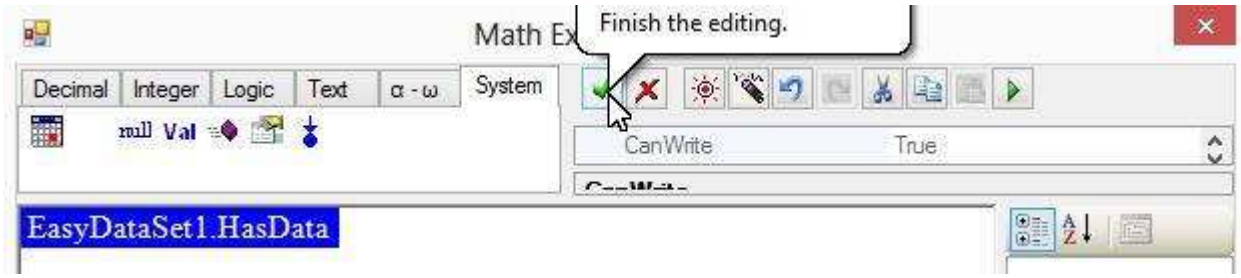


An abort action is assigned to the button. Rename it and set its ActionCondition:

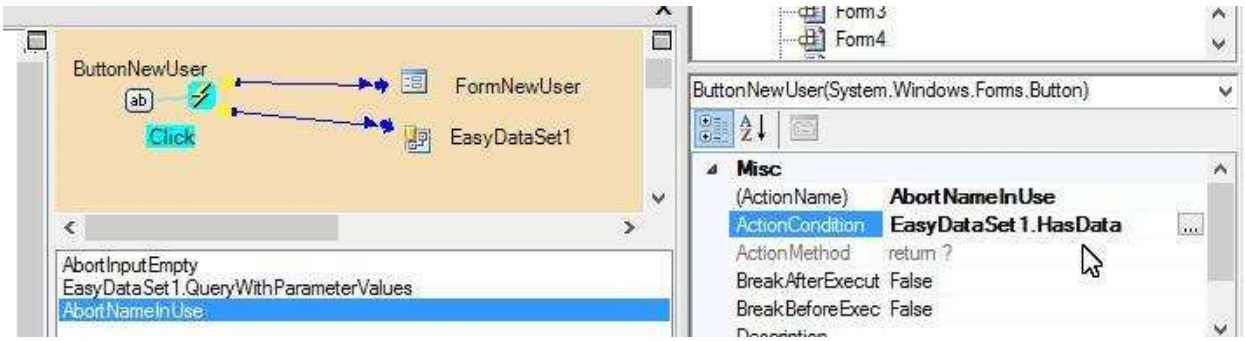


Select HasData property of the EasyDataSet:



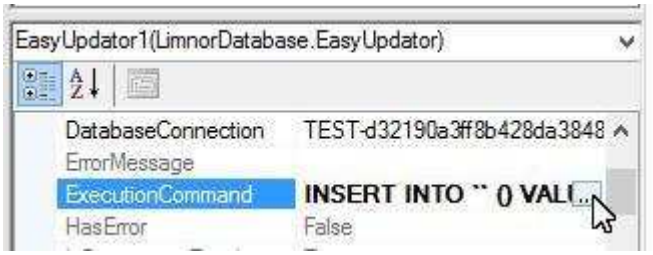


The login name usage checking is done:

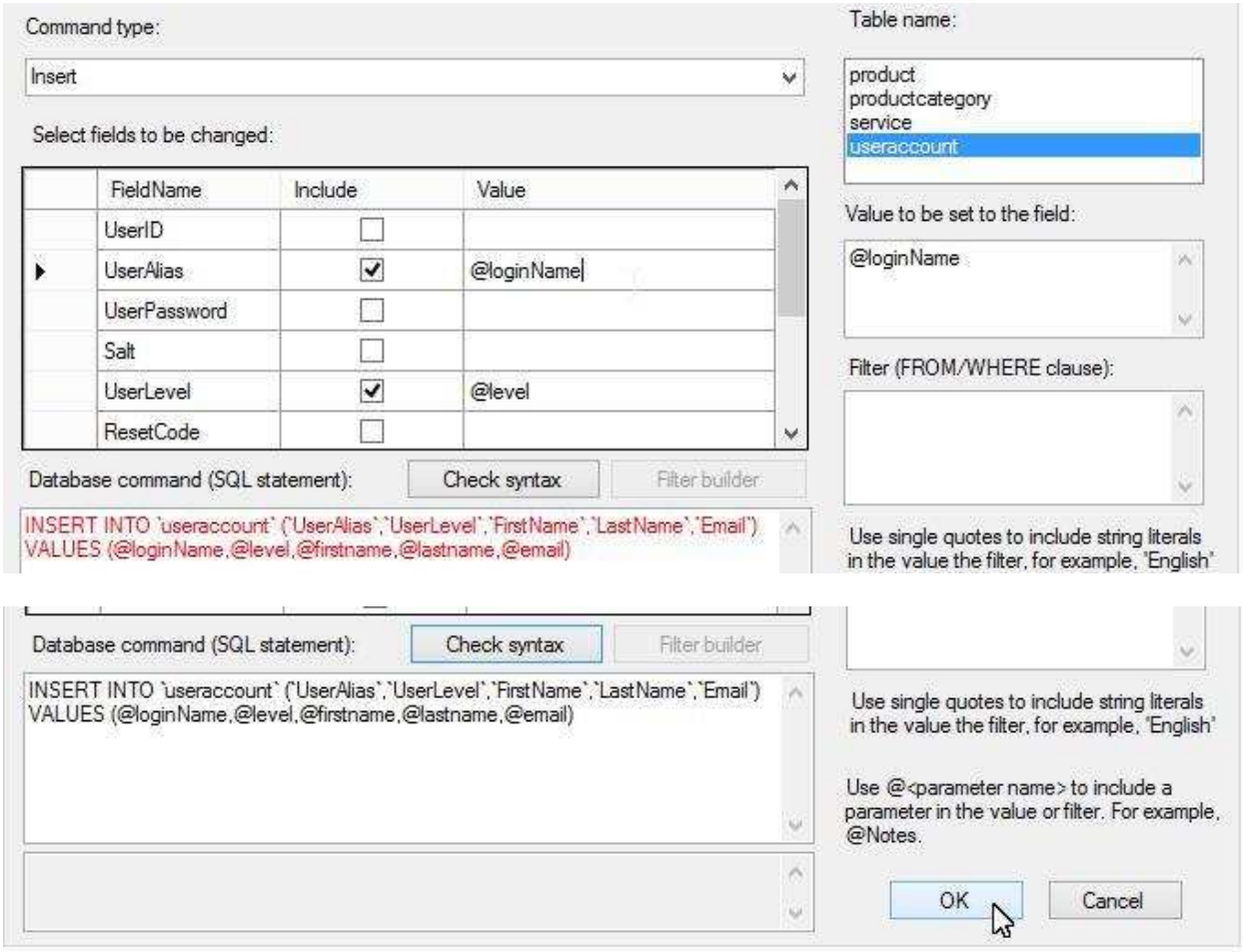


Create new user record

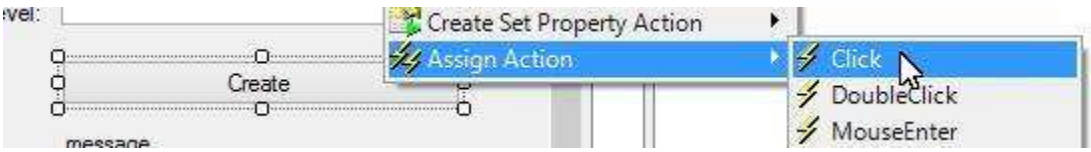
The ExecuteCommand property of the EasyUpdator is set to insert a new record into the user table:

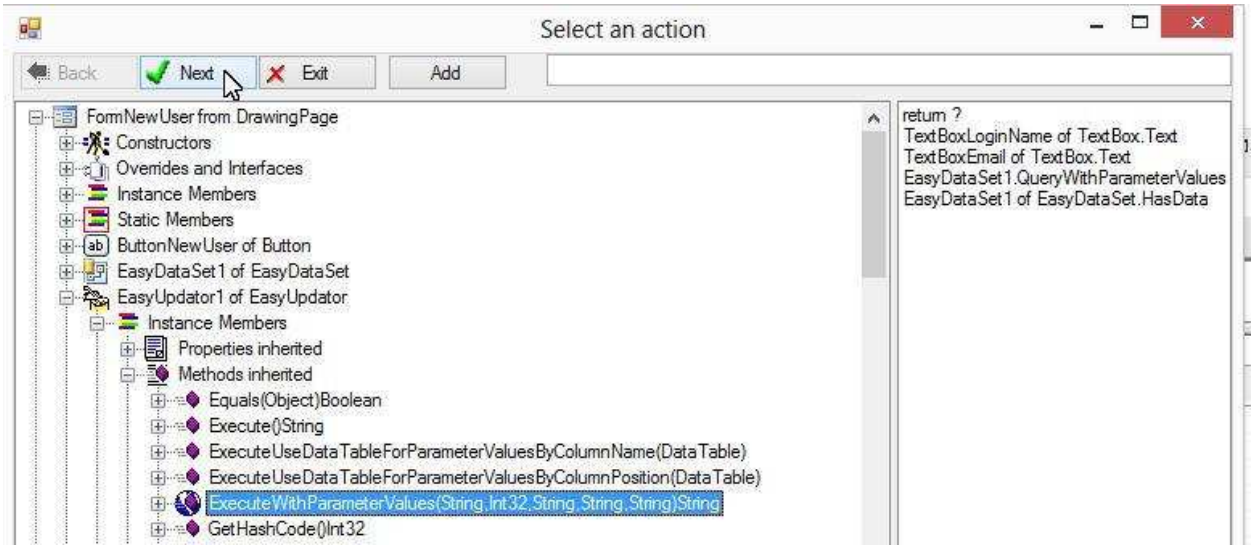


Check the fields to be used in creating new record and give a parameter for each field:

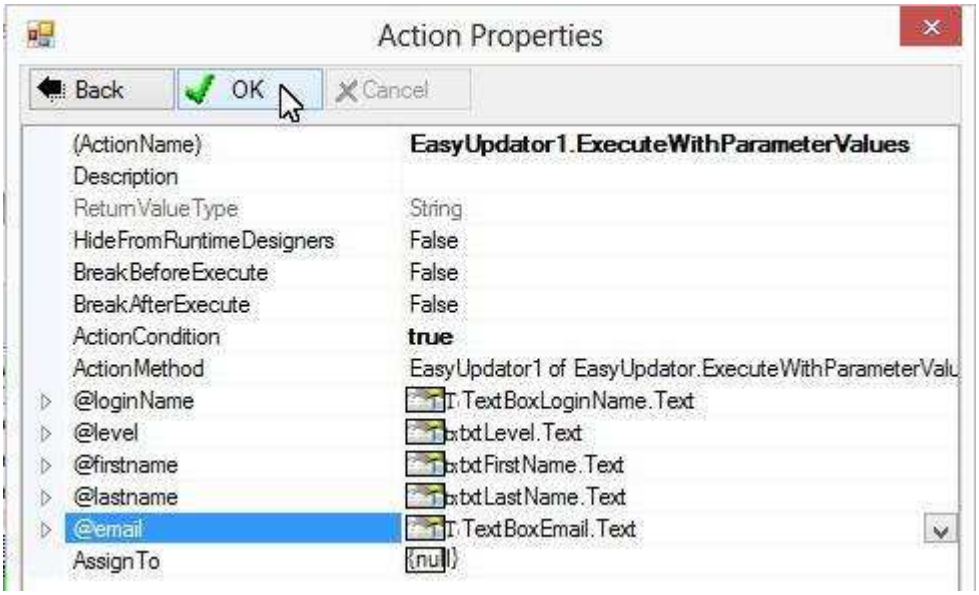


Now we may use it to create a new user record:

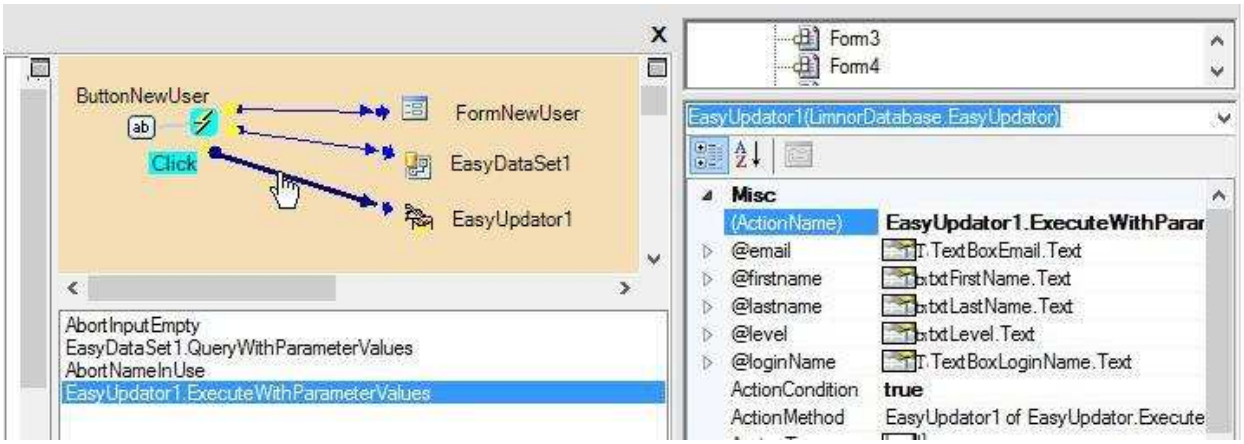




Pass Text property of the text boxes to the action parameters:

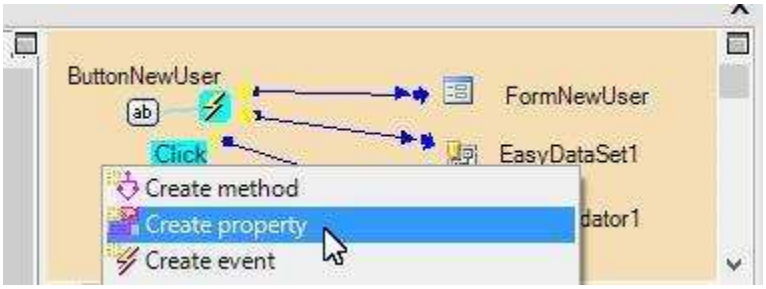


The action is created and assigned to the button:

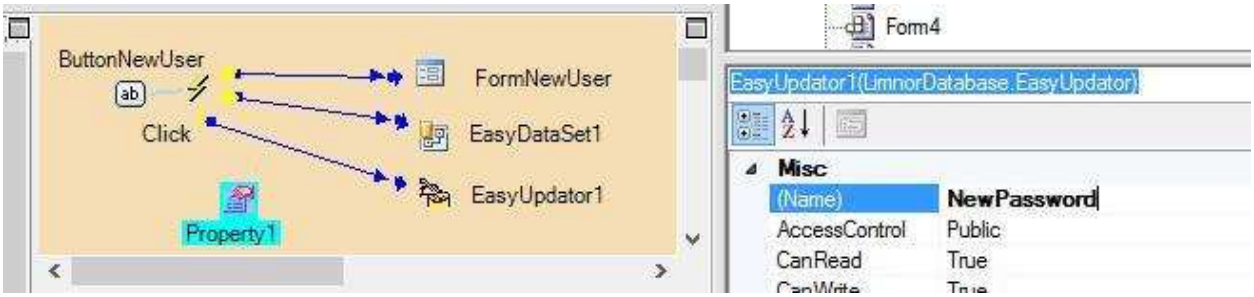


Create new password

Create a property to hold the new password so that we may send it to the new user:

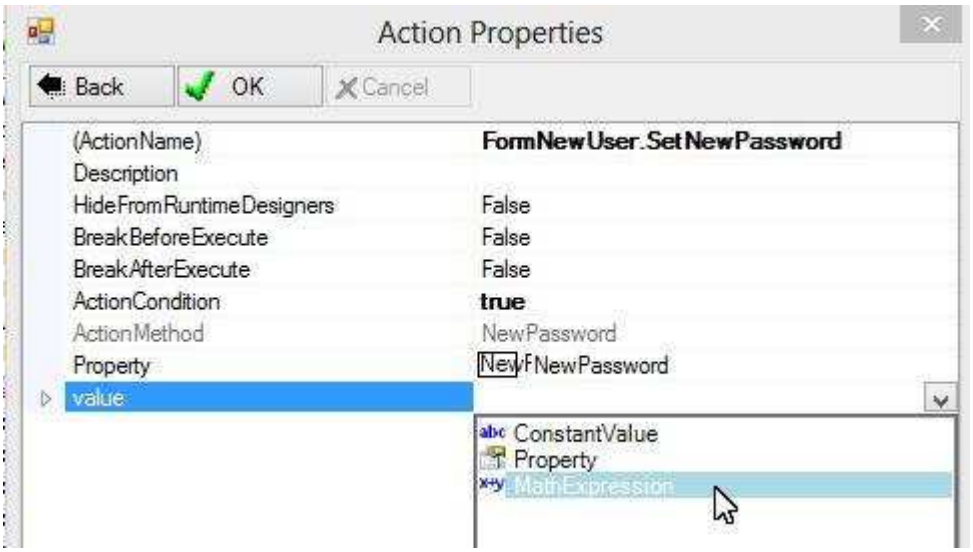


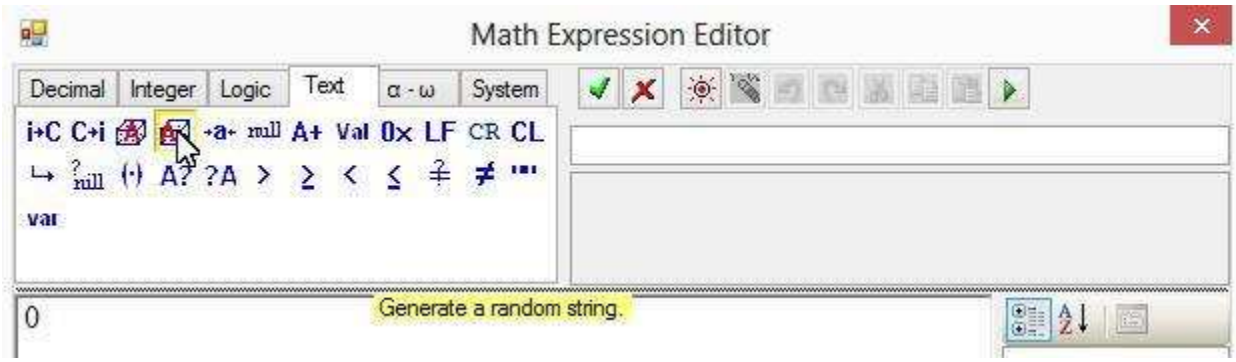
Rename it to “NewPassword”:



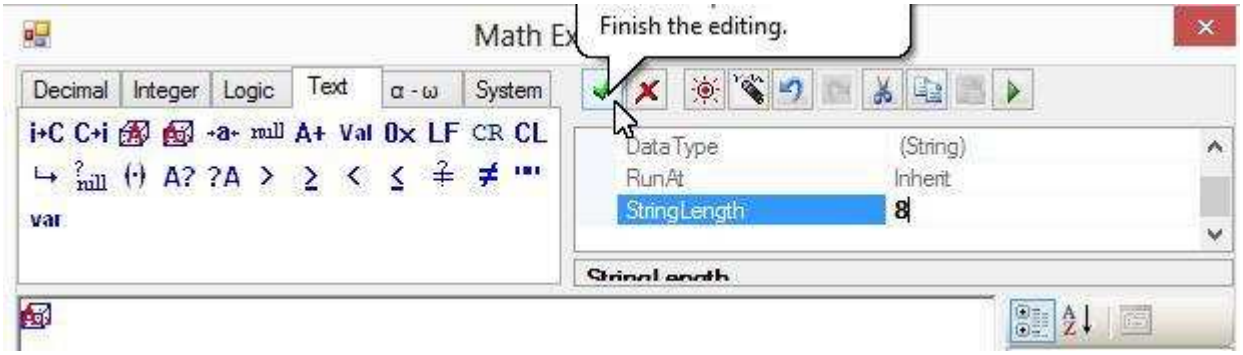
Set the property to a random string of 8 characters:

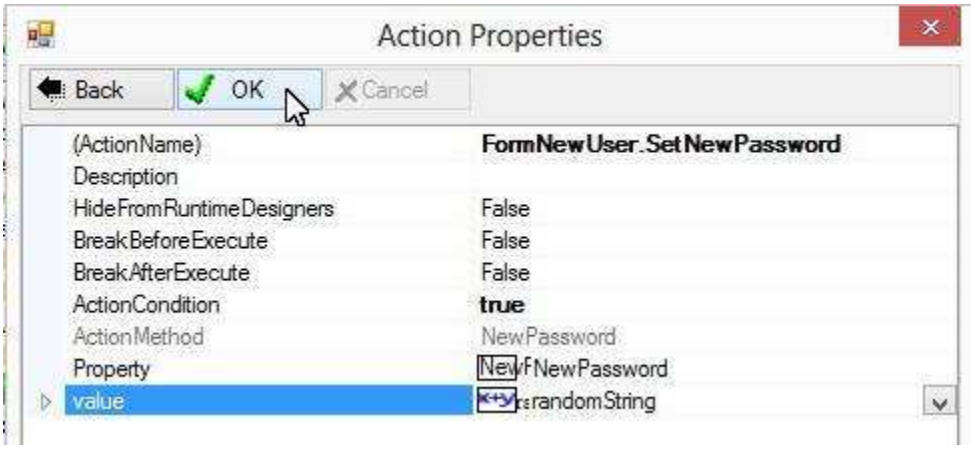
Set the value to a random string:



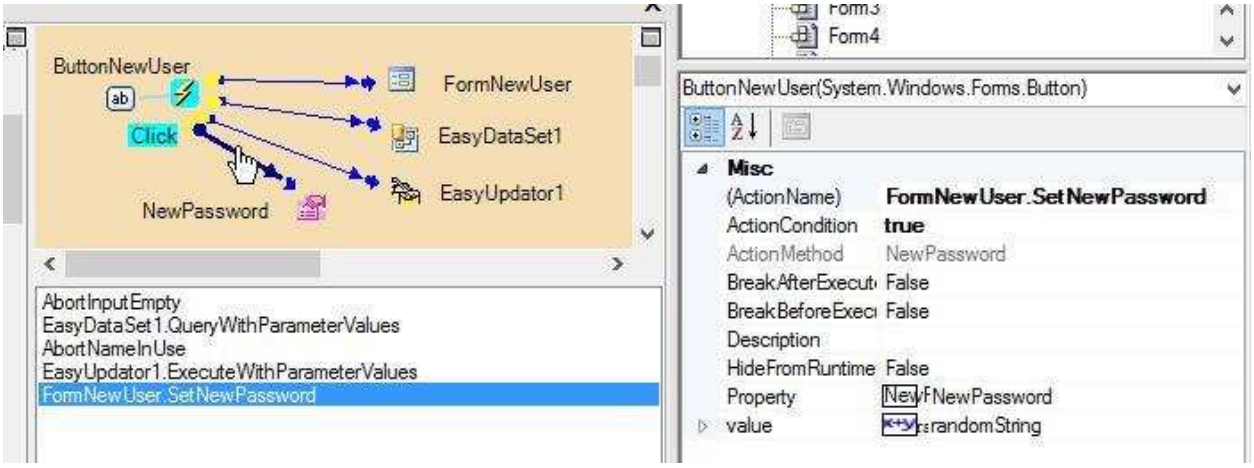


Set length to 8:

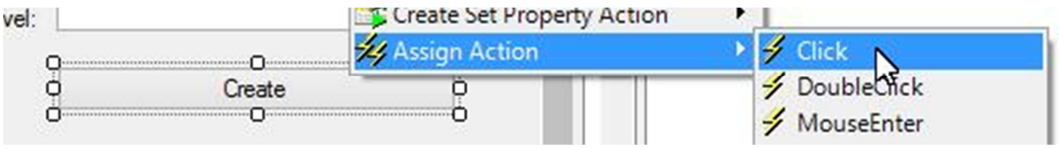




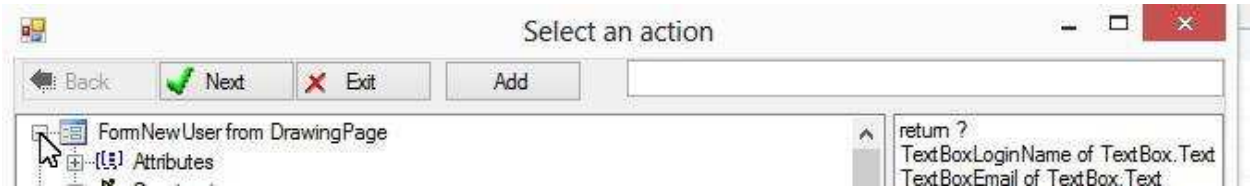
The action is created and assigned to the button:

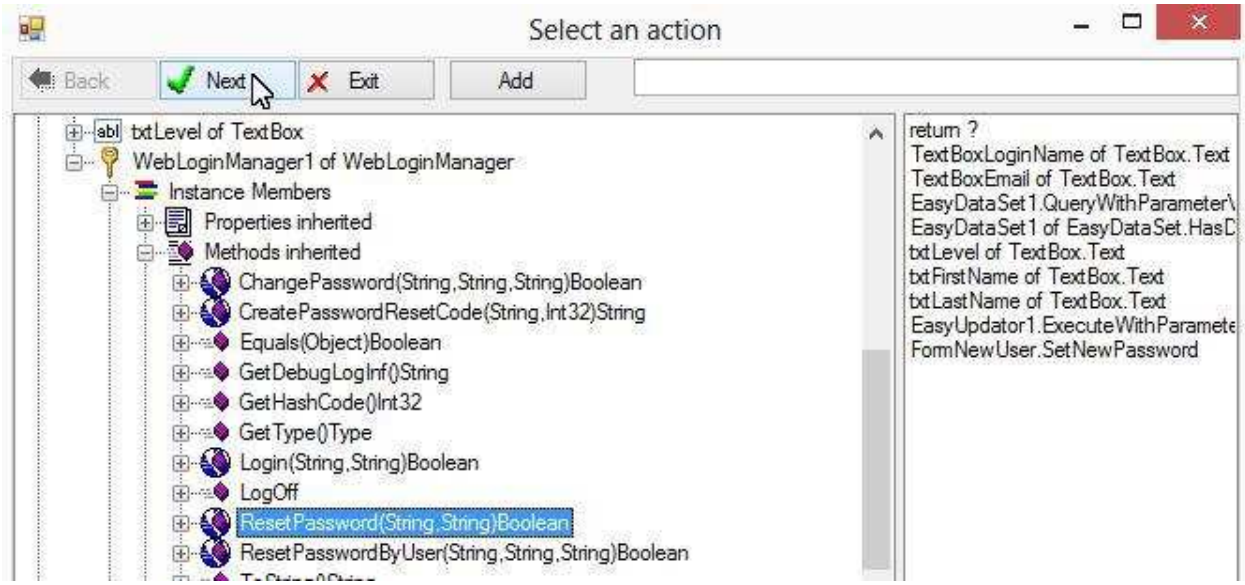


Now we can assign the new password to the new user record:

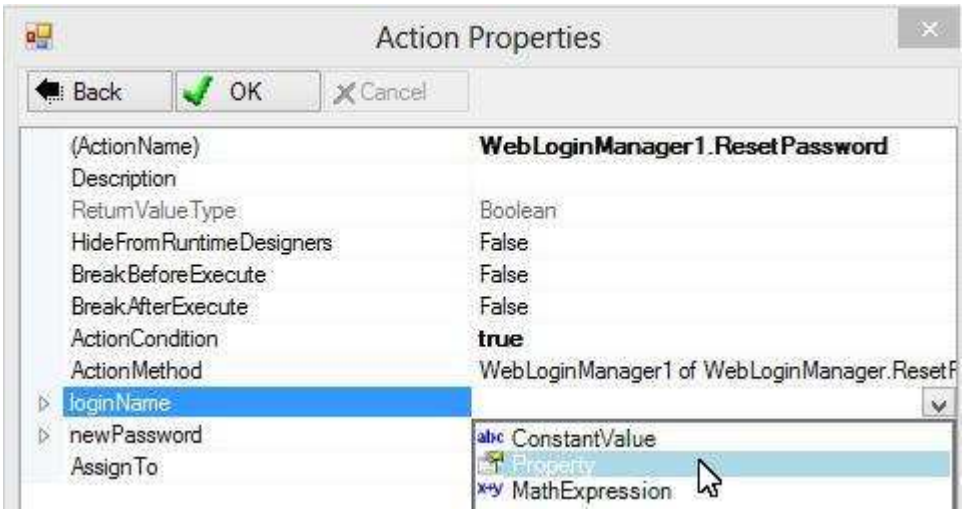


Select ResetPassword method of the Login Manager component:

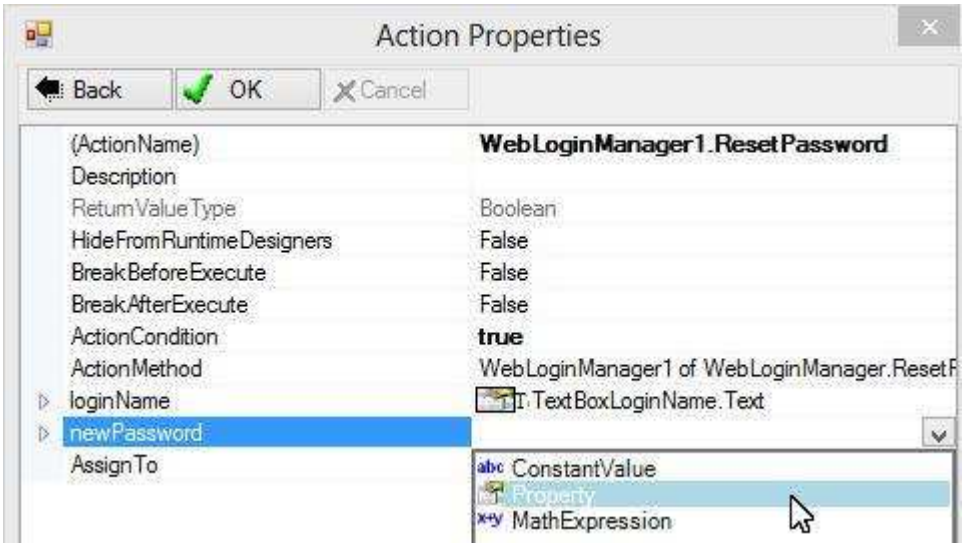




Pass the Text property of the login name text box to the action:

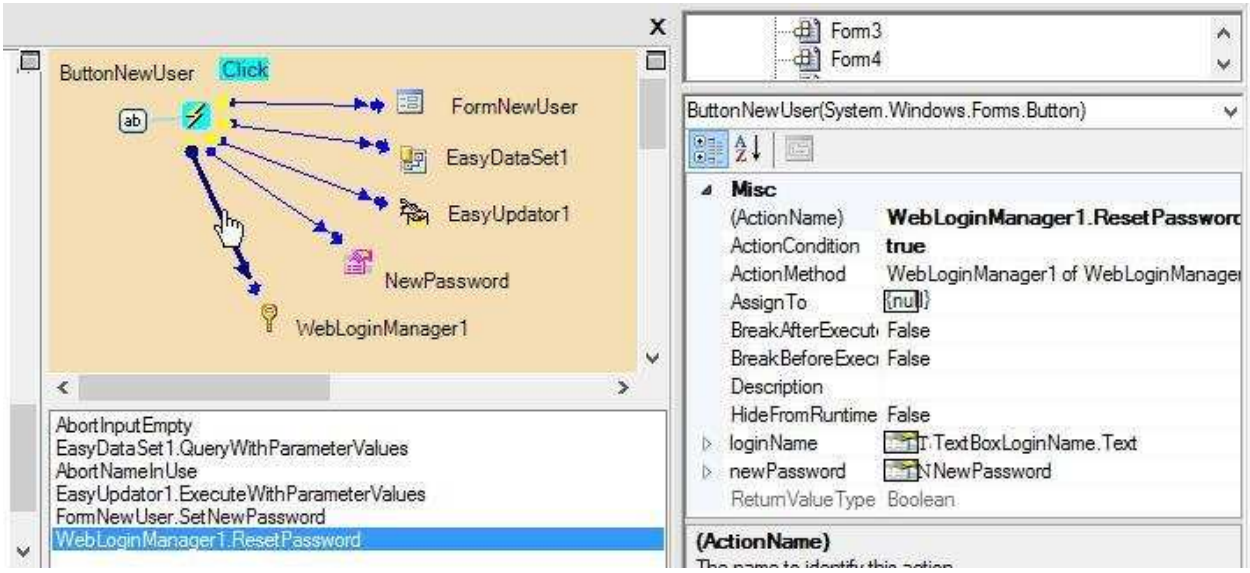


Pass the NewPassword property to the action:



Click OK to create the action:

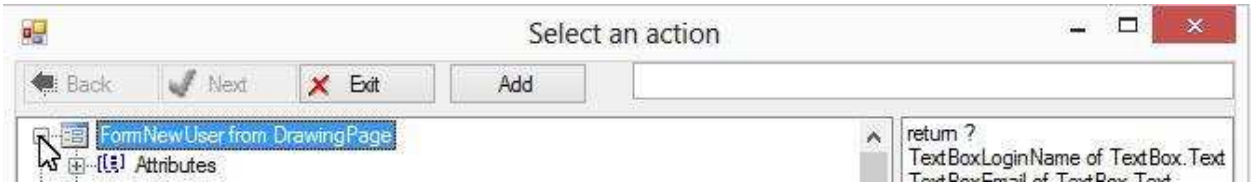
The action is created and assigned to the button:

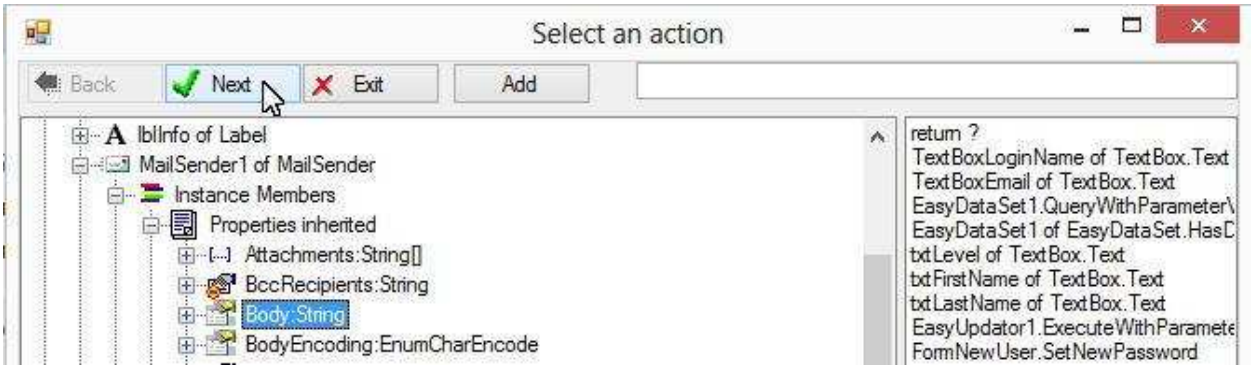


Note that you should never let an end user directly execute a ResetPassowrd action!

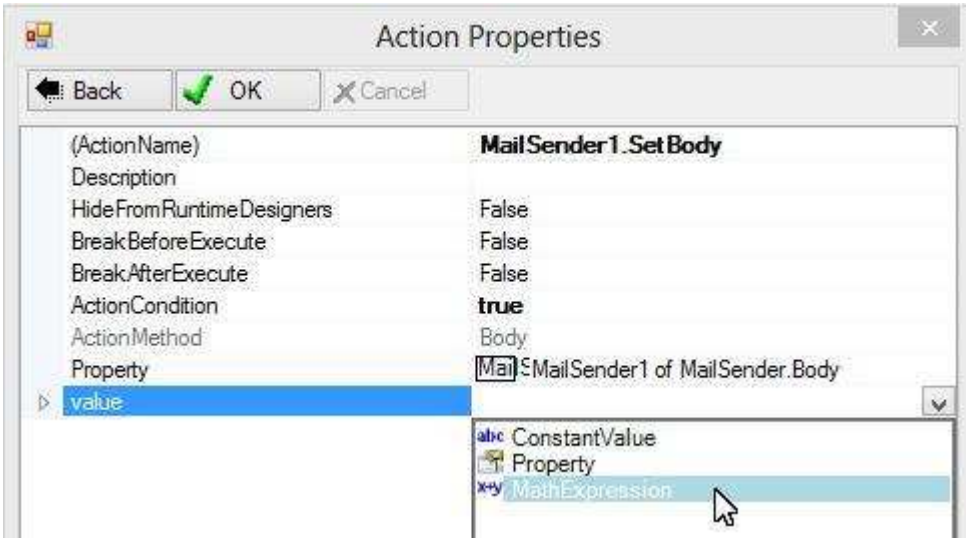
Send password to user

In this sample, we use a MailSender component to send the password to the user. Set the email body to include the new password:

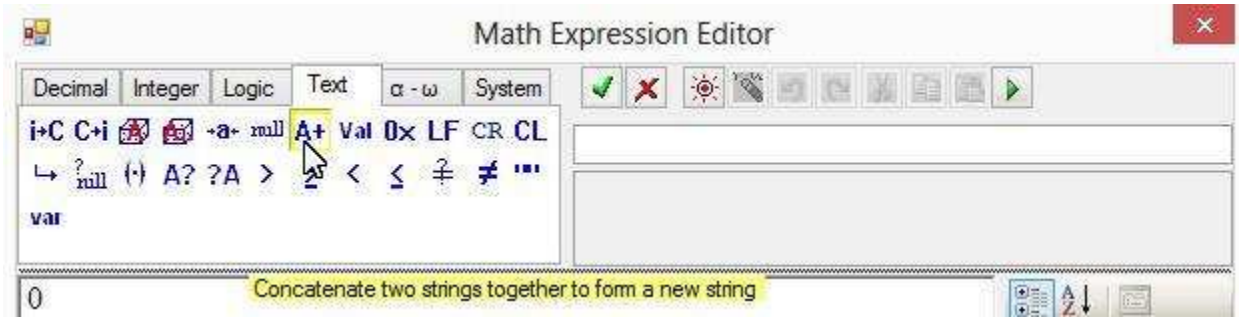




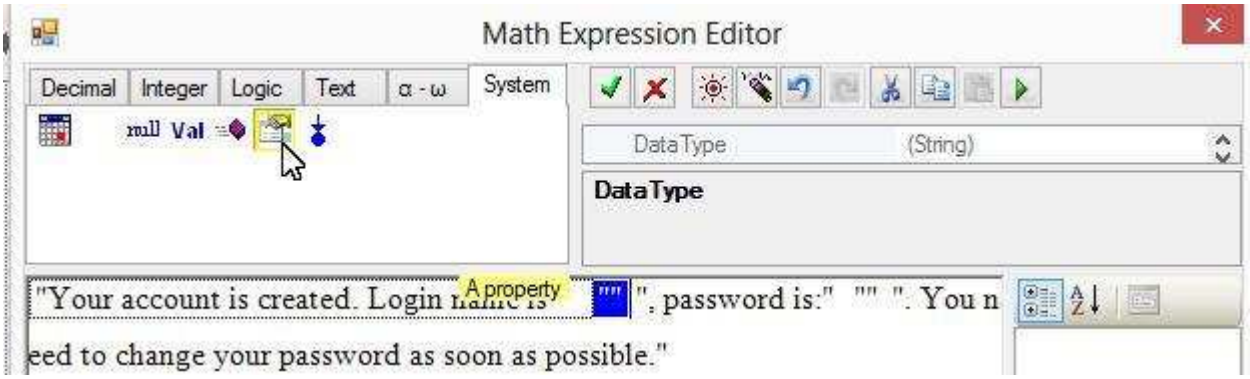
Use an expression to form email body:



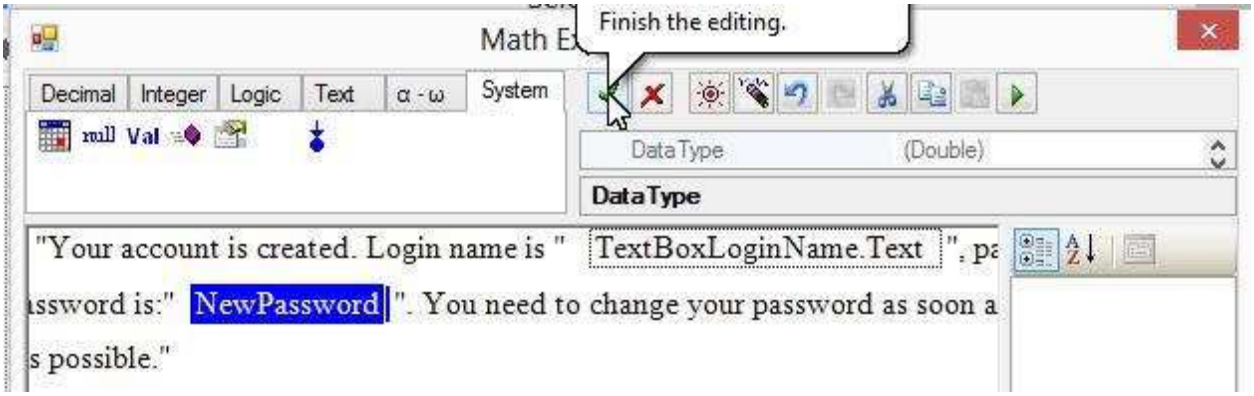
Click A+ a few times to create 5 parts of strings:

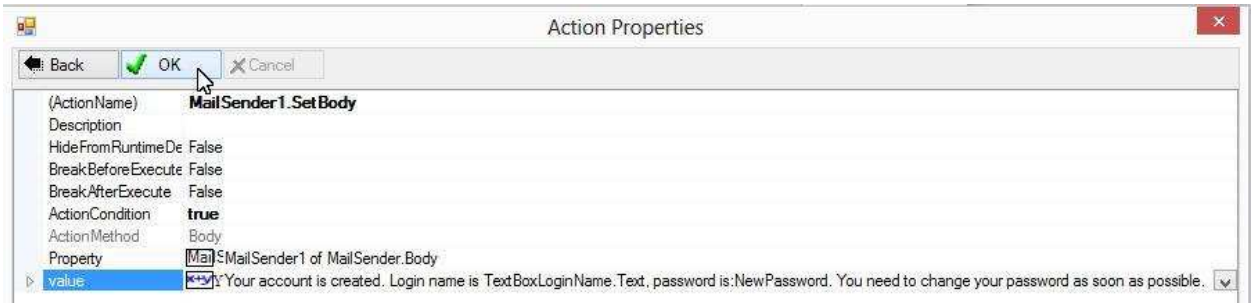


Select login name:

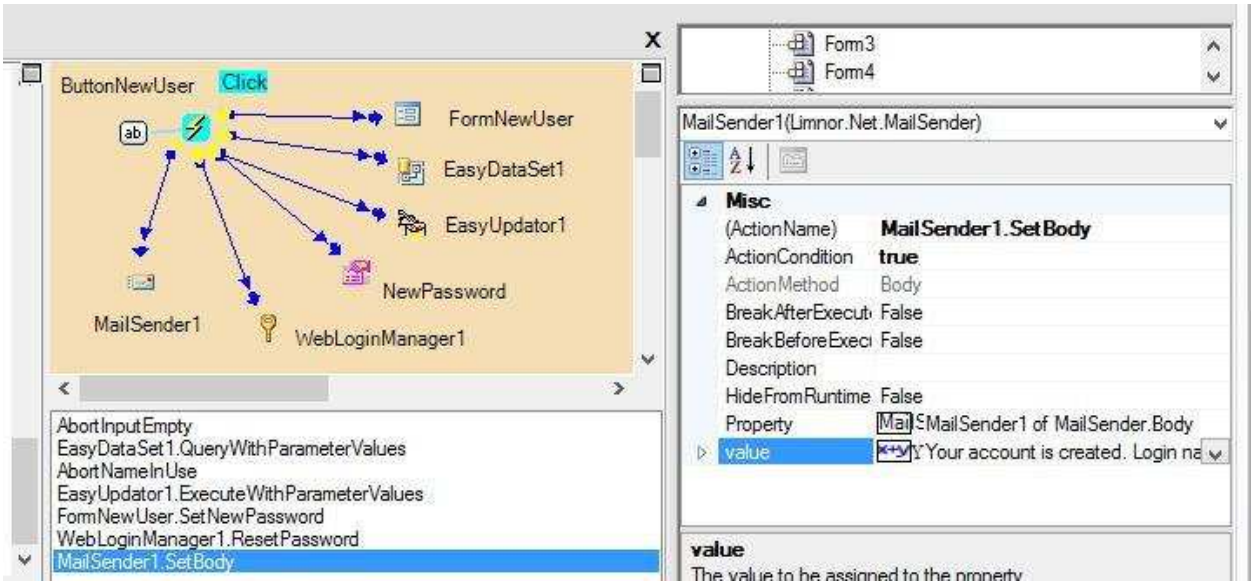


Select the new password into the expression:

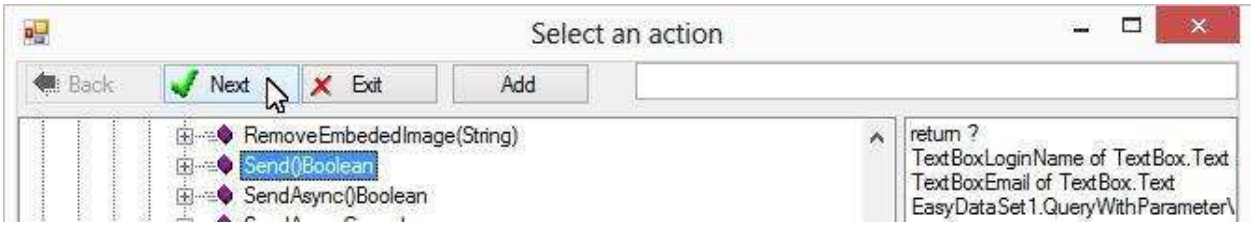


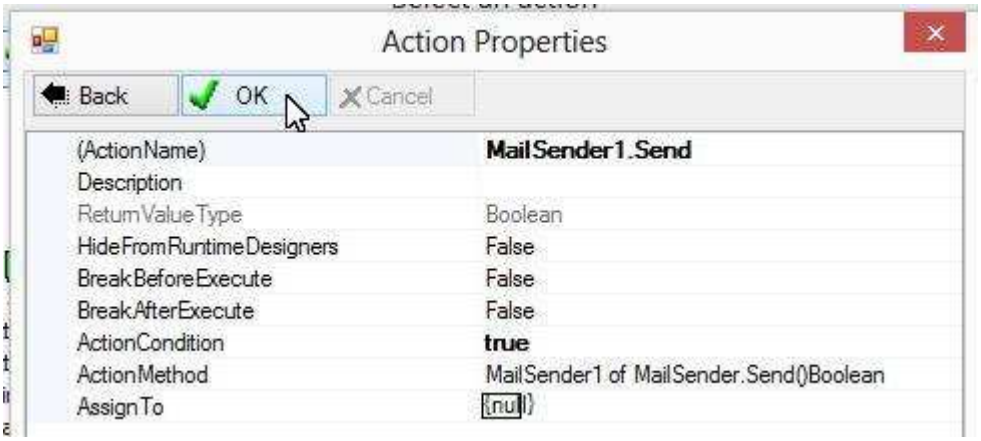


The action is created and assigned to the button:

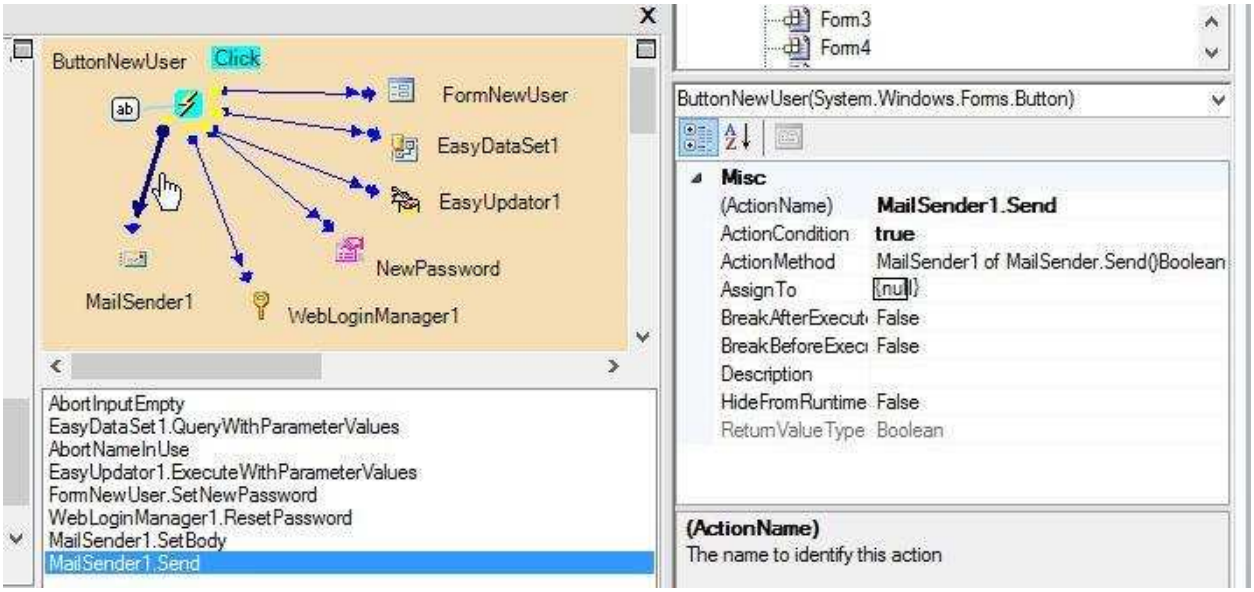


Execute a Send action to send the email:





The action is created and assigned to the button:



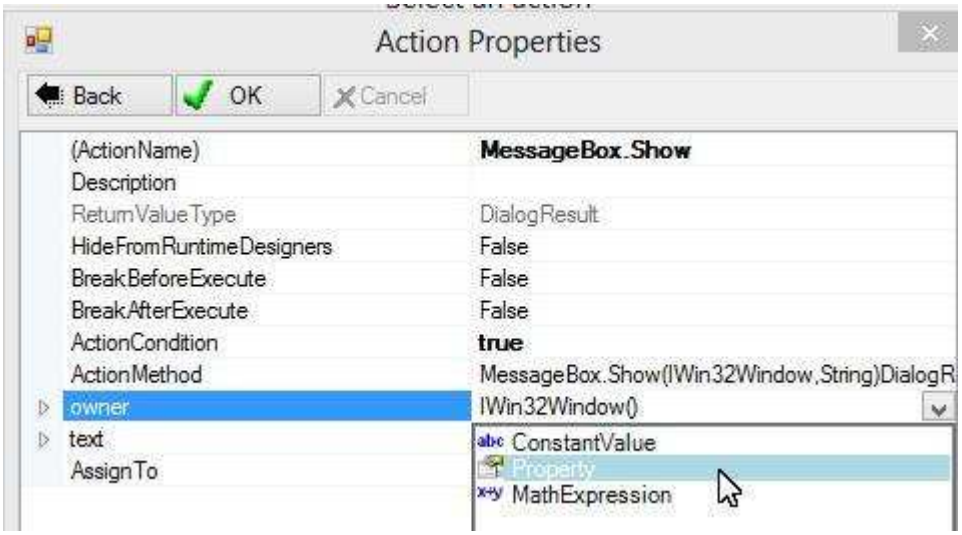
We did not show how to set properties of the MailSender.

Show notification message

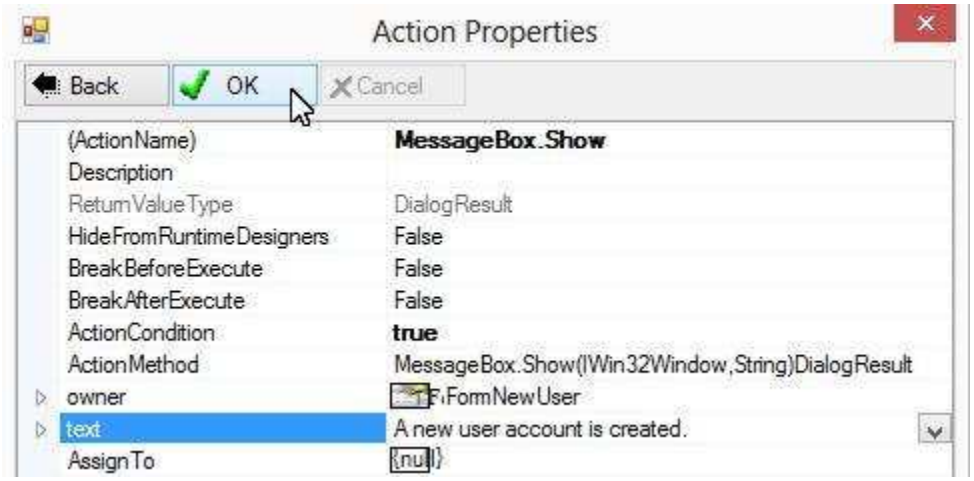
We may show a message box after all actions are executed:

Select a Show method of the MessageBox:

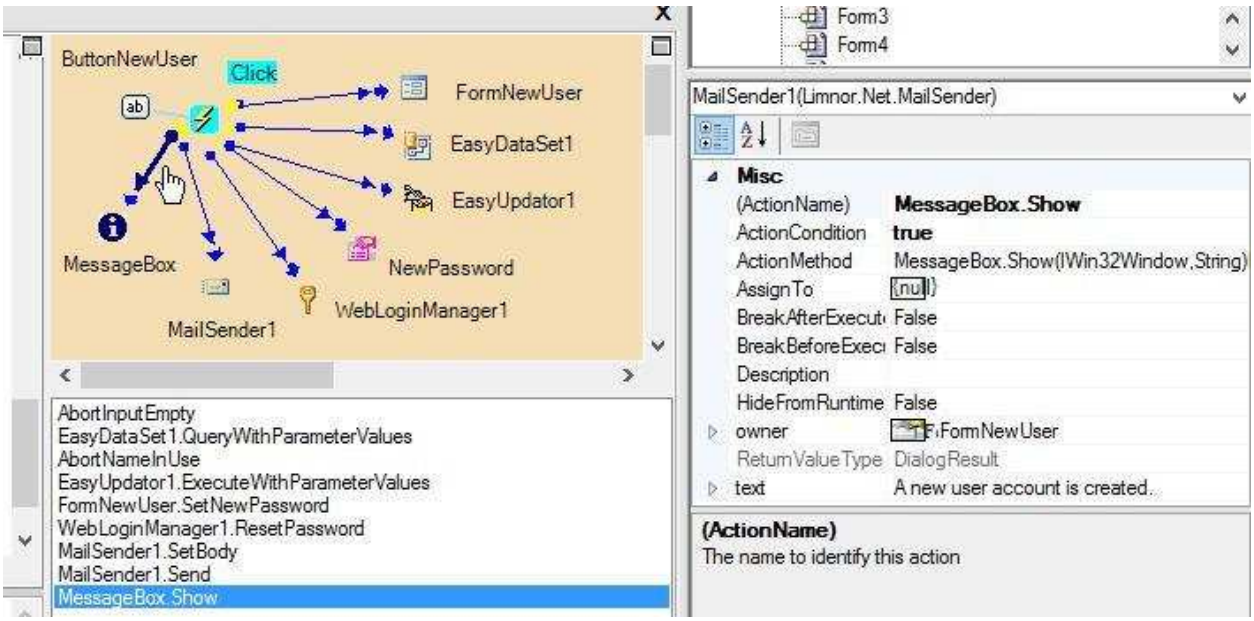
Select the current form as the message box owner:



Give a message and click OK:

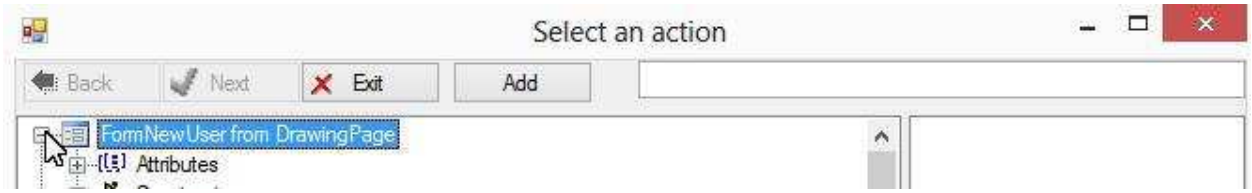


The action is created and assigned to the button:

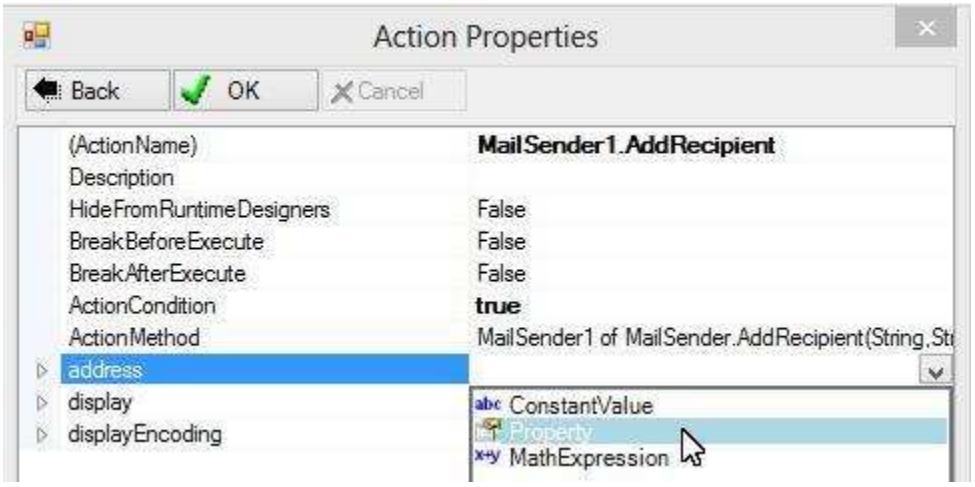


Set email address

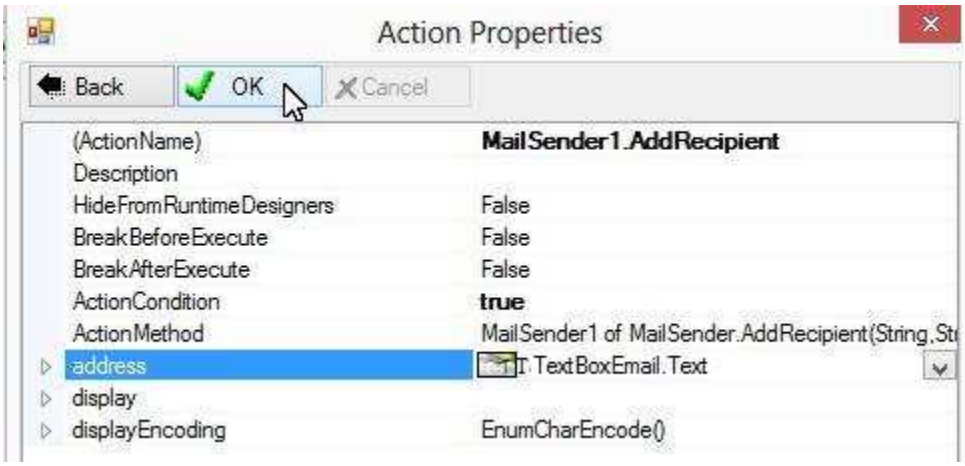
A reviewer spotted a bug in the above programming: email address is not used in sending the email. Let's do it now.


Select an action
Back
Next
Exit
Add
FormNewUser from DrawingPage
Attributes

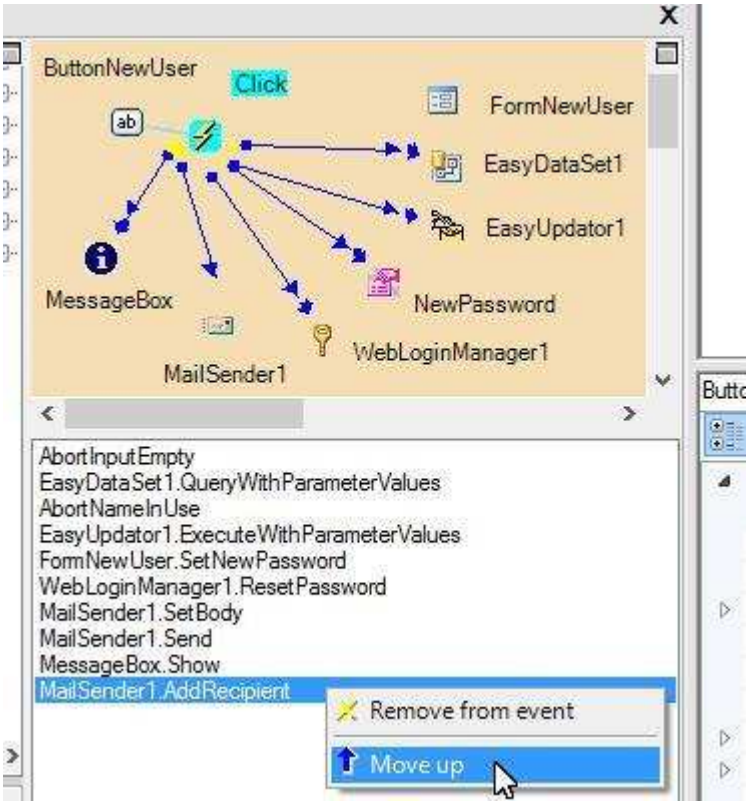

Select an action
Back
Next
Exit
Add
lblInfo of Label
MailSender1 of MailSender
Instance Members
Properties inherited
Methods inherited
AddAttachment(String)
AddBCCRecipient(String,String,EnumCharEncode)
AddCCRecipient(String,String,EnumCharEncode)
AddEmbededImage(String,String)
AddHeaderItem(String,String)
AddRecipient(String,String,EnumCharEncode)
ClearAttachments

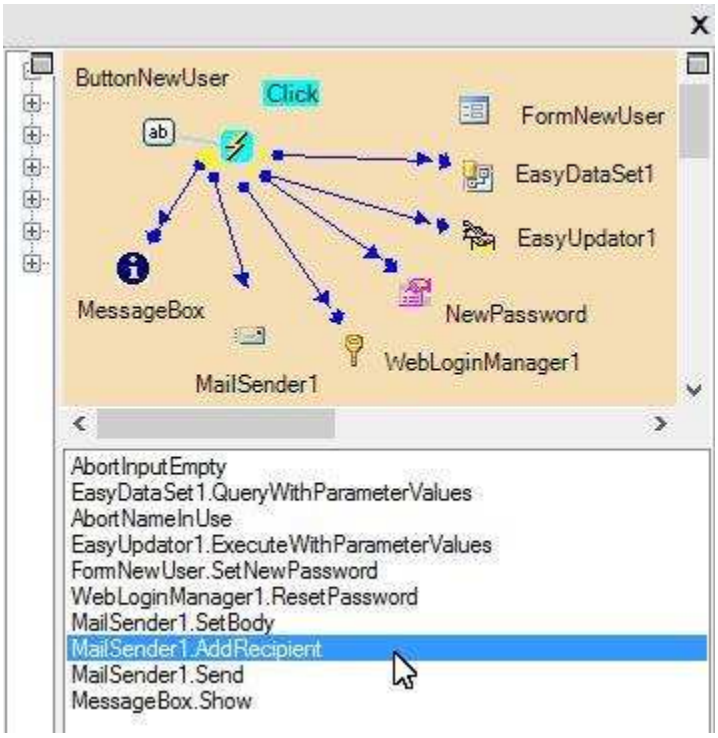

Action Properties
Back
OK
Cancel
(ActionName)
MailSender1.AddRecipient
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
MailSender1 of MailSender.AddRecipient(String,St
address
display
displayEncoding
ConstantValue
Property
MathExpression


Object picker
Back
Next
Exit
Add
FormNewUser from DrawingPage
Attributes
Constructors



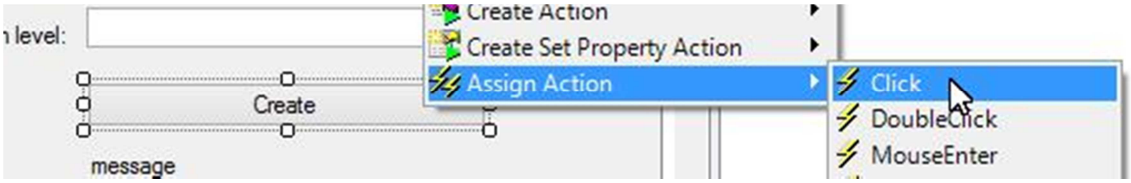
The action is created and assigned to the button. Move it up to before Send action:

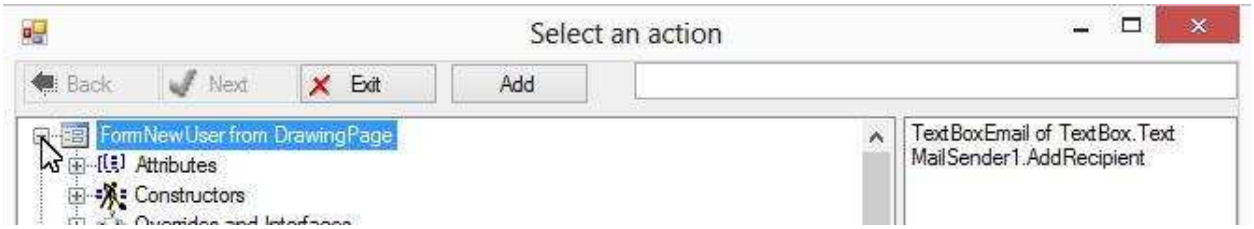


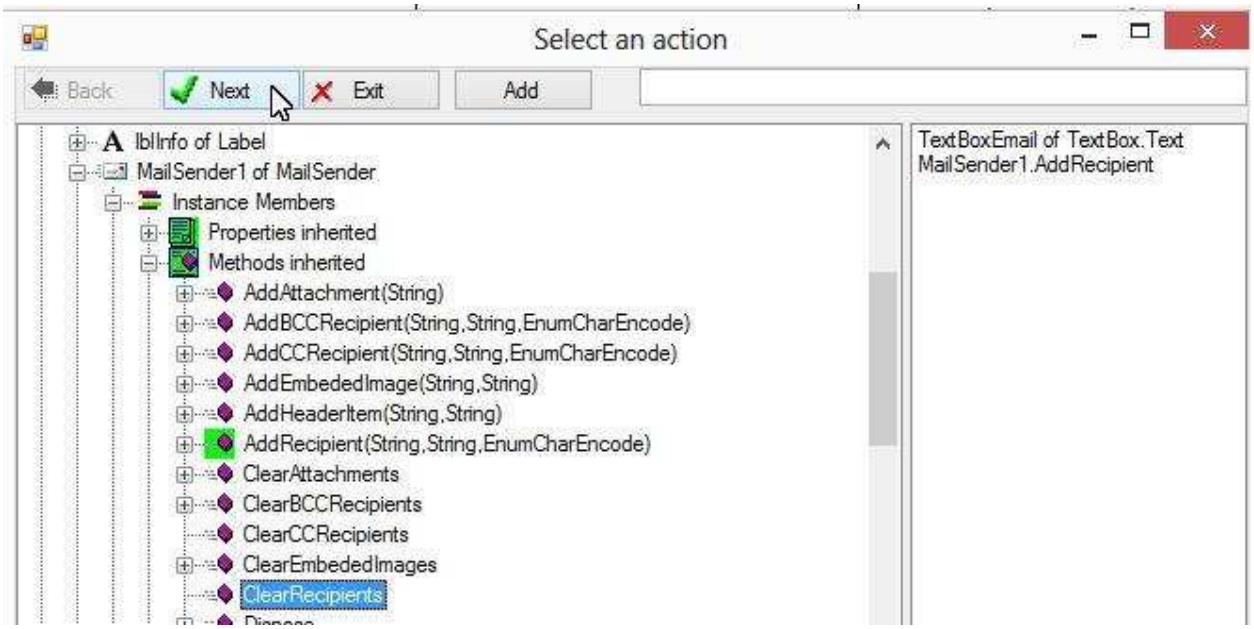


Remove email address

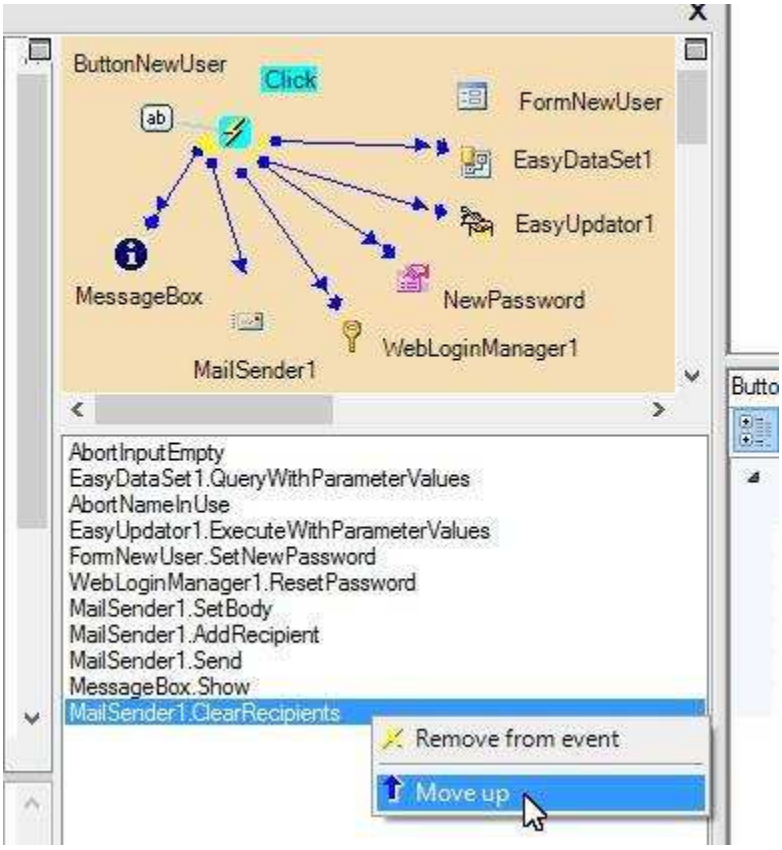
After sending email, the email address should be removed:

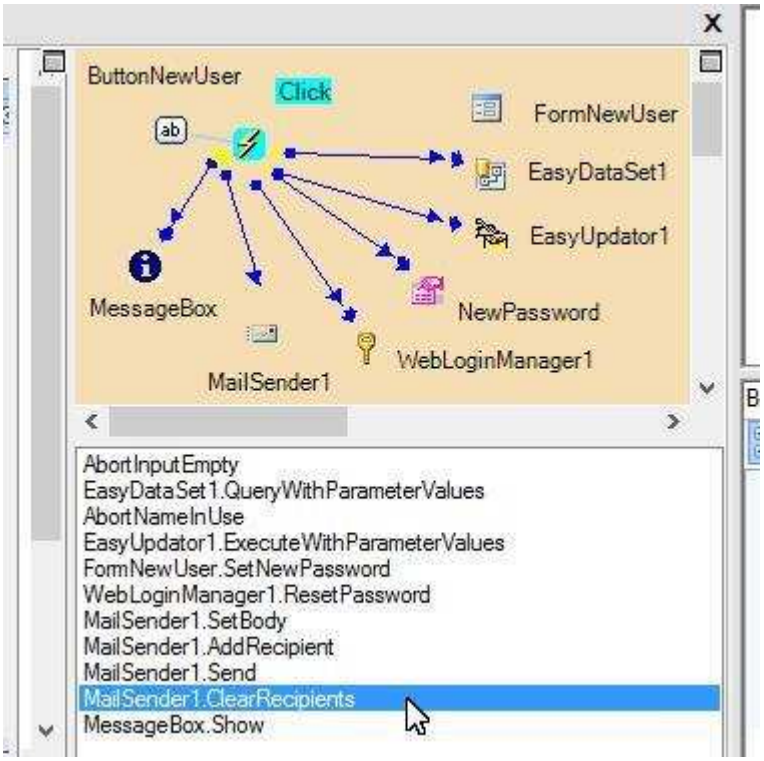






The action is created and assigned to the button. Move it up before the message box:

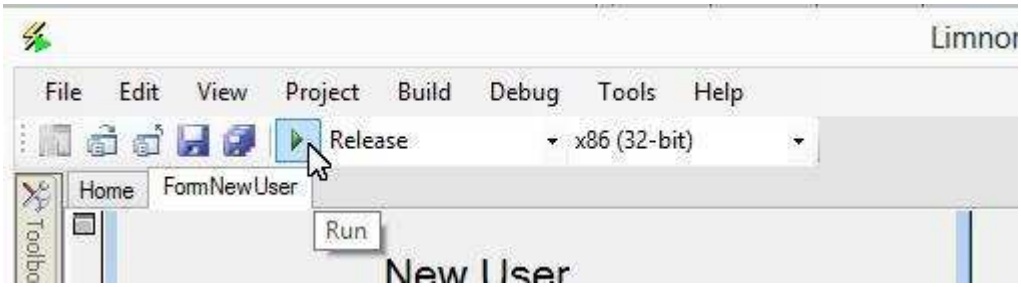




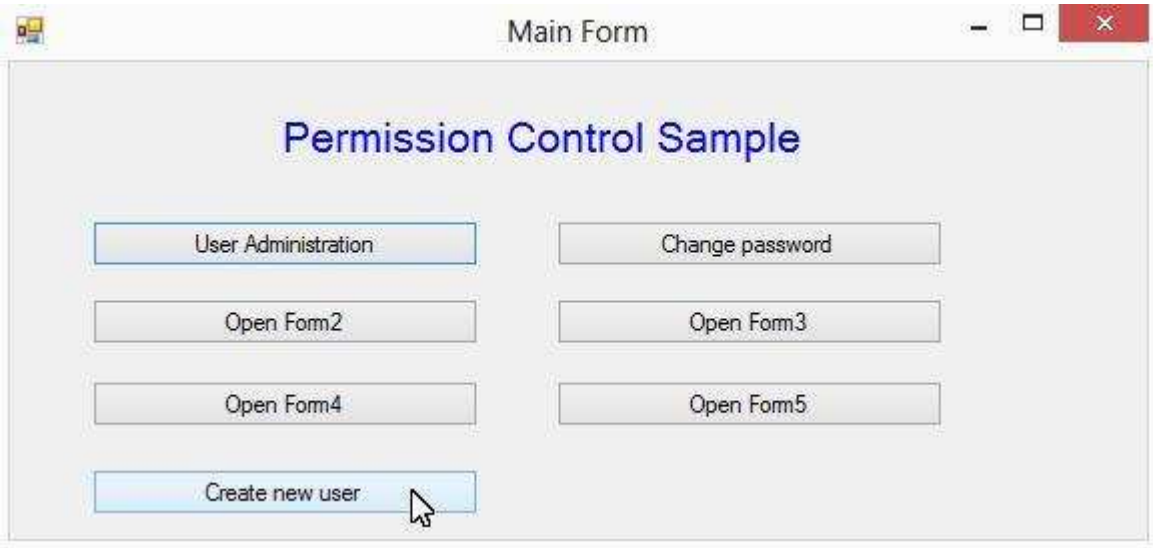
That is all the actions we need for creating a new user account.

Protect new user form

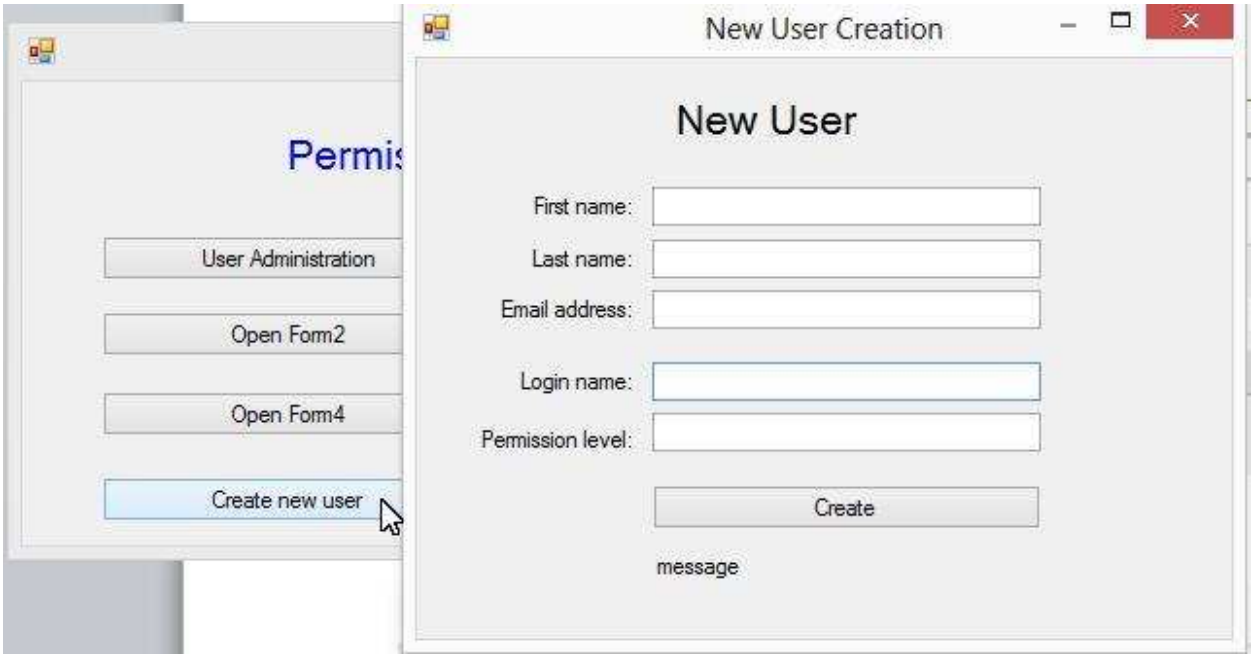
We want to add access restriction to this new user form. Before we do that, let's use it to create the first user record in the user table so that we may later log in using this first user to create other users. Run the project:



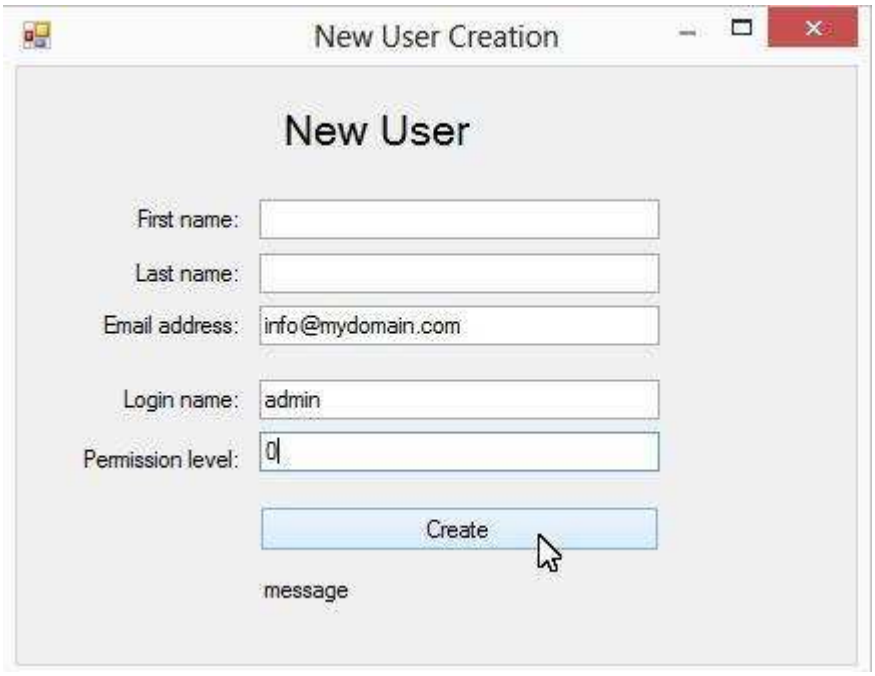
Click "Create new user":



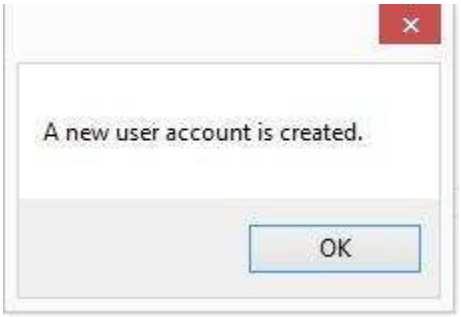
The new user form appears because we have not added access restriction to it yet:



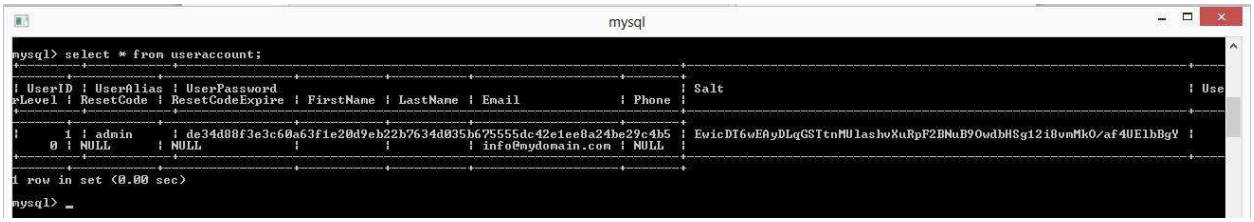
Enter new user data and click “Create” button:



A message box appears:

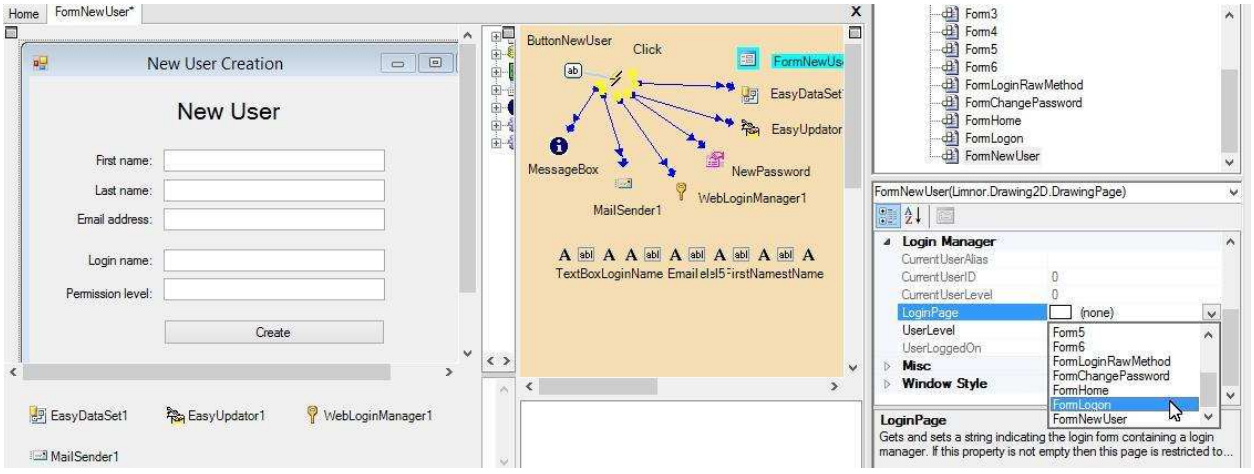


From the email message, we get the password for the new user: "Y9jjNsxGi+". The user table has its first record:

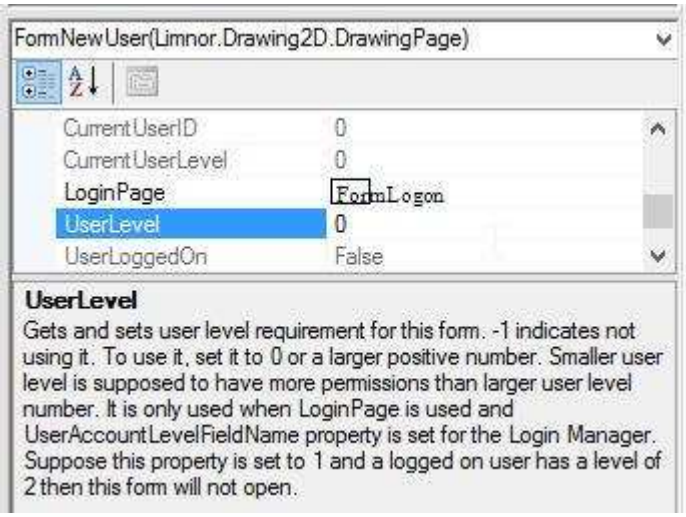


Note that the UserPassword field is de34d88f3e3c60a63f1e20d9eb22b7634d035b675555dc42e1ee8a24be29c4b5. It is the SHA256 hash of password "Y9jjNsxGi+" together with the Salt field. From string “de34d88f3e3c60a63f1e20d9eb22b7634d035b675555dc42e1ee8a24be29c4b5” it is not possible to figure out that the password is "Y9jjNsxGi+".

Now we may add access restriction to the new user form:



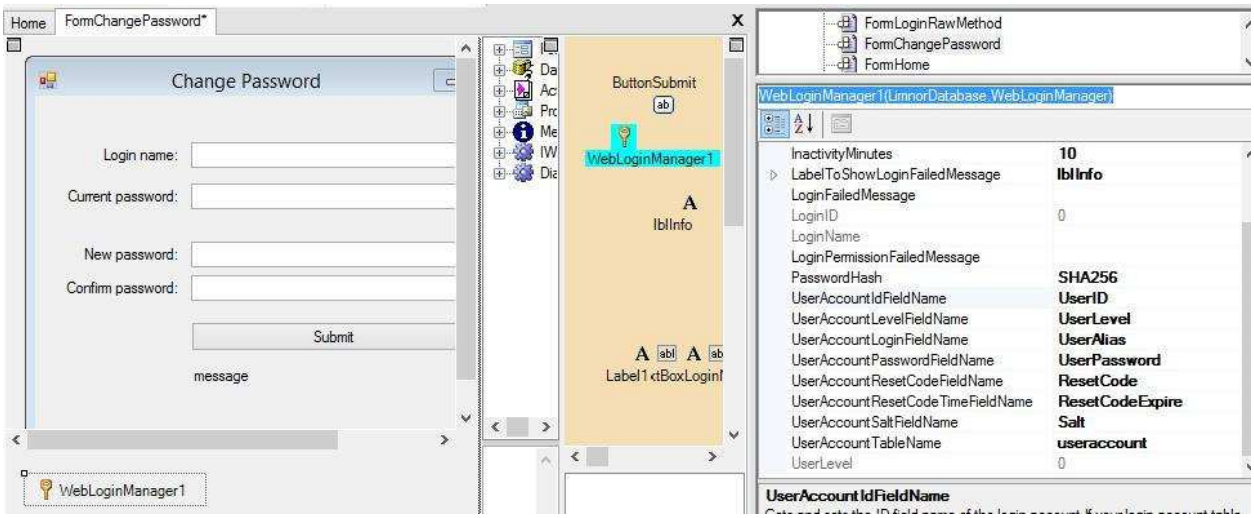
Keep the UserLevel to be 0 so that the maximum permission is required to use this form:



Compile and run the program again, clicking "Create new user" button, the login form will appear. Without a successful log in this new user form will not appear.

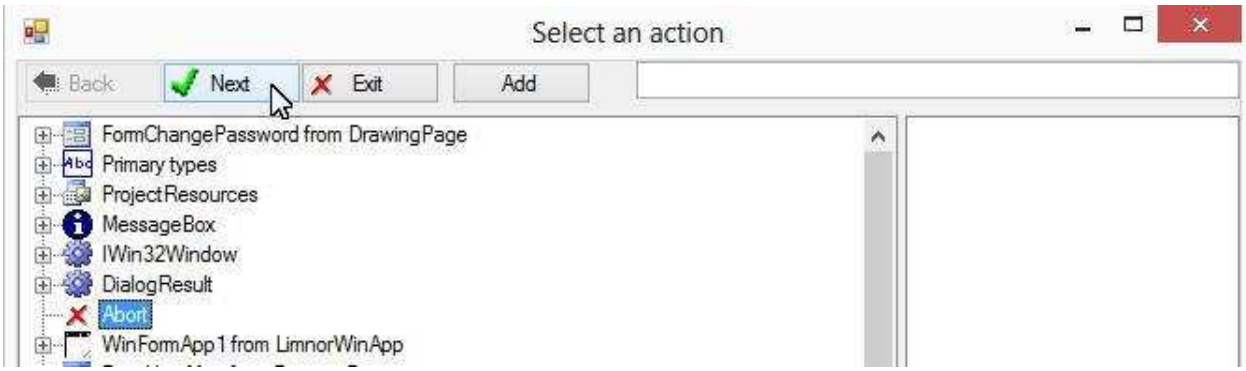
Change password

The ChangePassword method of the Login Manager Component can be used to allow a user to change his/her password. Let's use a form to give users this capability. Note that the properties of the Login Manager Component must be set the same as we did for the Login Manager component in the login form.

