

Login Manager Windows Form Sample

Contents

Introduction	2
Login Management Framework.....	2
Windows Form Application Sample.....	2
Start Form	2
Login Form	6
UI	6
User table.....	6
Add Login Manager Component.....	7
Set properties of Login Manager	8
Execute login action	11
Close login form	15
Protect Forms.....	17
Password Management	19
Add new user	20
Input verification.....	20
Login name usage checking	24
Create new user record	29
Create new password	32
Send password to user.....	37
Show notification message	42
Set email address	44
Remove email address	47
Protect new user form	50
Change password	53
Input verification.....	53
Execute password change.....	61
Show error message	66

Show success.....	69
Test.....	72
Reset password and permission controls	75
Web Samples	75
PHP Web Project Sample	75
ASPX Web Project Sample	75

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

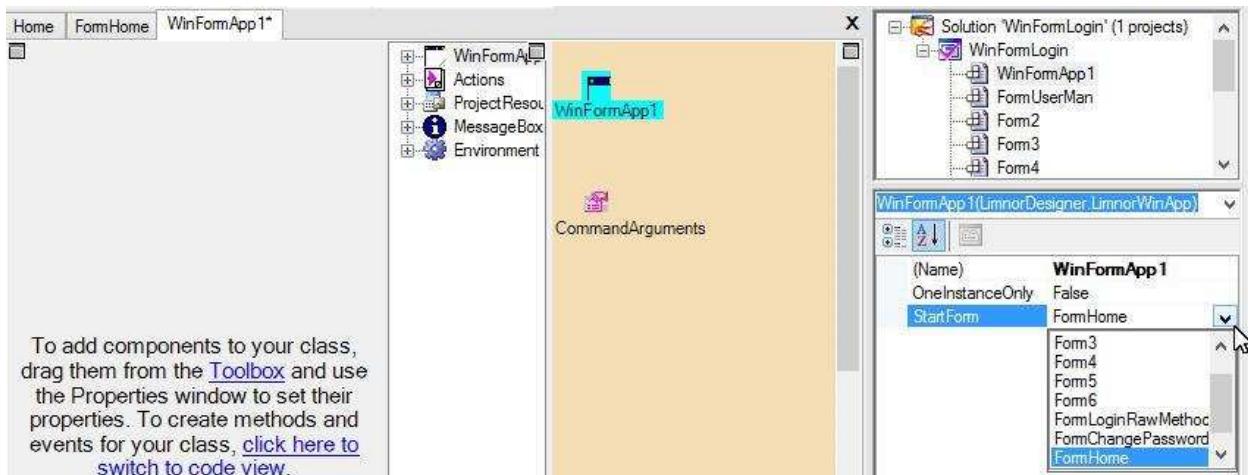
See part A from <http://www.limnor.com/support/LoginManagerPartA.PDF>

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.

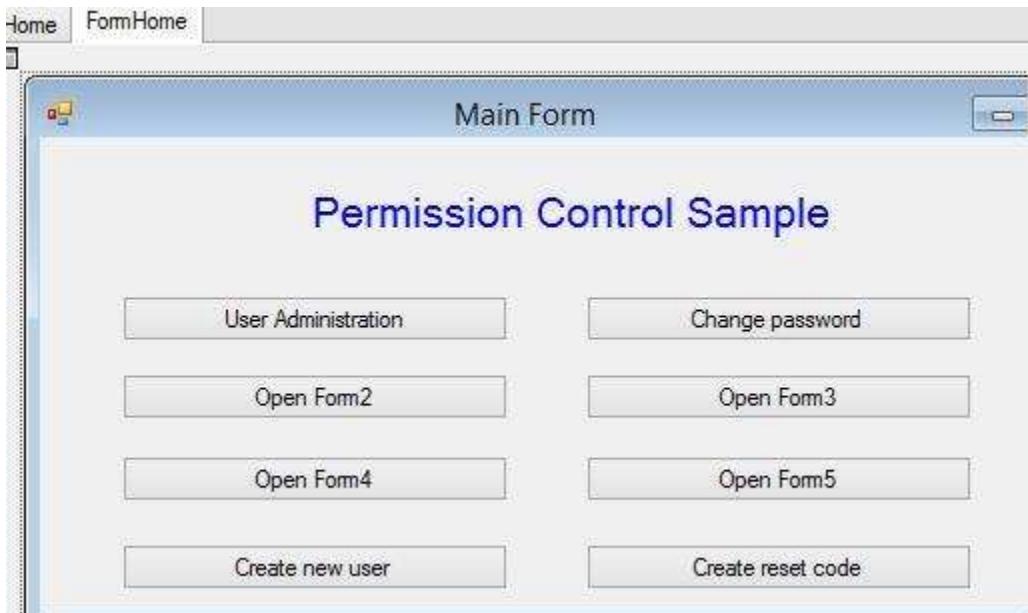
Start Form

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.