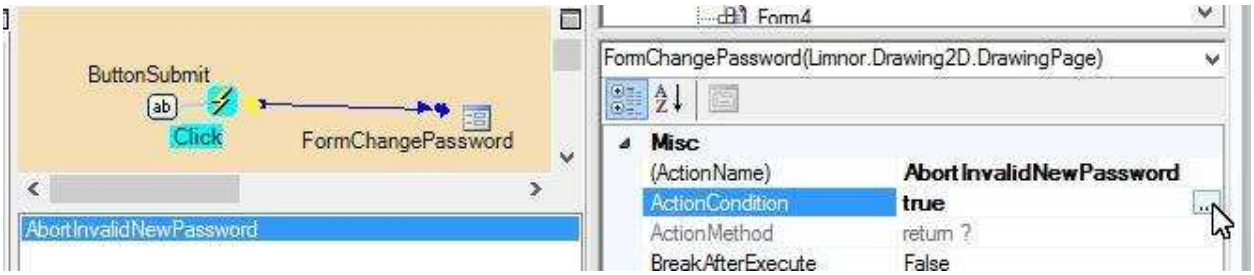


Input verification

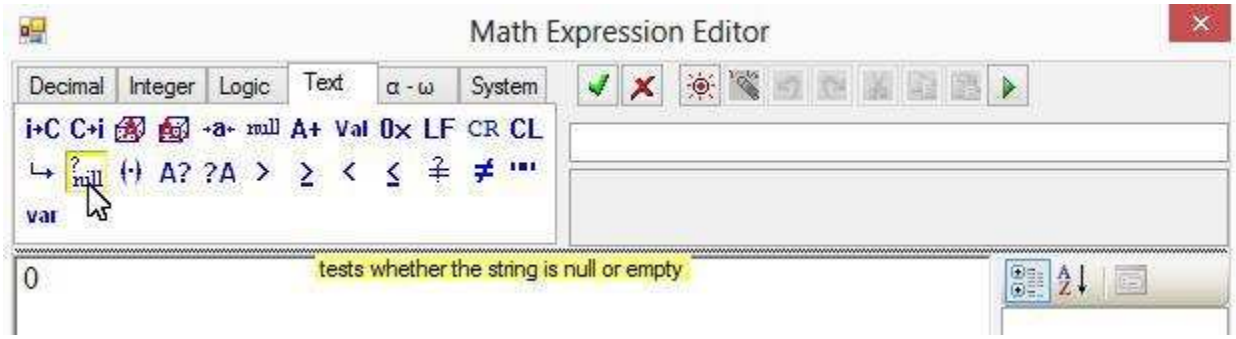
New password cannot be empty and must match confirm password. We abort the event handling if verification fails:



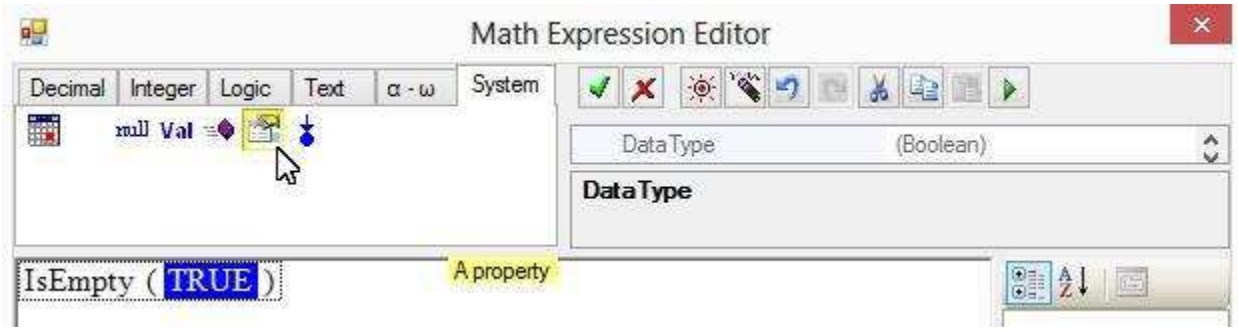
Rename the action and set the ActionCondition:



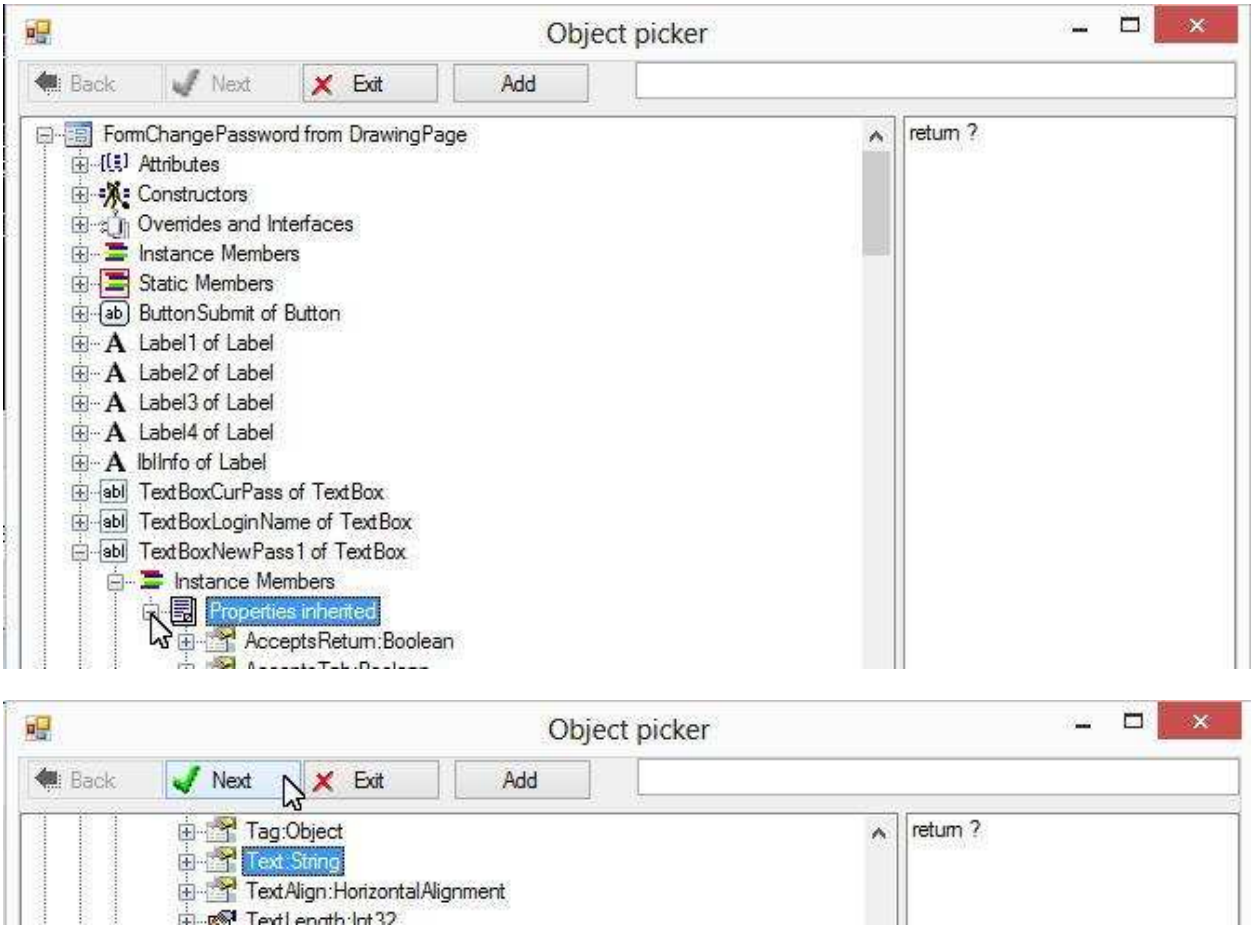
Click “?null”:



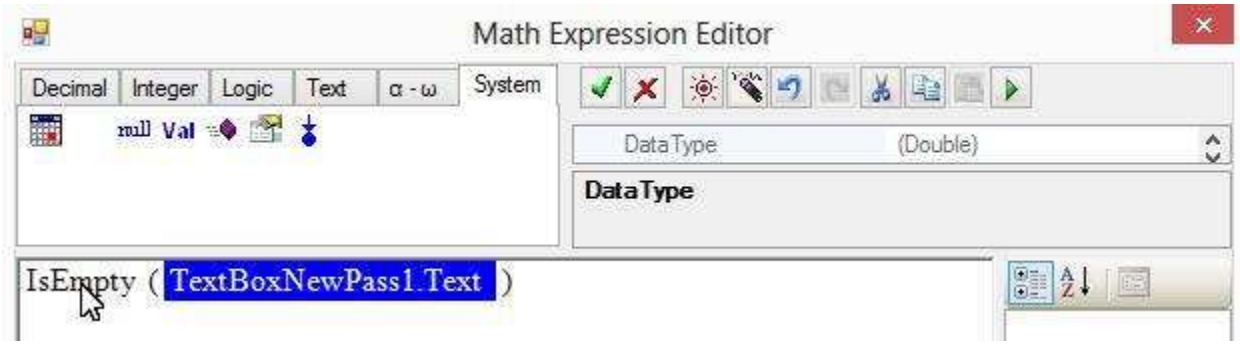
Click “TRUE” inside IsEmpty and click the Property icon:



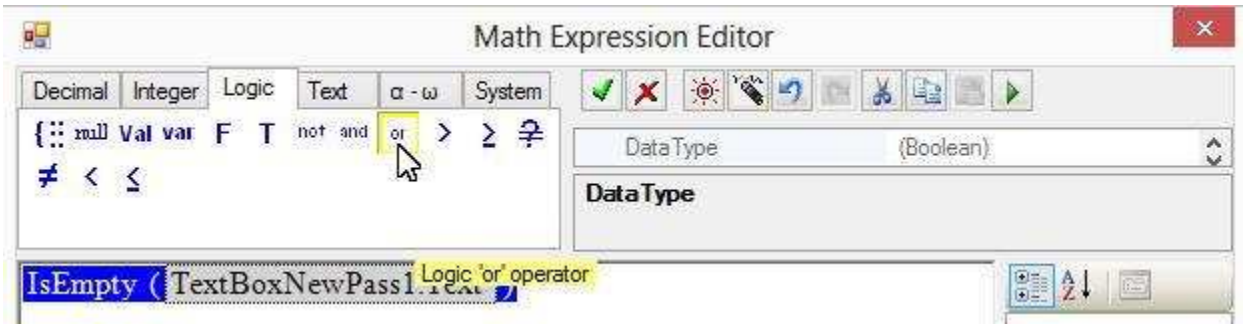
Select the Text property of the new password text box:



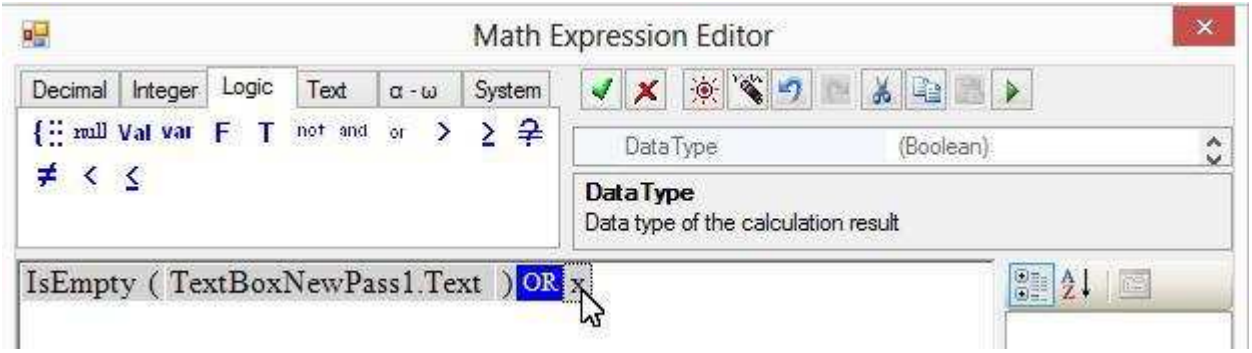
Click IsEmpty:



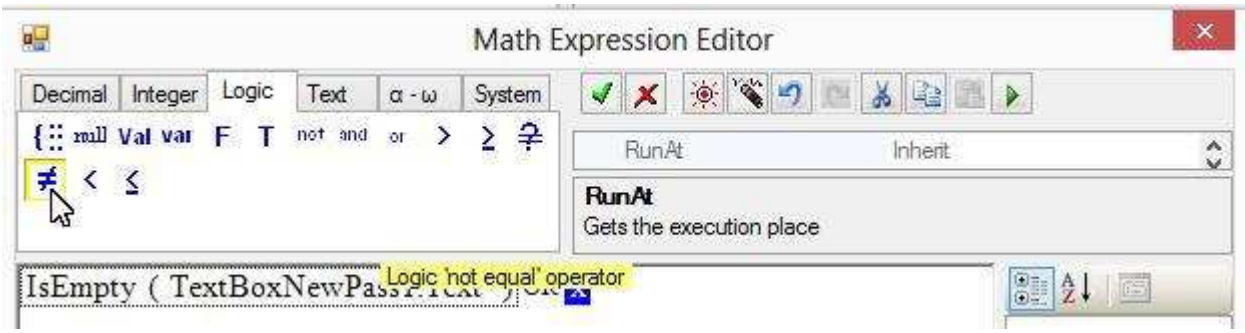
Click or:



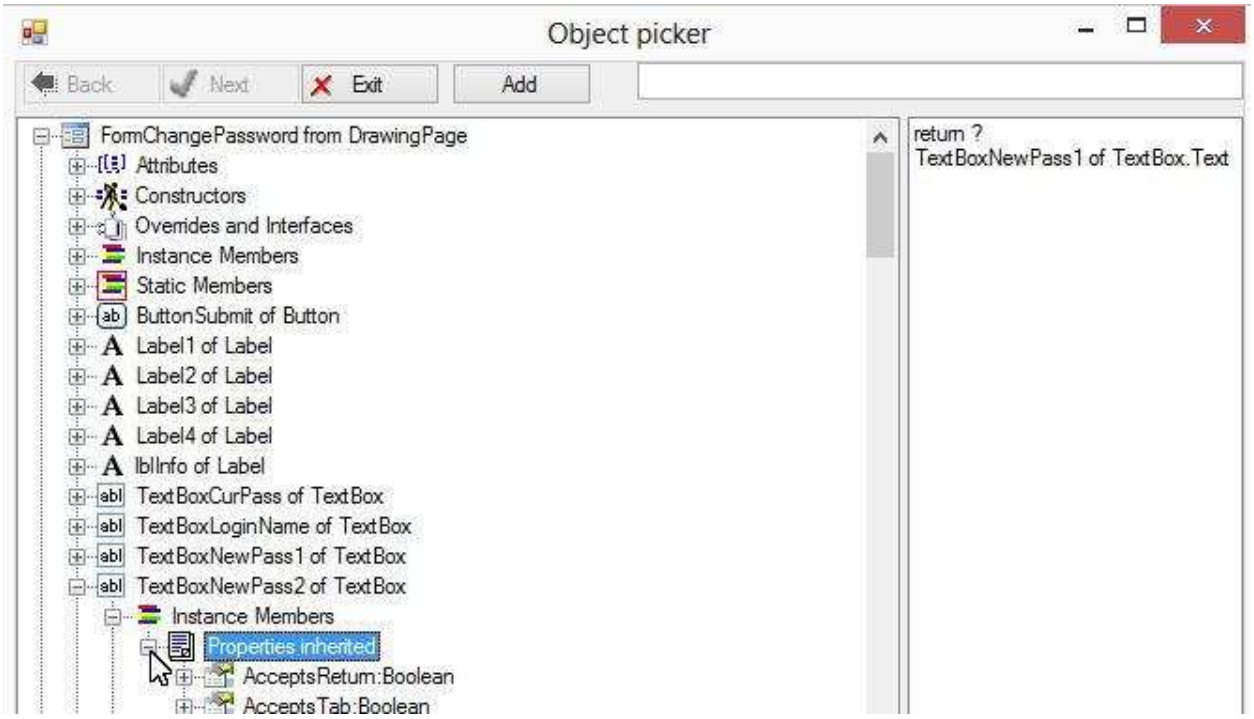
Click x:

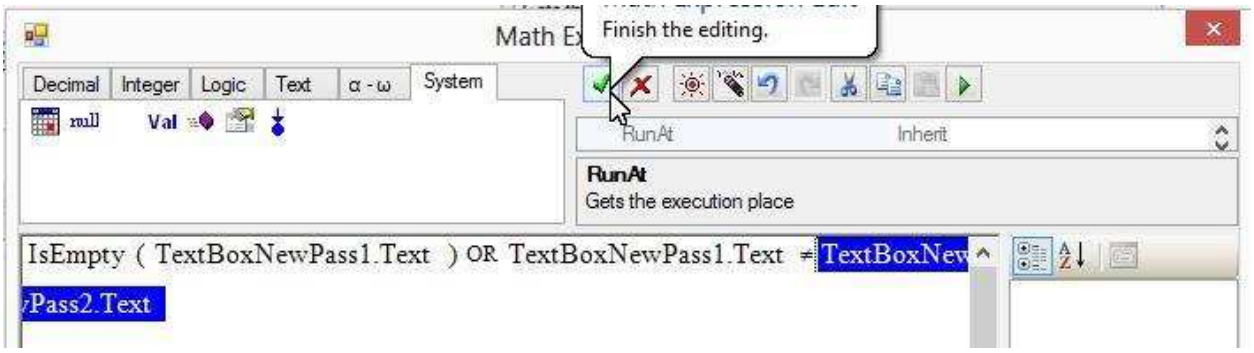


Click "=/=":

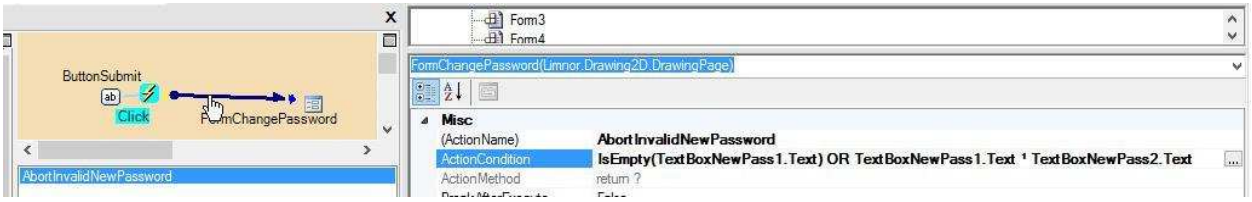


Select the Text property of the text boxes for the new password and confirm password on the two sides of "=/=":



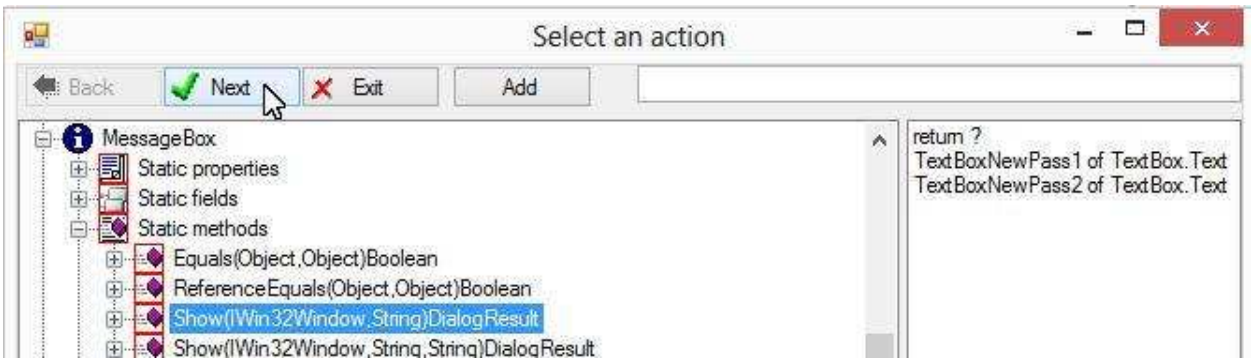


This is our input verification:

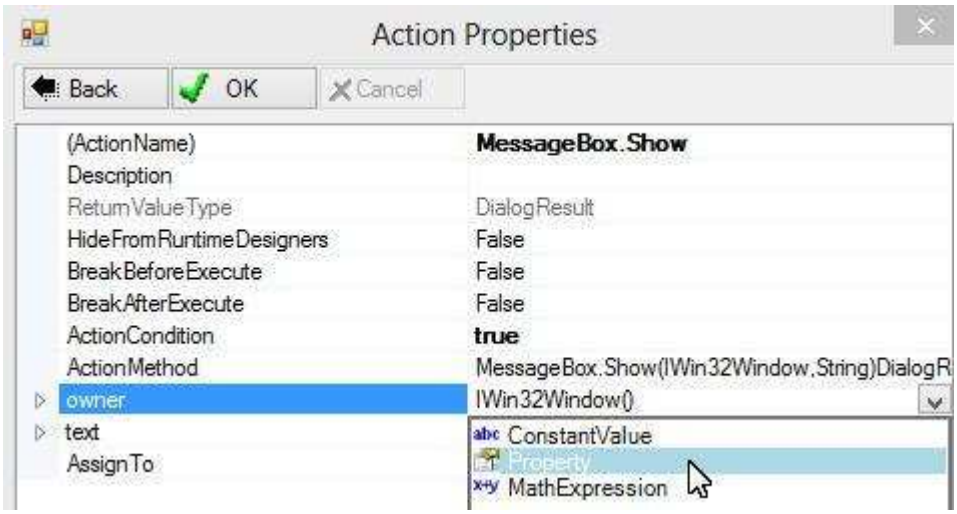


We may show a message box on verification failure:

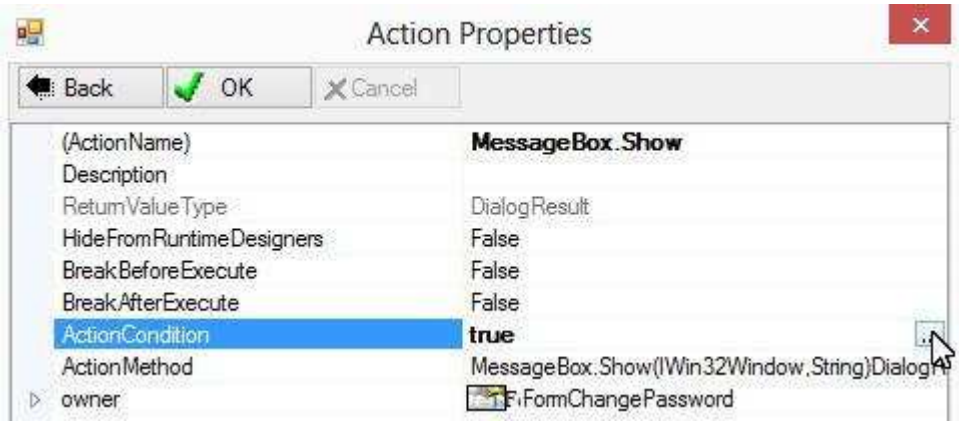
Select a Show method of the MessageBox:

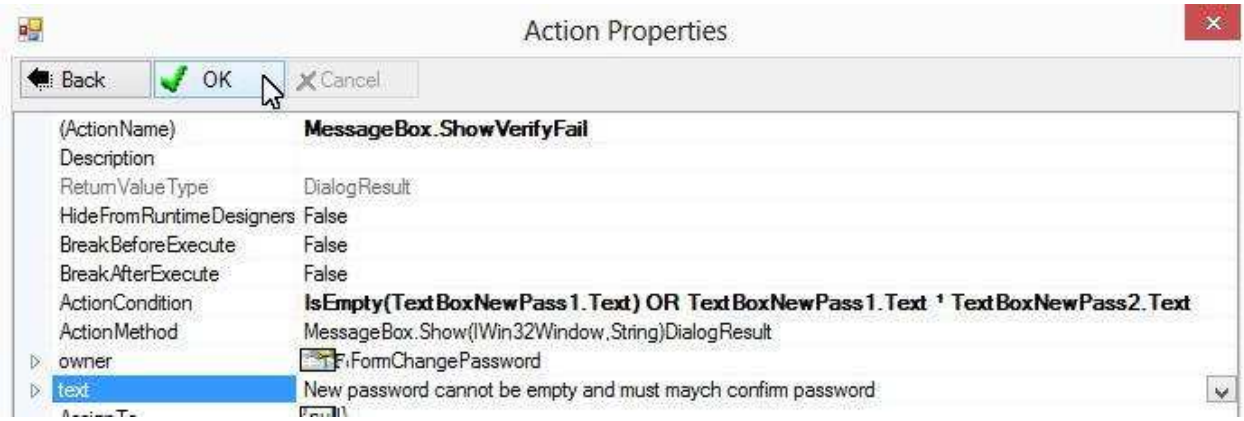


Use the current form as the owner of the message box:

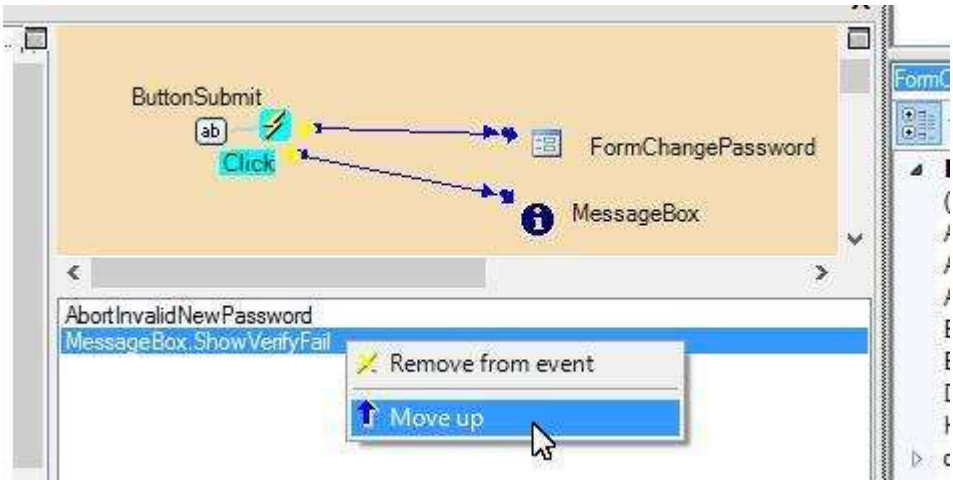


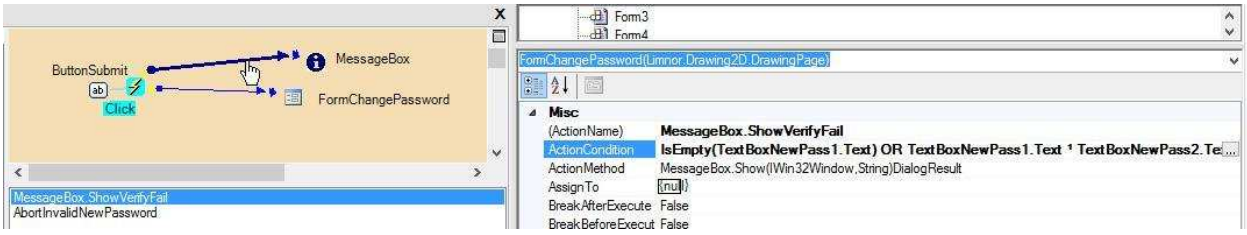
Set its ActionCondition to the same condition as we did before:





The action is created and assigned to the button. Move it up:

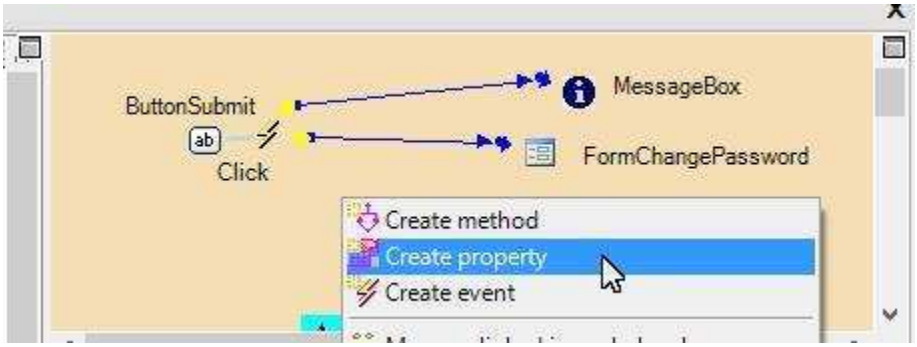




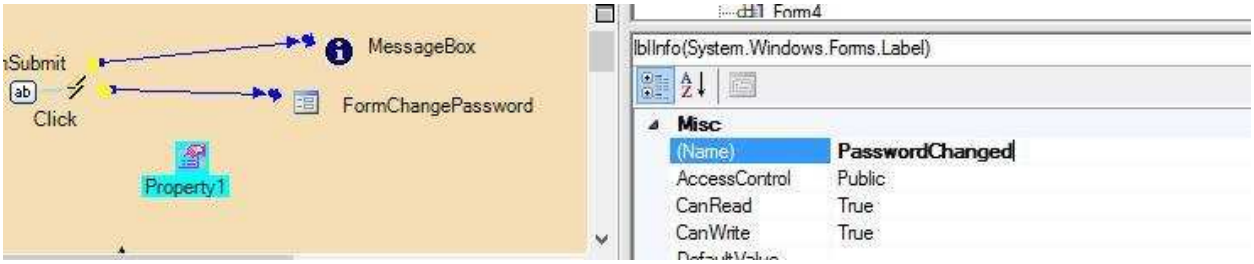
When verification fail, a message box appears, then the event handling aborts.

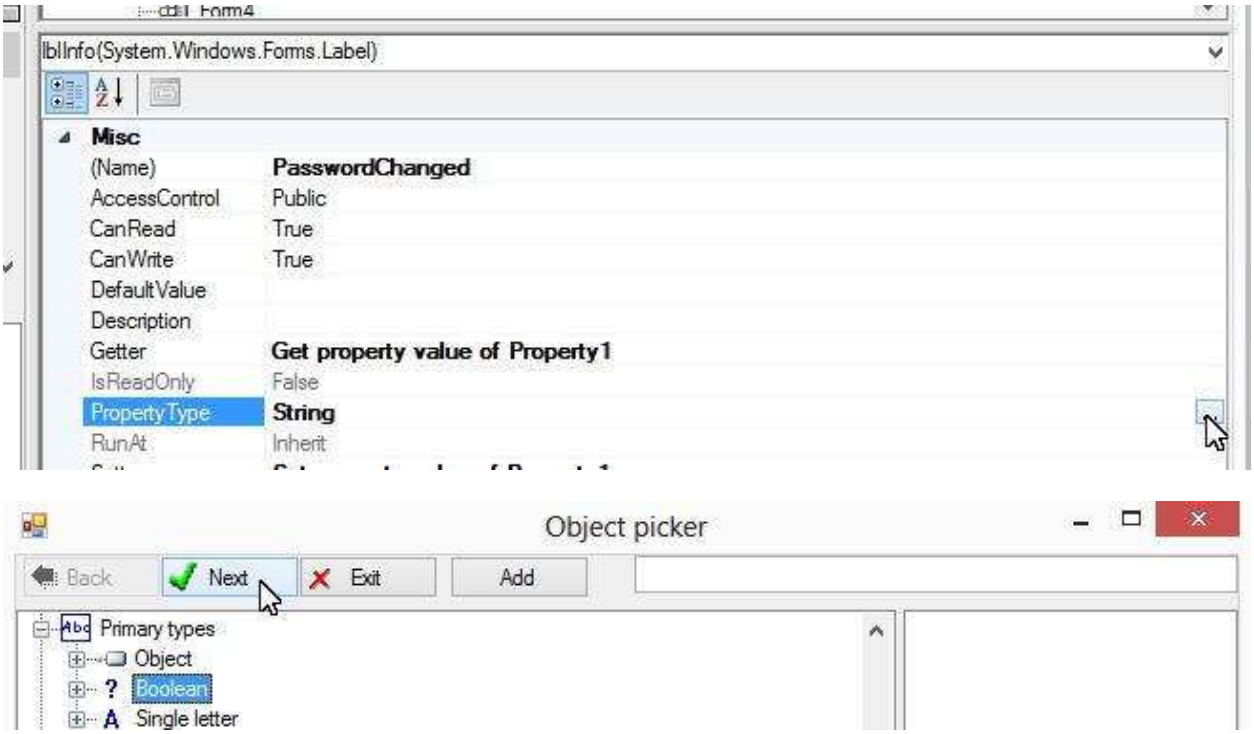
Execute password change

We use a property to indicate whether password changing is successful:

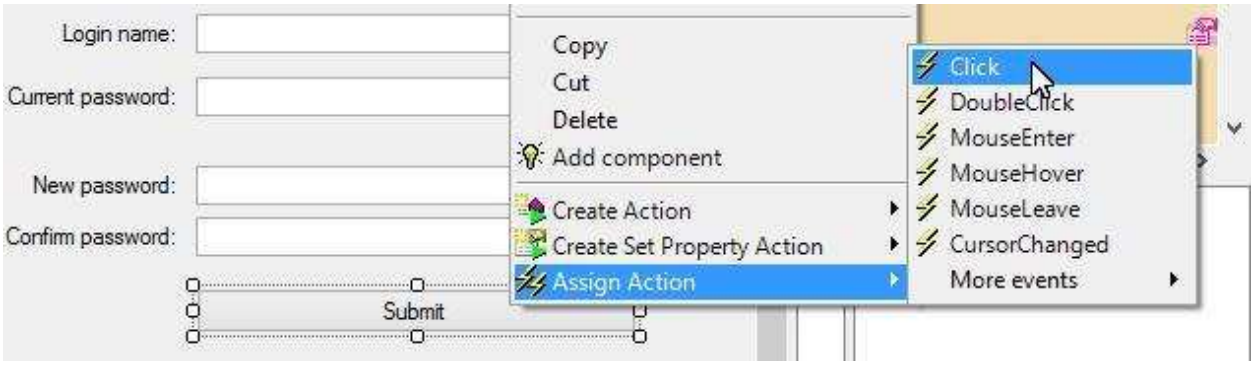


Rename the property and change its type to be Boolean:

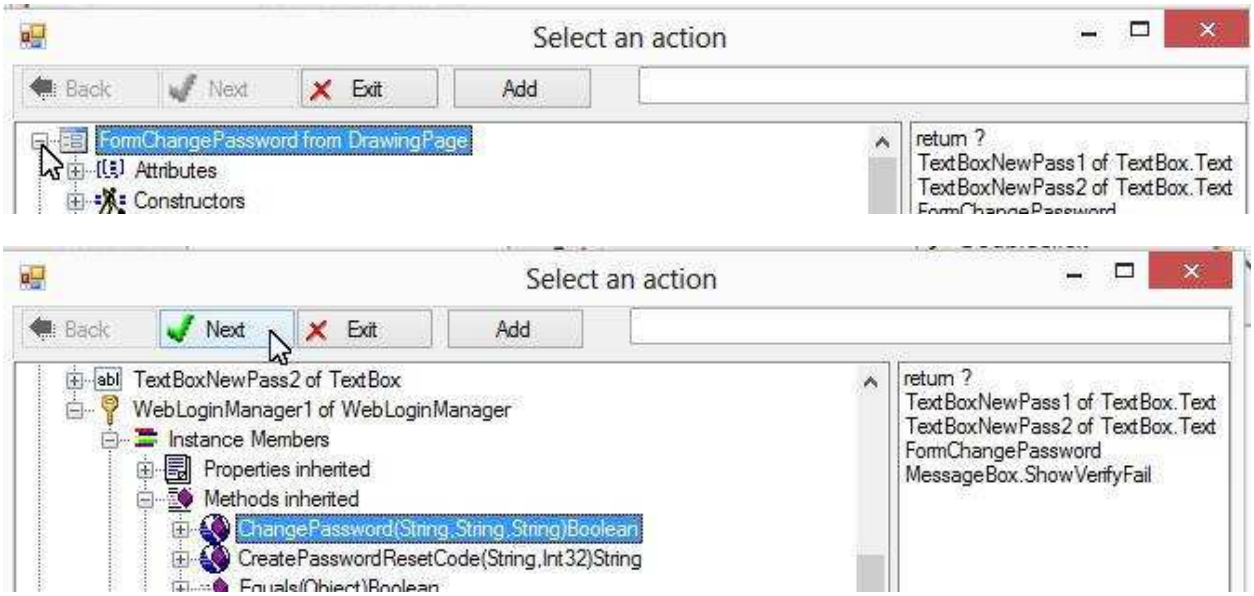




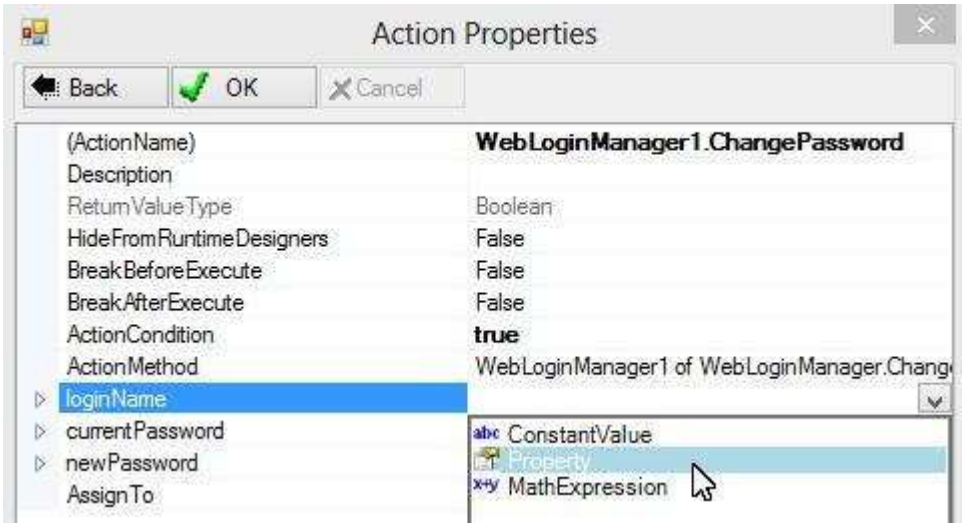
Now we may execute a ChangePassword action:



Select ChangePassword method:



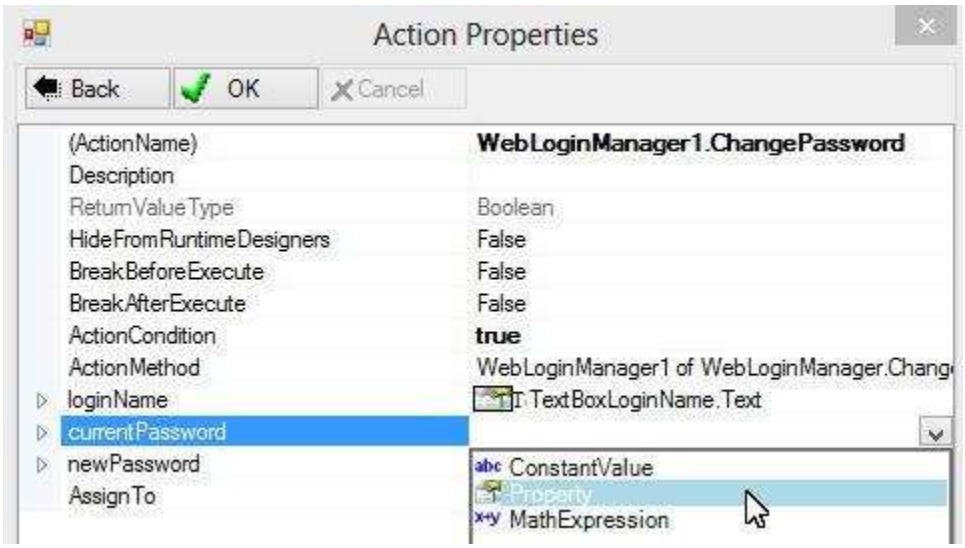
Pass Text property of the text boxes to the action:

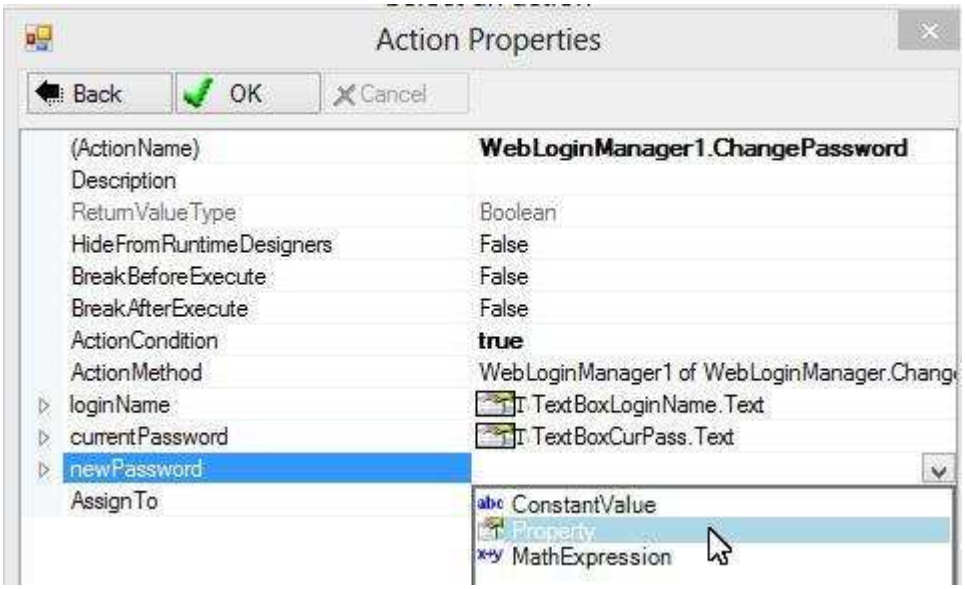



Object picker
Back
Next
Exit
Add
FormChangePassword from DrawingPage
Attributes
Constructors
return ?
TextBoxNewPass1 of TextBox.Text
TextBoxNewPass2 of TextBox.Text

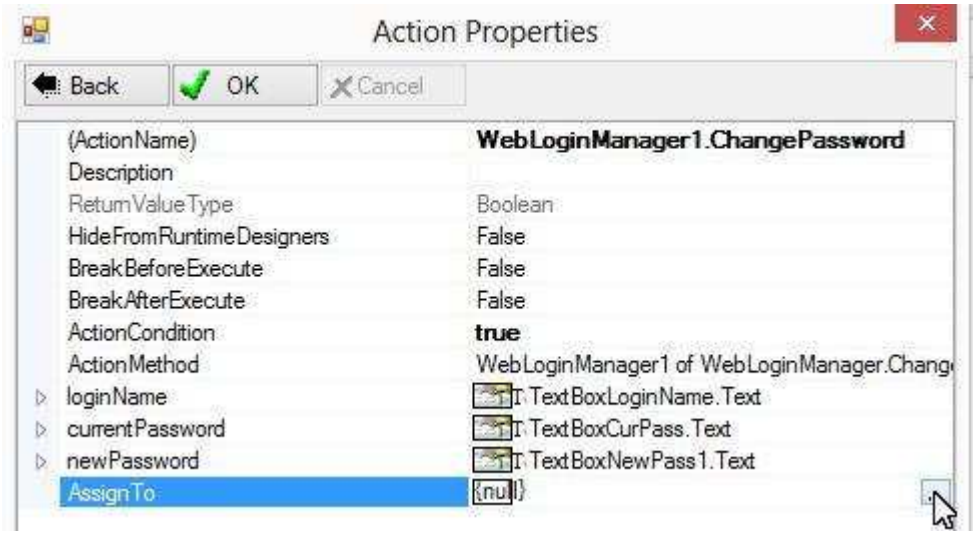

Object picker
Back
Next
Exit
Add
TextBoxCurPass of TextBox
TextBoxLoginName of TextBox
Instance Members
Properties inherited
AcceptsReturn:Boolean
return ?
TextBoxNewPass1 of TextBox.Text
TextBoxNewPass2 of TextBox.Text
FormChangePassword
MessageBox.ShowVerifyFail