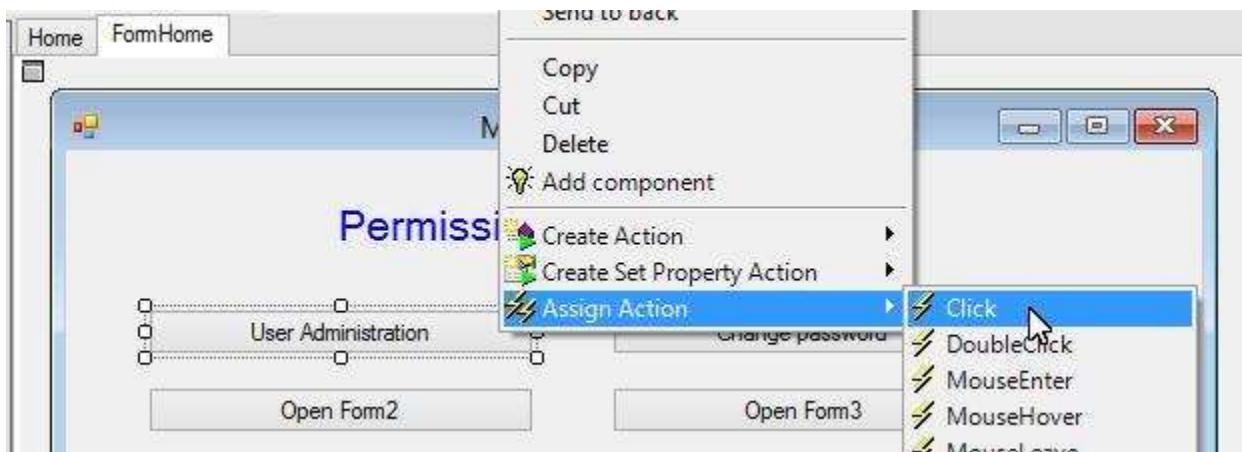


When the Start Form is closed the application shuts down. Thus, usually you should use a form which is always visible as the Start Form. For example, a MDI parent form is usually good for being the Start Form. Since a login form usually will be closed after log in, a login form usually is not good to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

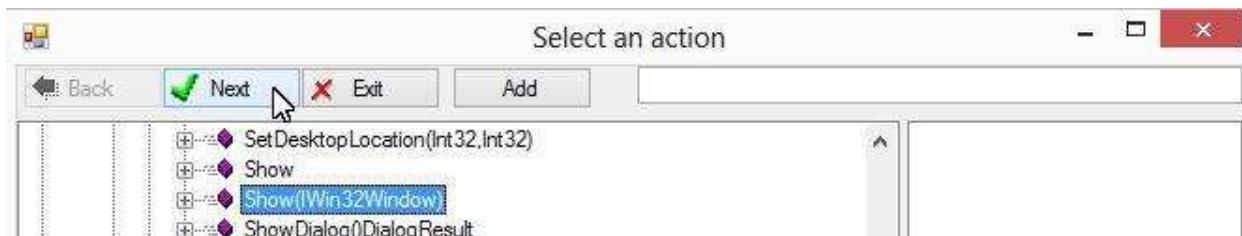
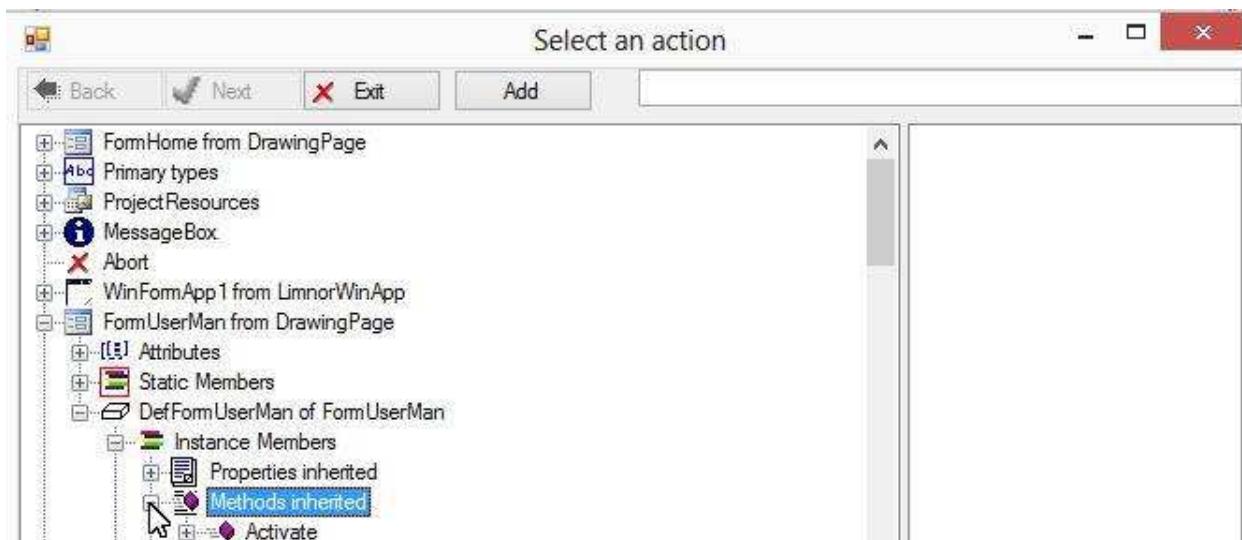
In this sample, we use such a Start Form as shown below:



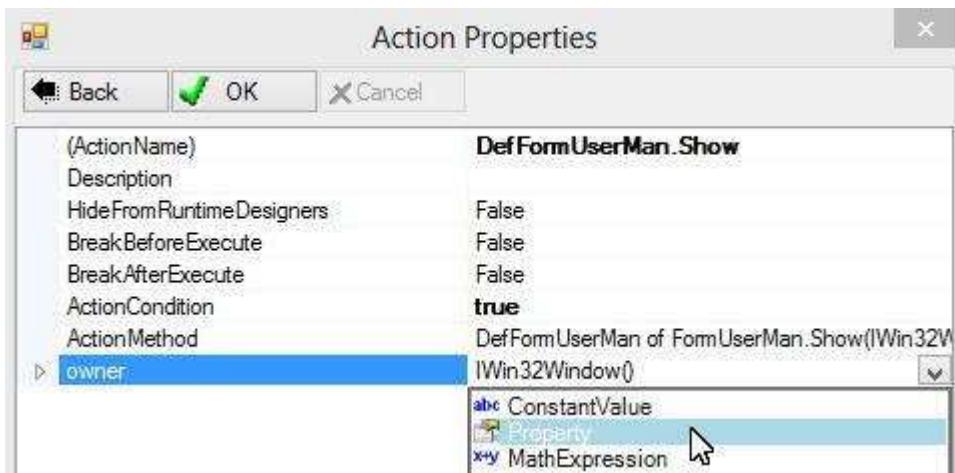
In this sample form, we have several buttons for opening other forms. The programming of these buttons is not affected by permission controls. Here we show the programming of the “User Administration” button. Right-click the button; choose “Assign Action”; choose “Click” event:



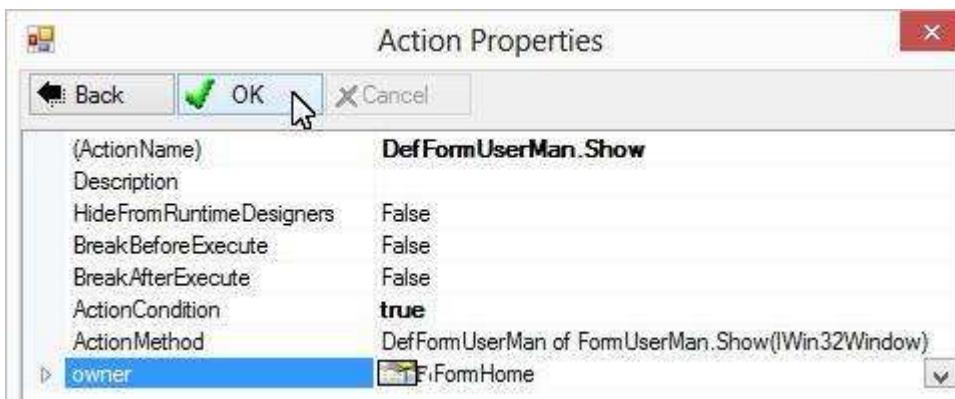
Select Show method of the default instance of the form FormUserMan:



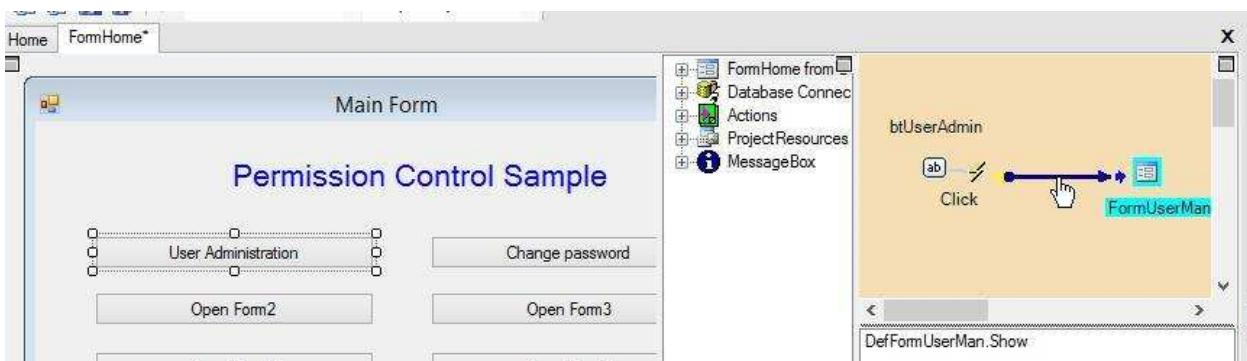
Select the current form as the owner for showing the user administration form:



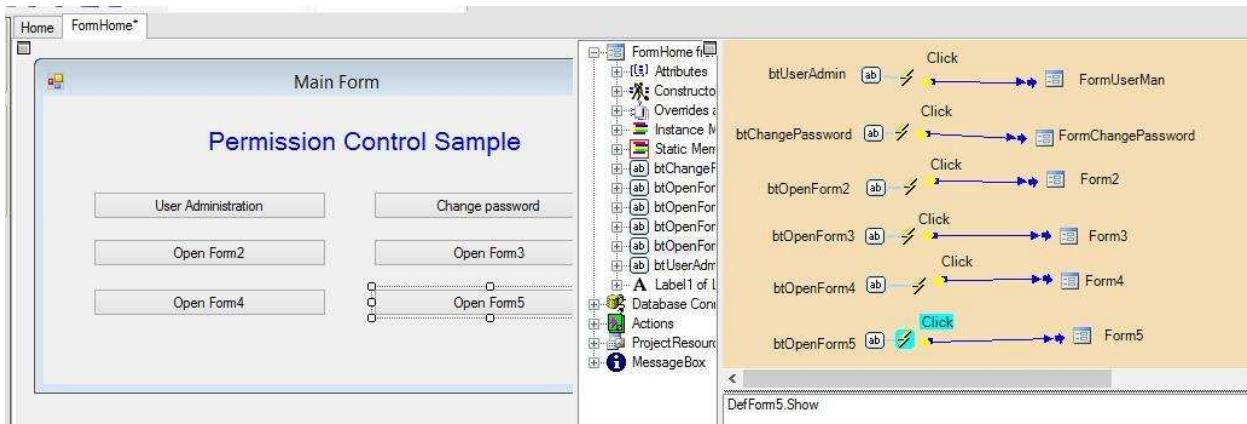
Click OK to finish creating this action:



The action is created and assigned to the button:



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



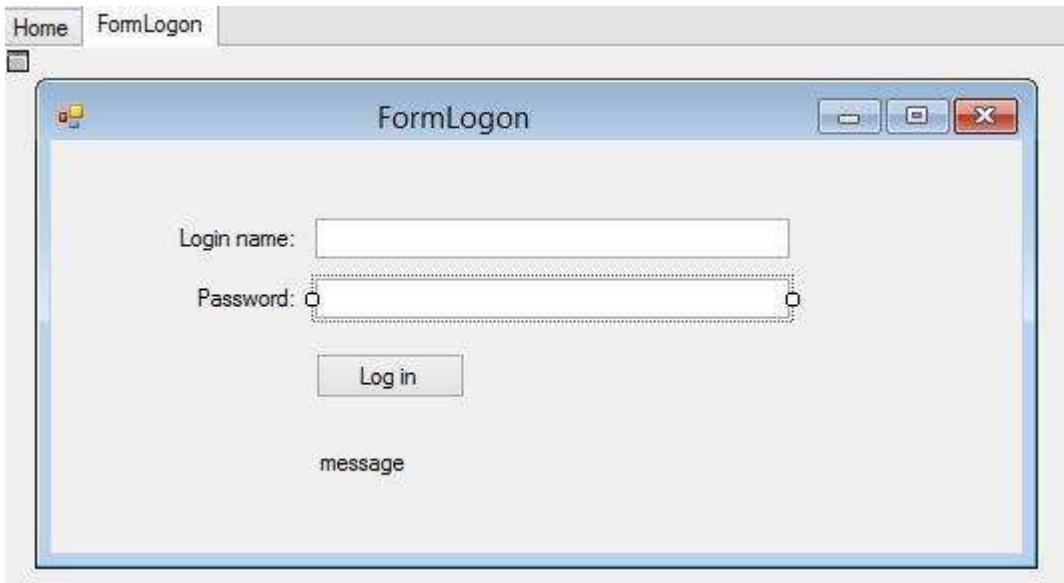
The programming in this section has nothing to do with login management and permission controls. It is normal Windows Form programming. For more information on Window Form programming, see <http://www.limnor.com/support/UseDefaultFormInstance.pdf>.

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	

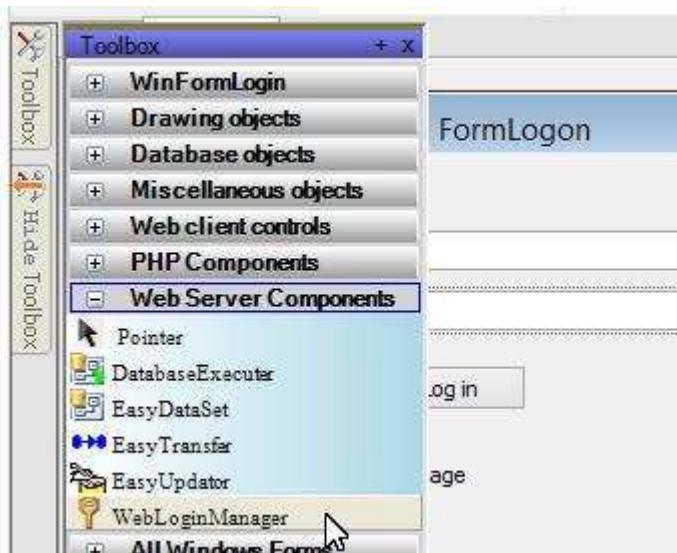
The SQL script for creating the above table is as following:

```
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.

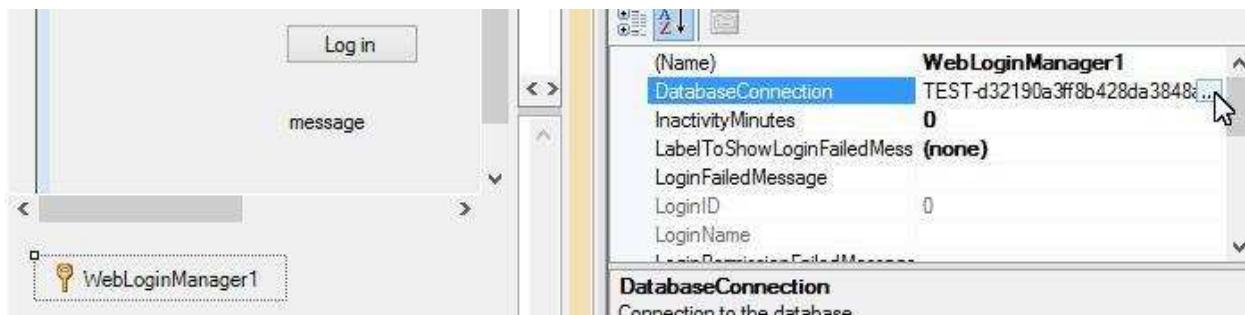
Add Login Manager Component

Drop WebLoginManager to the form.

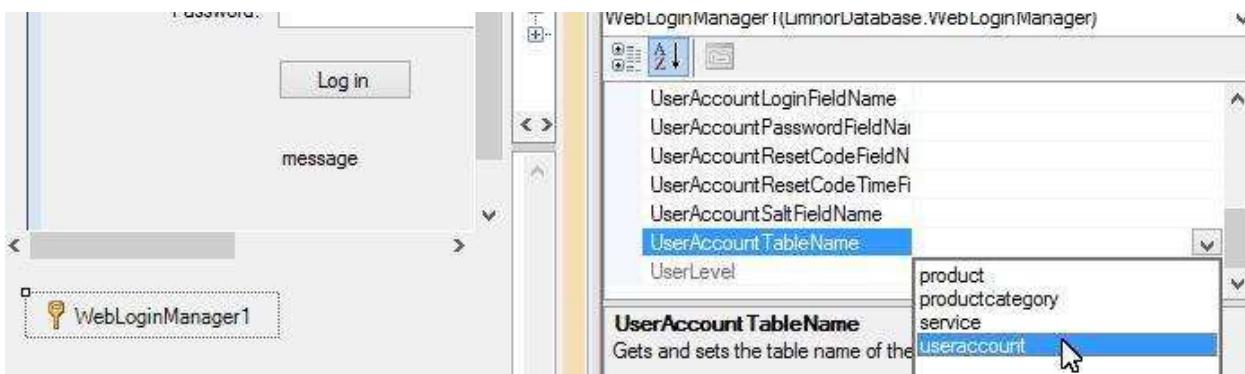


Set properties of Login Manager

Set database connection:

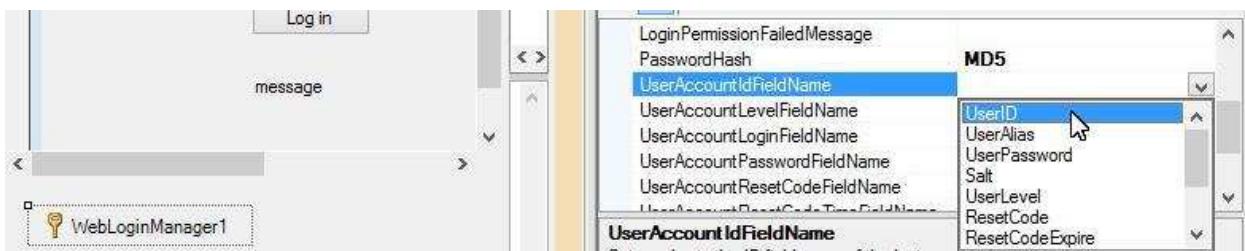


Specify user table:

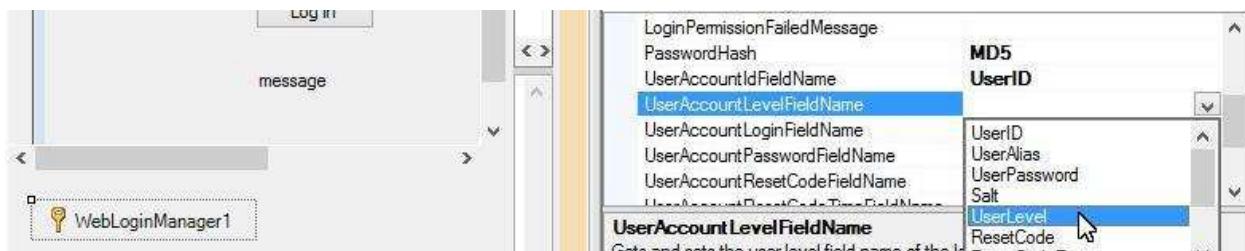


Specify user ID field (this is optional):

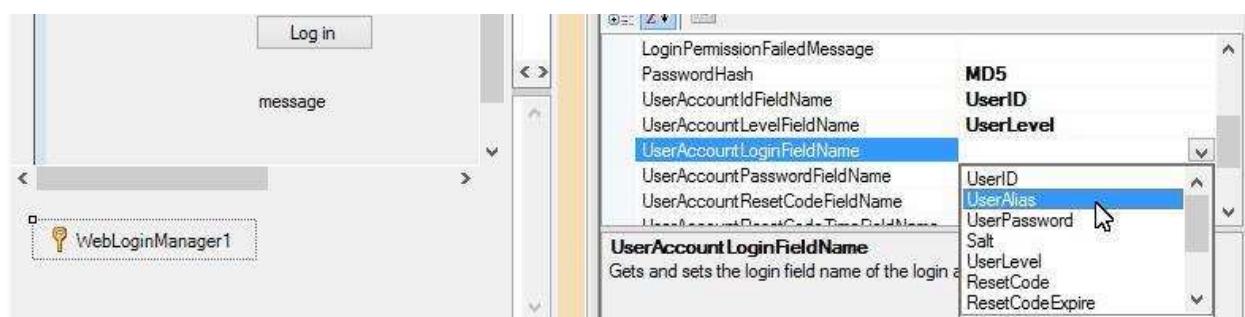
Login Manager Windows Form Sample – Part 1 | 2013



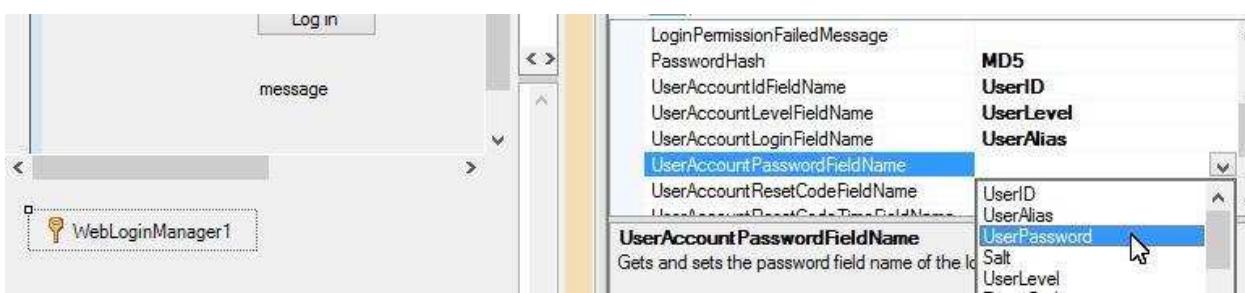
Specify user level field (this is optional):



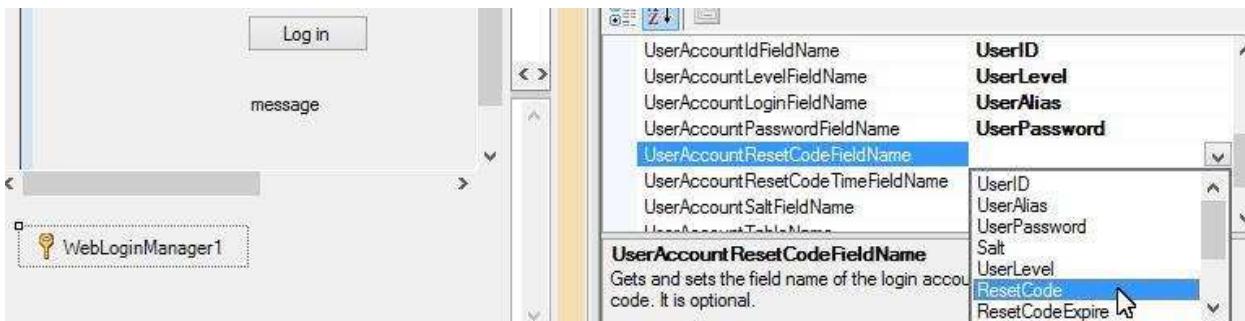
Specify user login name field (this is required):



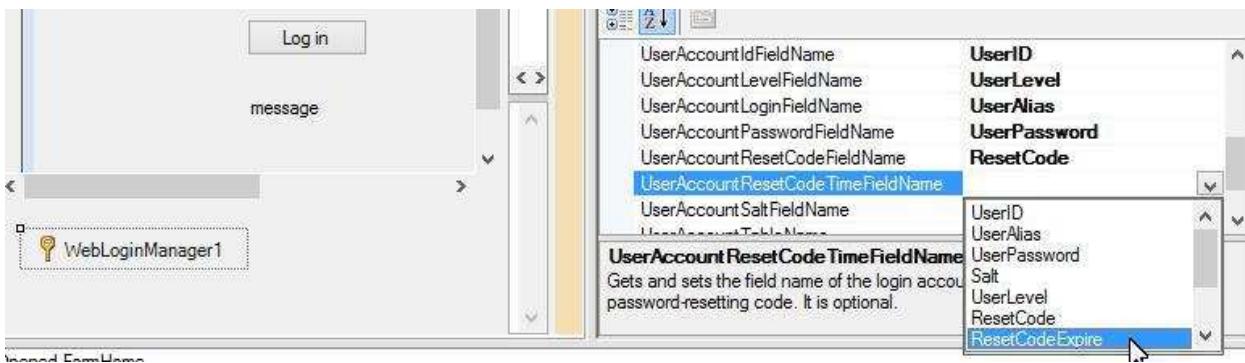
Specify user password field (this is required):



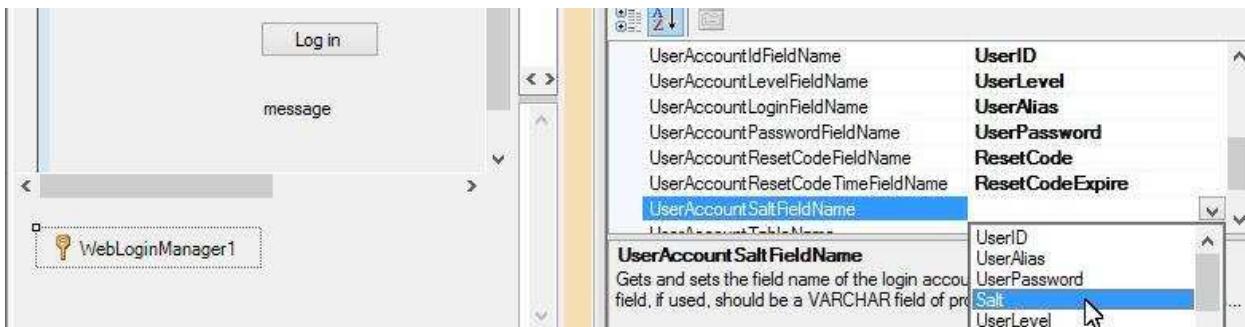