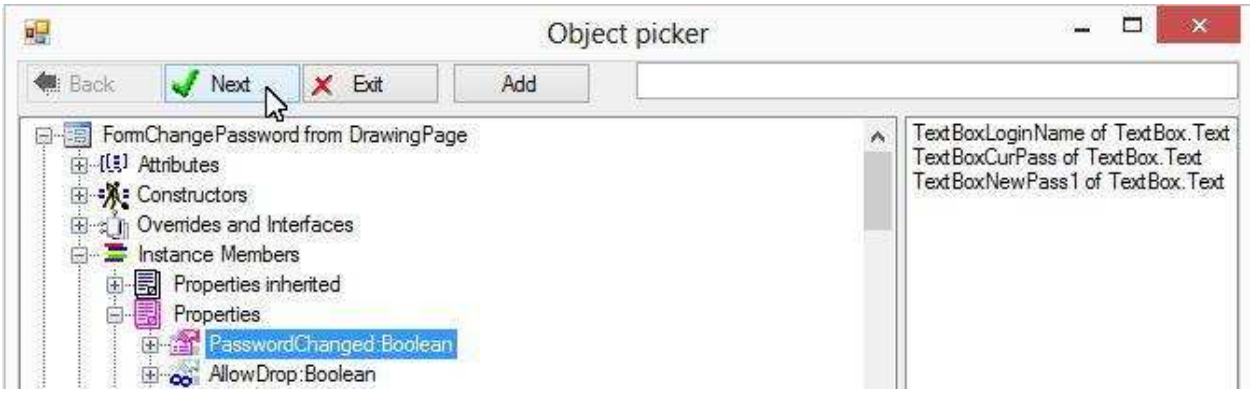

Object picker
Back
Next
Exit
Add
TabStop:Boolean
Tag:Object
Text:String
TextAlign:HorizontalAlignment
return ?
TextBoxNewPass1 of TextBox.Text
TextBoxNewPass2 of TextBox.Text
FormChangePassword
MessageBox.ShowVerifyFail

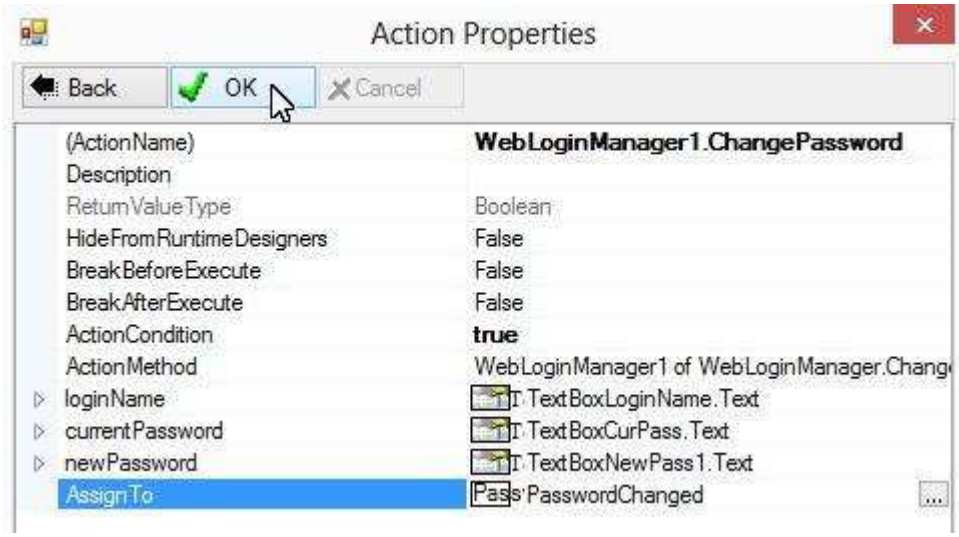

Action Properties
Back
OK
Cancel
(ActionName)
WebLoginManager1.ChangePassword
Description
ReturnValueType
Boolean
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
WebLoginManager1 of WebLoginManager.Chang
loginName
TextBoxLoginName.Text
currentPassword
newPassword
AssignTo
ConstantValue
Property
MathExpression



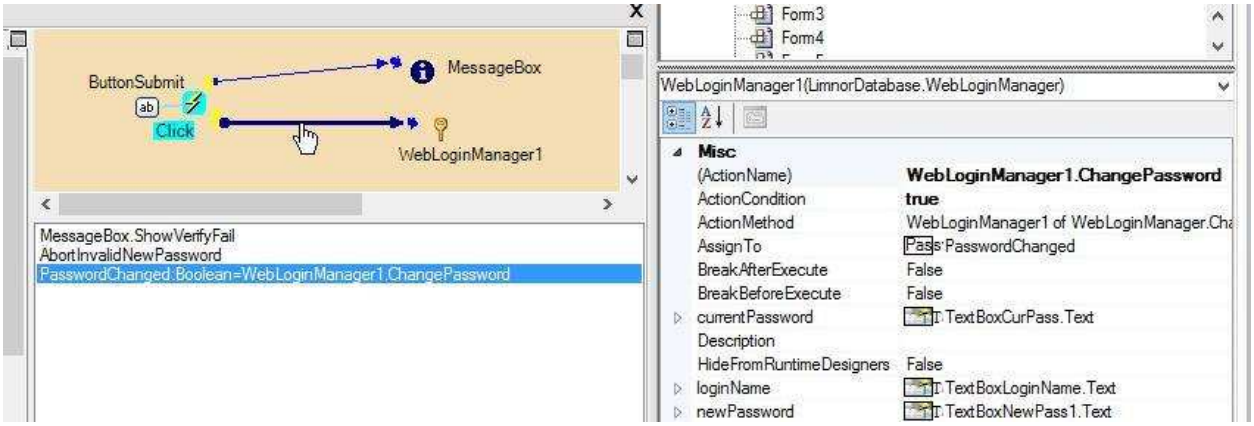
Set “AssignTo” to PasswordChanged property:



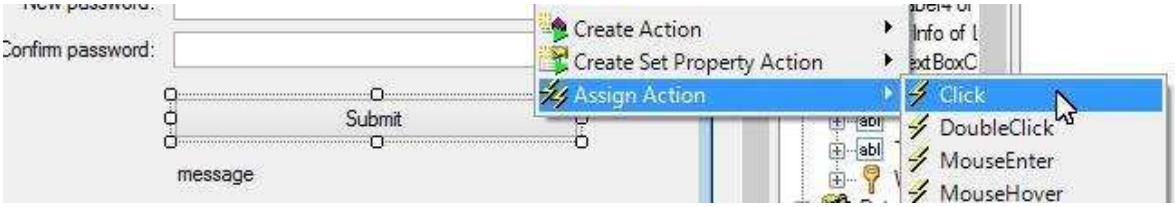


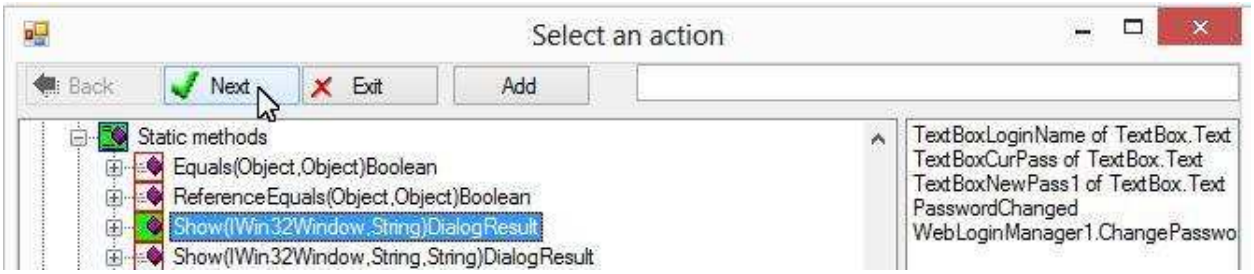


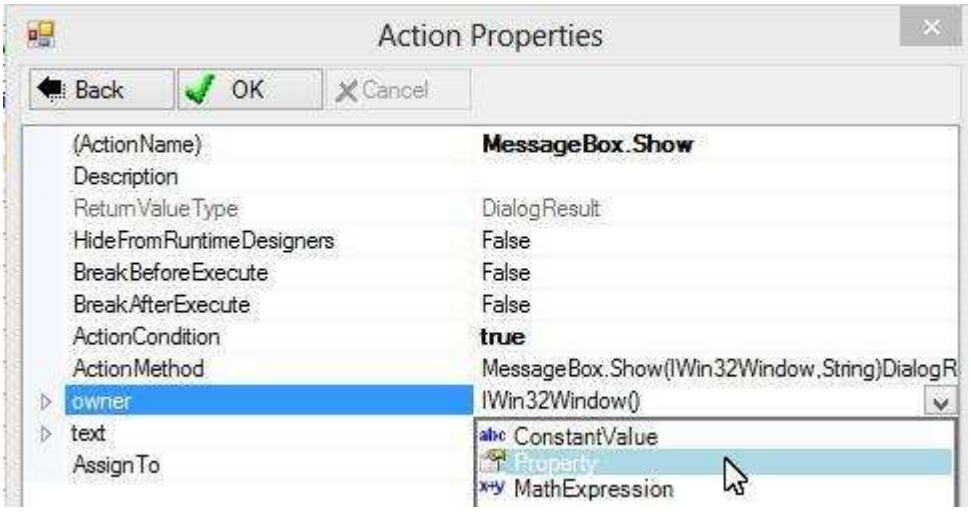
The action is created and assigned to the button:



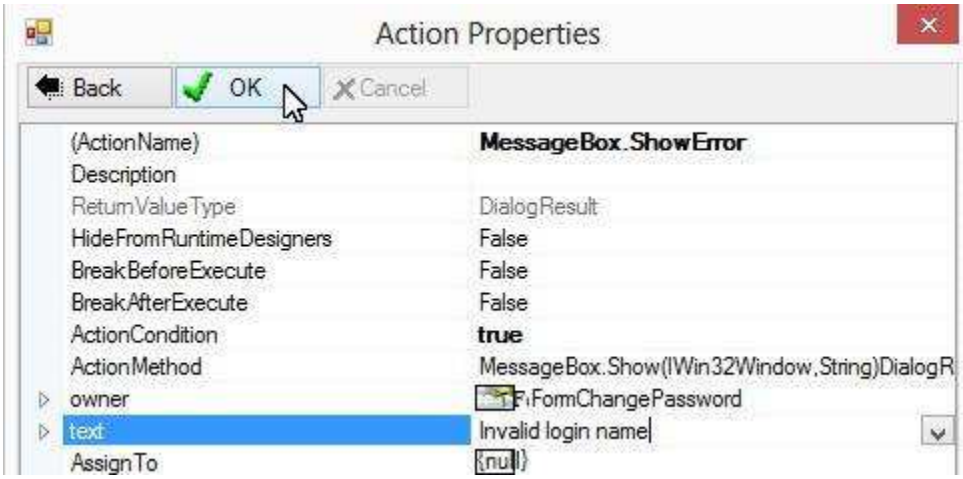
Show error message

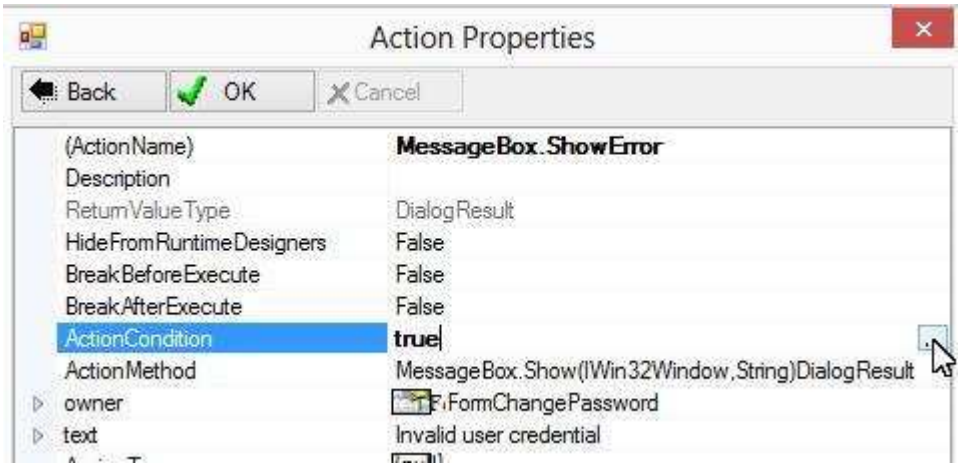


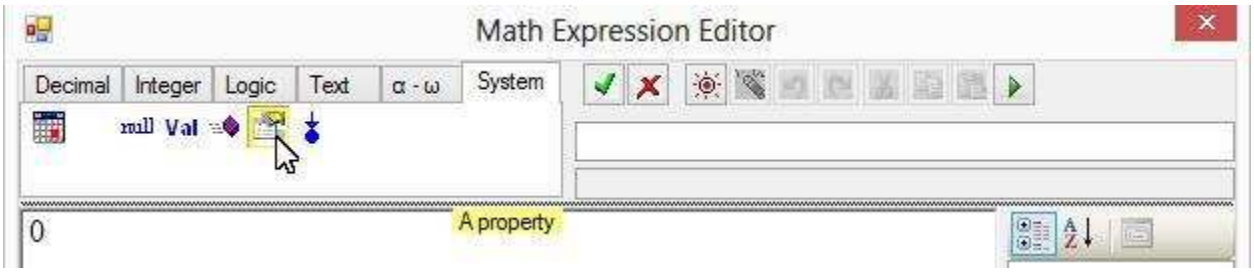


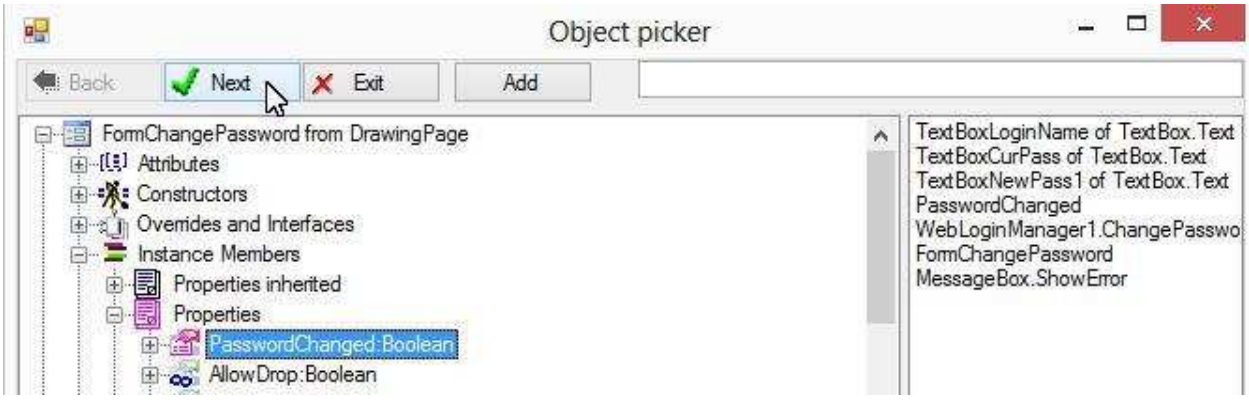


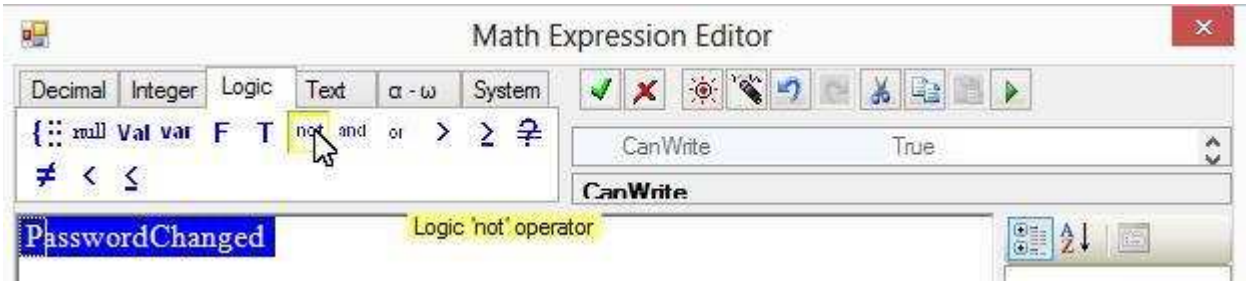

Object picker
Back
Next
Exit
Add
FormChangePassword from DrawingPage
Attributes
TextBoxLoginName of TextBox.Text
TextBoxCurPass of TextBox.Text
TextBoxNewPass1 of TextBox.Text

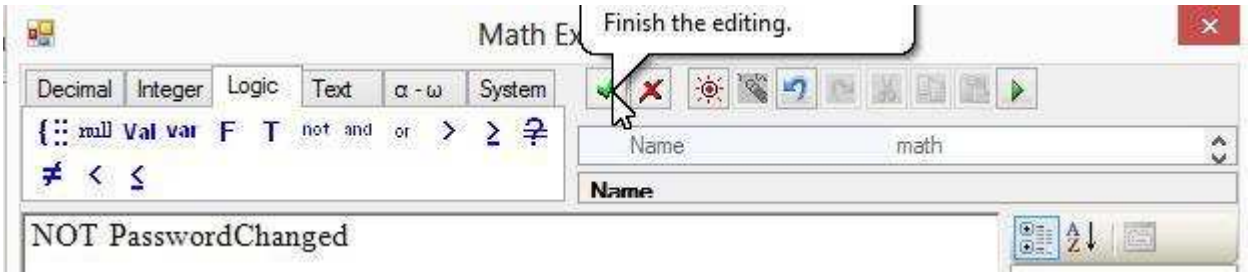

Action Properties
Back
OK
Cancel
(ActionName) MessageBox.ShowError
Description
ReturnValueType DialogResult
HideFromRuntimeDesigners False
BreakBeforeExecute False
BreakAfterExecute False
ActionCondition true
ActionMethod MessageBox.Show(IWin32Window,String)DialogR
owner FormChangePassword
text Invalid login name
AssignTo {null}

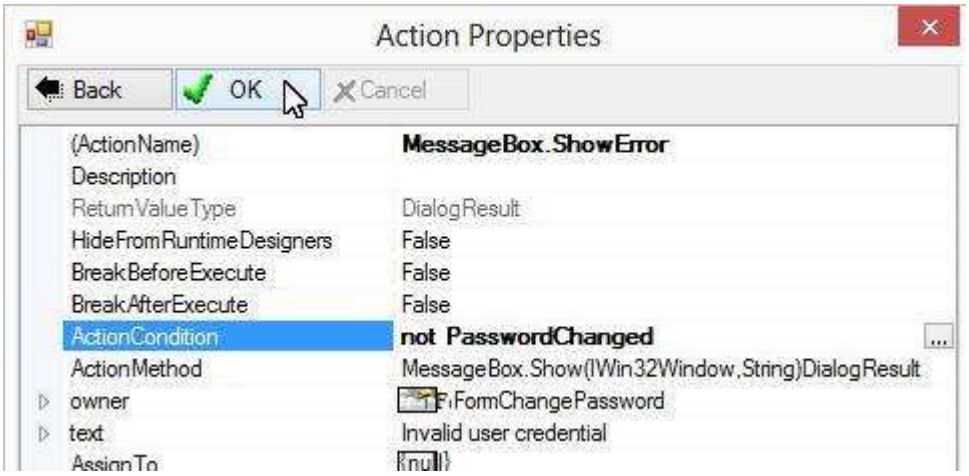

Action Properties
Back
OK
Cancel
(ActionName) MessageBox.ShowError
Description
ReturnValueType DialogResult
HideFromRuntimeDesigners False
BreakBeforeExecute False
BreakAfterExecute False
ActionCondition true
ActionMethod MessageBox.Show(IWin32Window,String)DialogResult
owner FormChangePassword
text Invalid user credential


Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
null
Val
0
A property

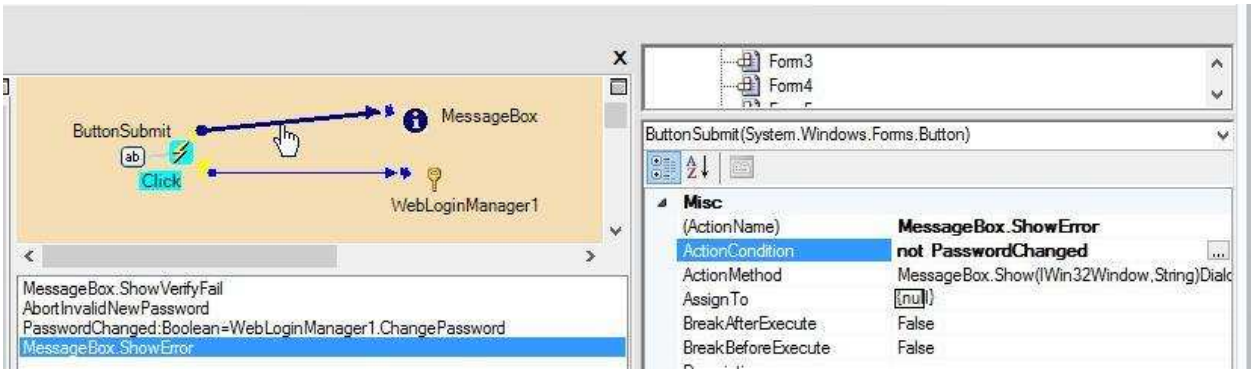




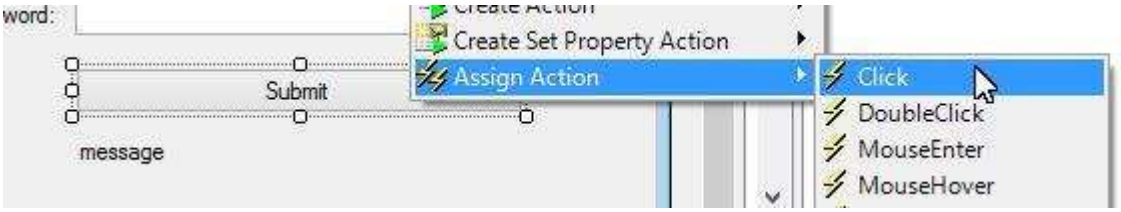


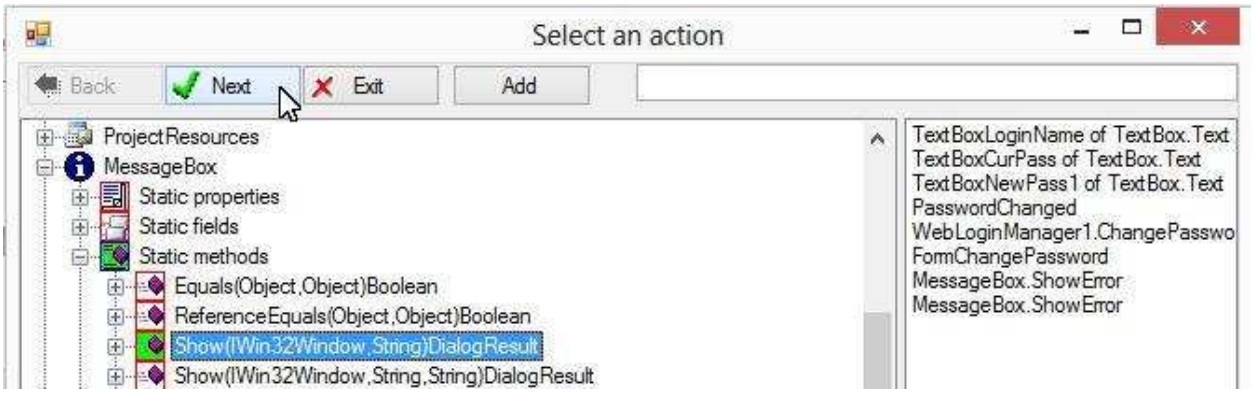


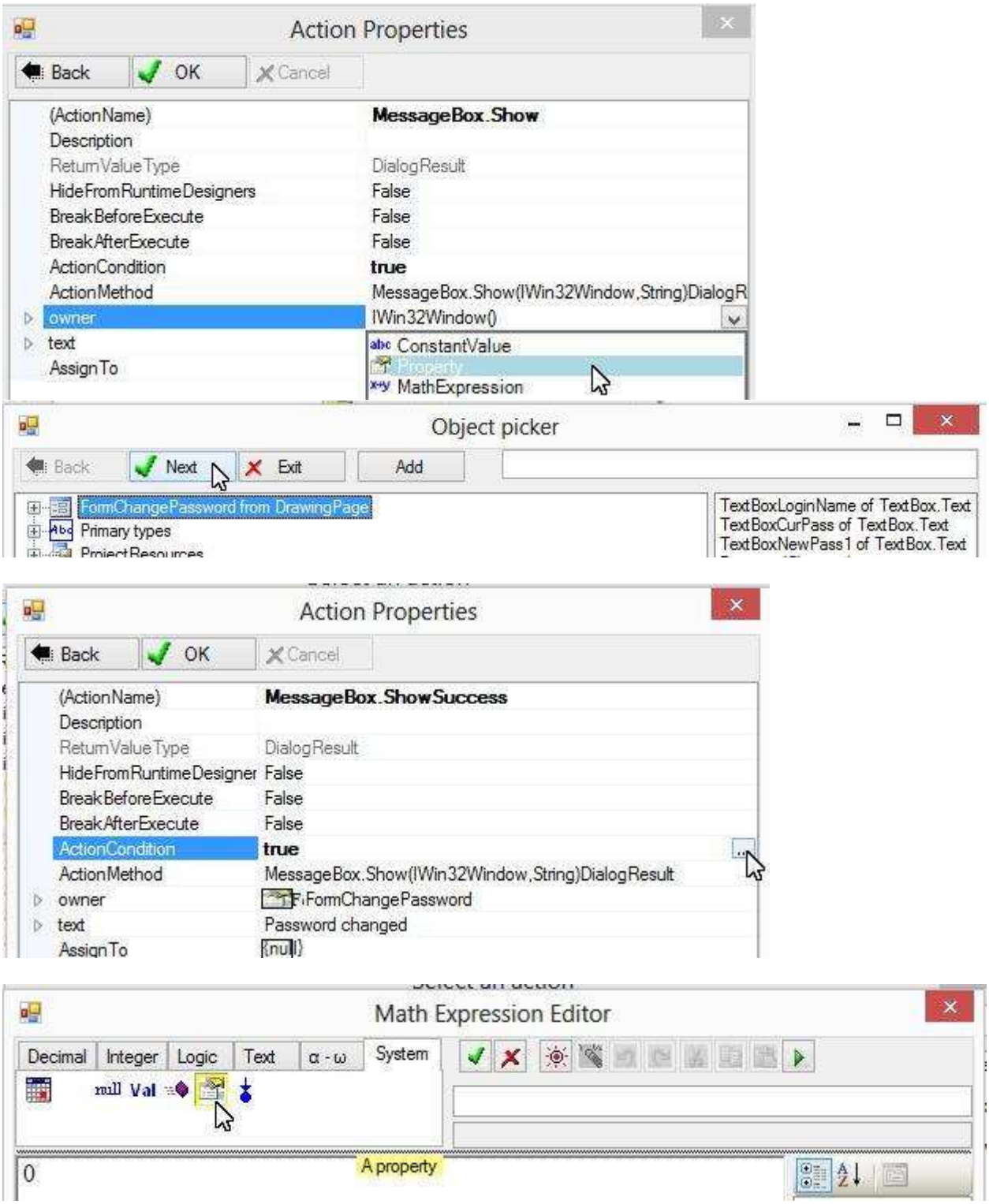
The action is created and assigned to the button:

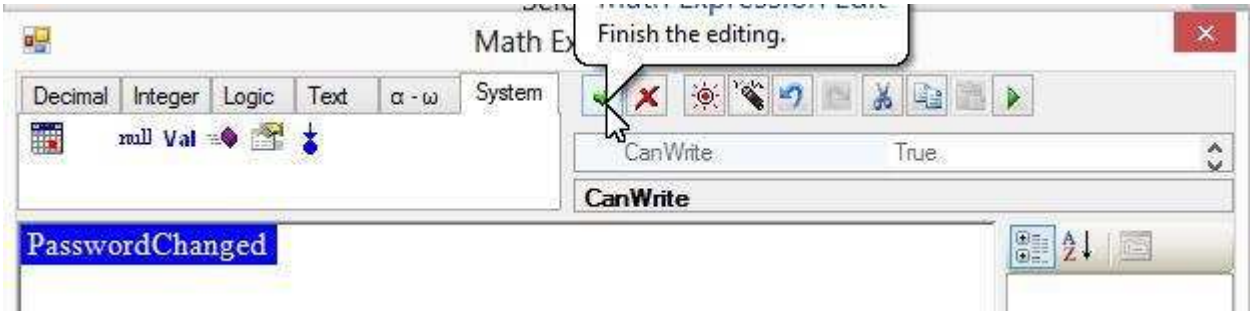


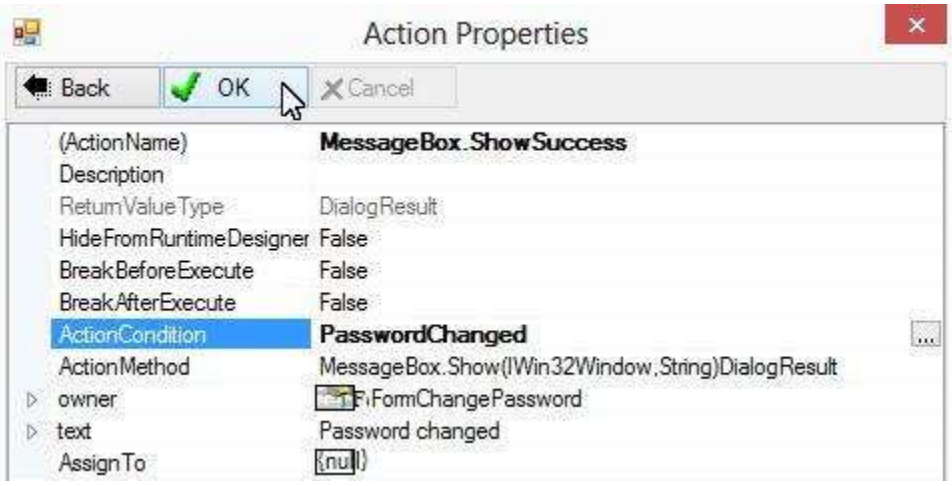
Show success



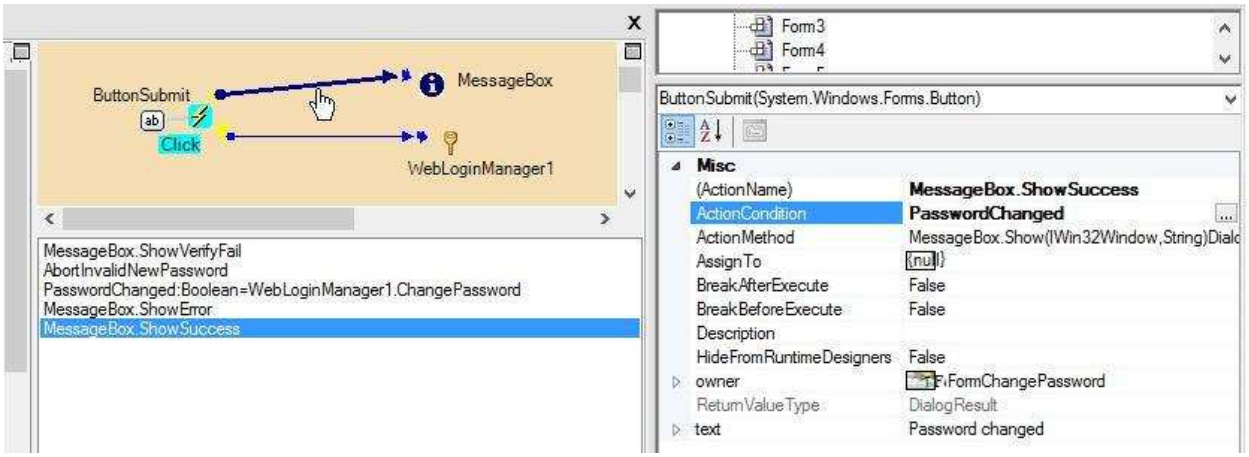



Action Properties
Back
OK
Cancel
(ActionName)
MessageBox.Show
Description
ReturnValueType
DialogResult
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
MessageBox.Show(IWin32Window,String)DialogR
owner
IWin32Window()
text
AssignTo
ConstantValue
Property
MathExpression
Object picker
Back
Next
Exit
Add
FormChangePassword from DrawingPage
Primary types
TextBoxLoginName of TextBox.Text
TextBoxCurPass of TextBox.Text
TextBoxNewPass1 of TextBox.Text
Action Properties
Back
OK
Cancel
(ActionName)
MessageBox.ShowSuccess
Description
ReturnValueType
DialogResult
HideFromRuntimeDesigner
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
MessageBox.Show(IWin32Window,String)DialogResult
owner
FormChangePassword
text
Password changed
AssignTo
(null)
Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
null
Val
0
A property



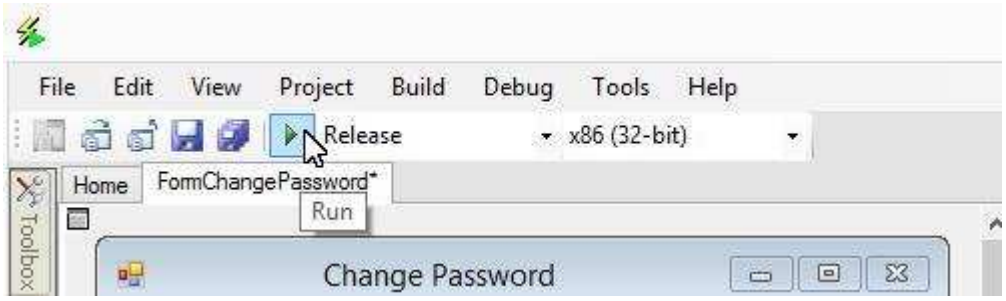


The action is created and assigned to the button:

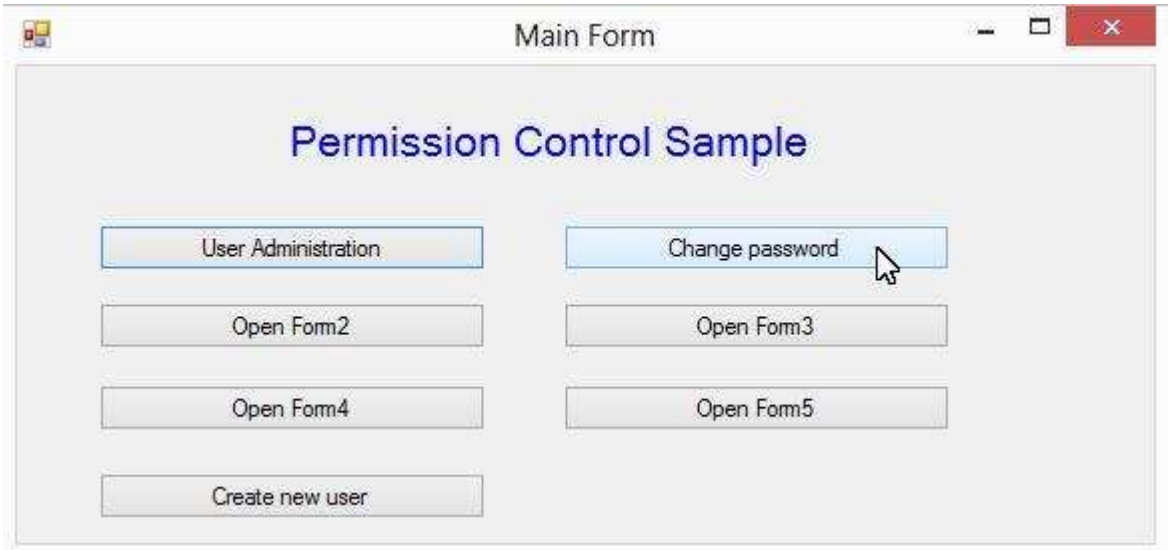


Test

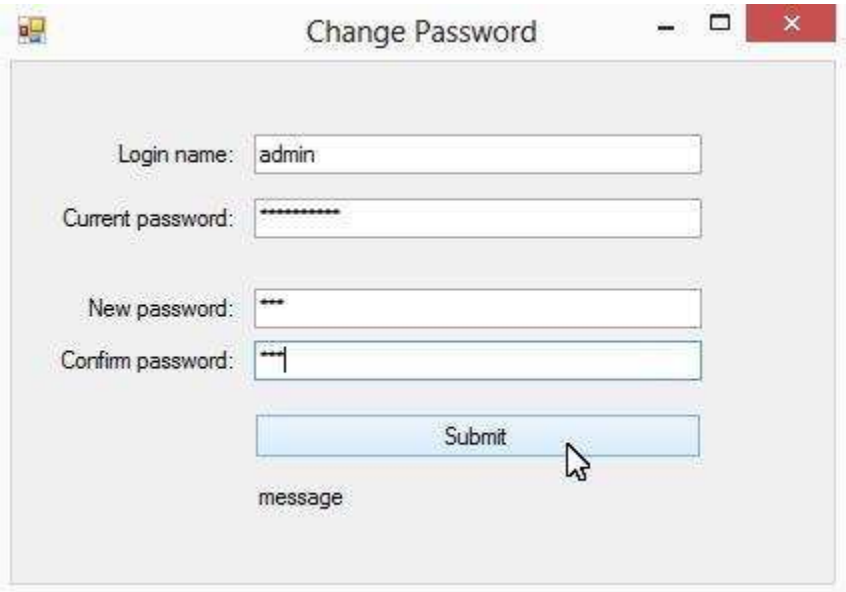
Now we may change the password of the first user record.



Click “Change password”:

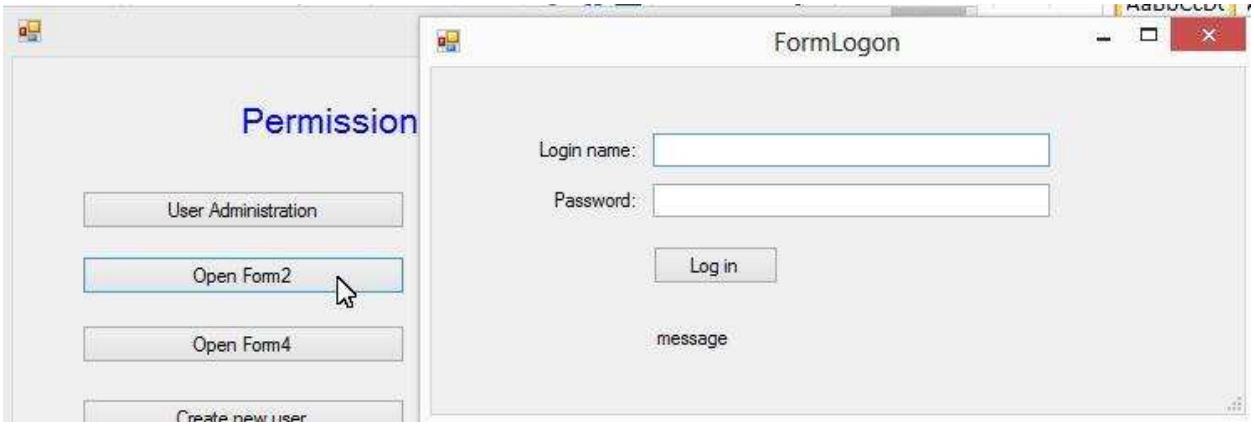


Enter the login name and the password; enter new password; click Submit:

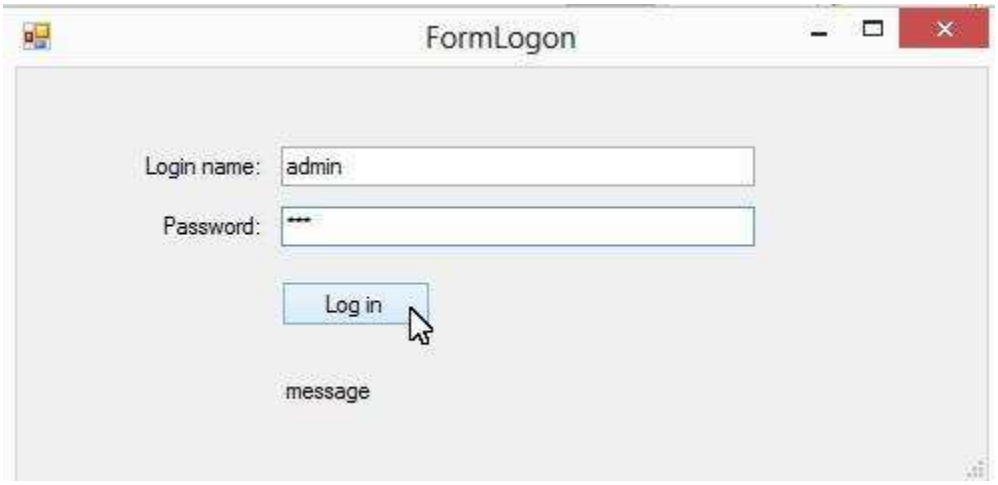


We get message “Password changed”:

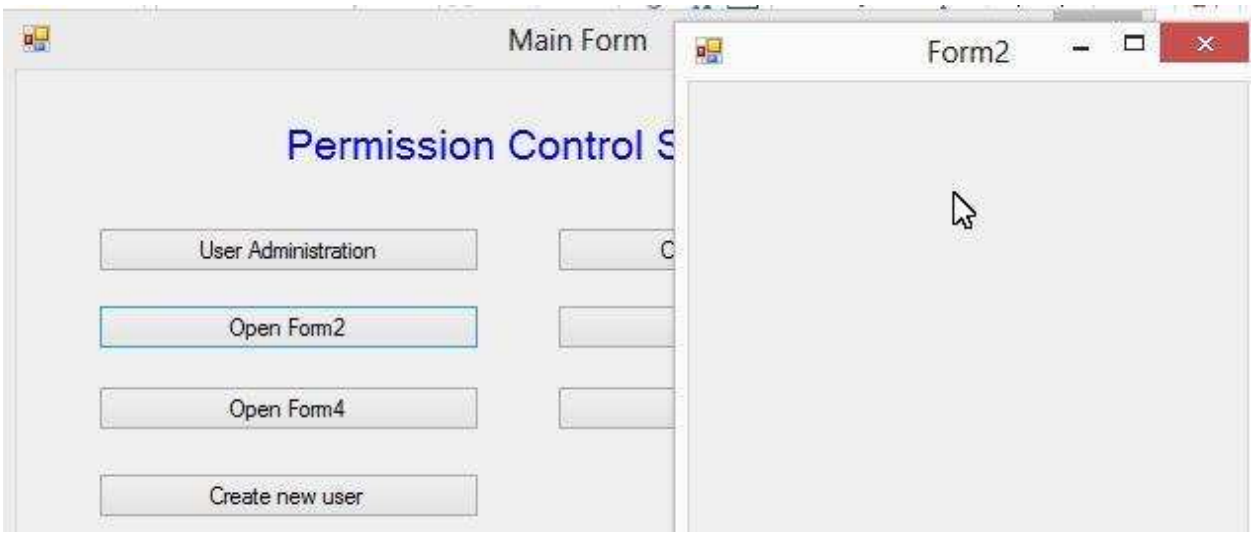
Let’s try to use our new password. Click “Open Form 2”, the login form appears:



Enter login name and new password; click “Log in”:



The login form closes and Form 2 appears:



We can see that our new password works.

Reset password and permission controls

See Part B2 from http://www.limnor.com/support/LoginManagerPartB2.PDF

Web Samples

PHP Web Project Sample

See Part C from http://www.limnor.com/support/LoginManagerPartC.PDF

ASPX Web Project Sample

See Part D from http://www.limnor.com/support/LoginManagerPartD.PDF