Specify reset code field (this is optional):



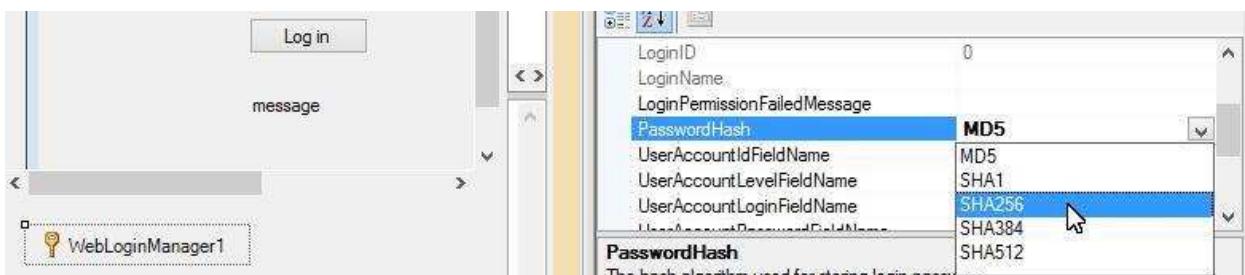
Specify reset code expiration field (this is optional):



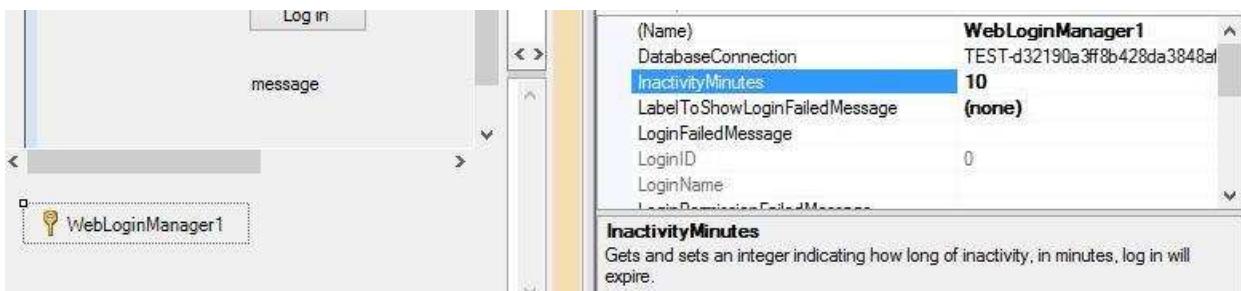
Specify salt field (this is optional):



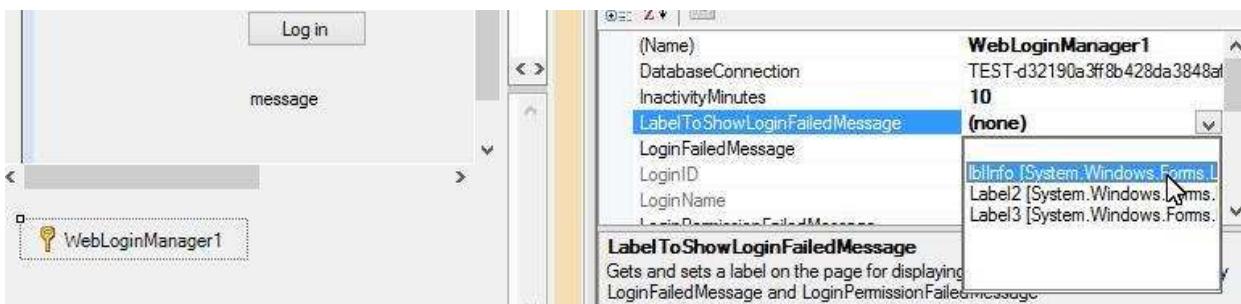
Specify password hash algorithm. For this sample, we use SHA256:



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.

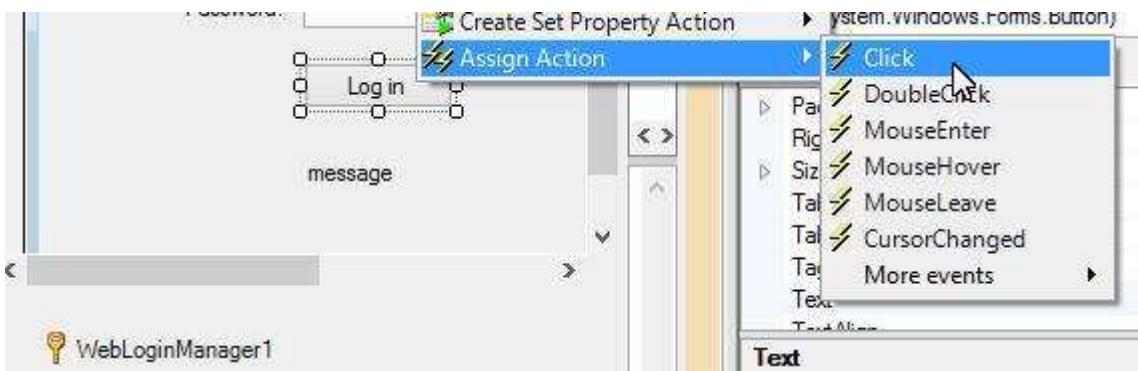


Specify error message display label:



Execute login action

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



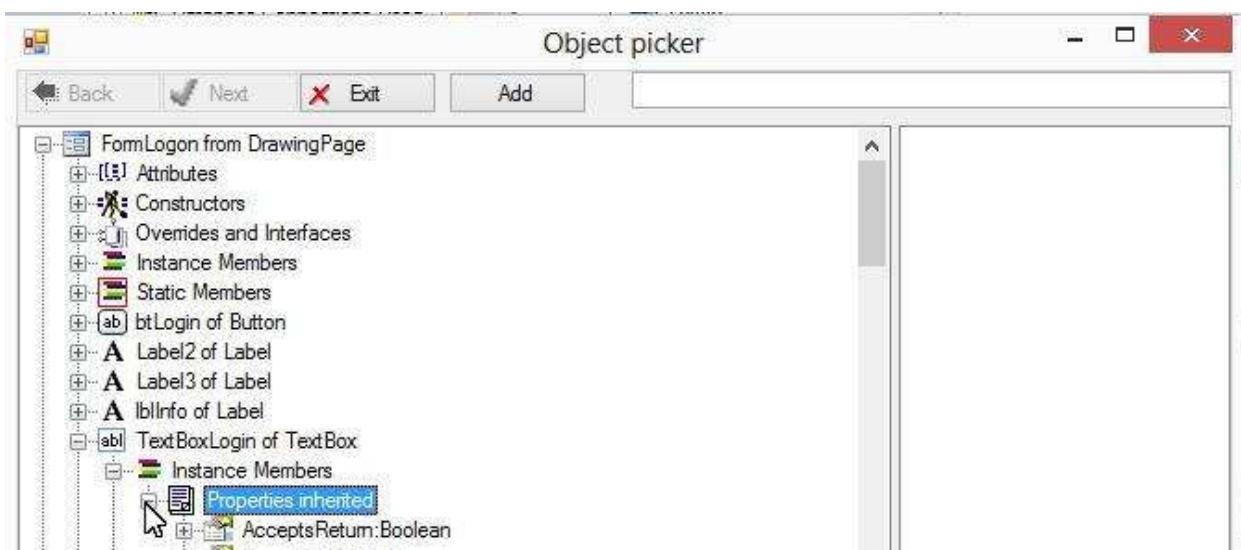
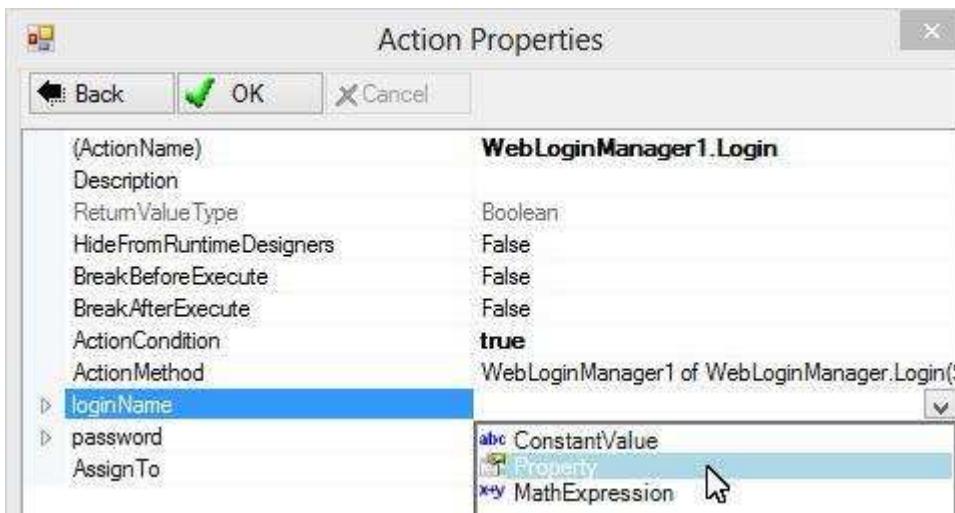
Choose the Login method of the Login Manager:

The screenshot shows two instances of the Object Browser window. The top window displays the members of `FormLogon`, which inherits methods from `WebLoginManager1`. The bottom window displays the members of `WebLoginManager1`, which includes its own methods like `ChangePassword` and `Login`.

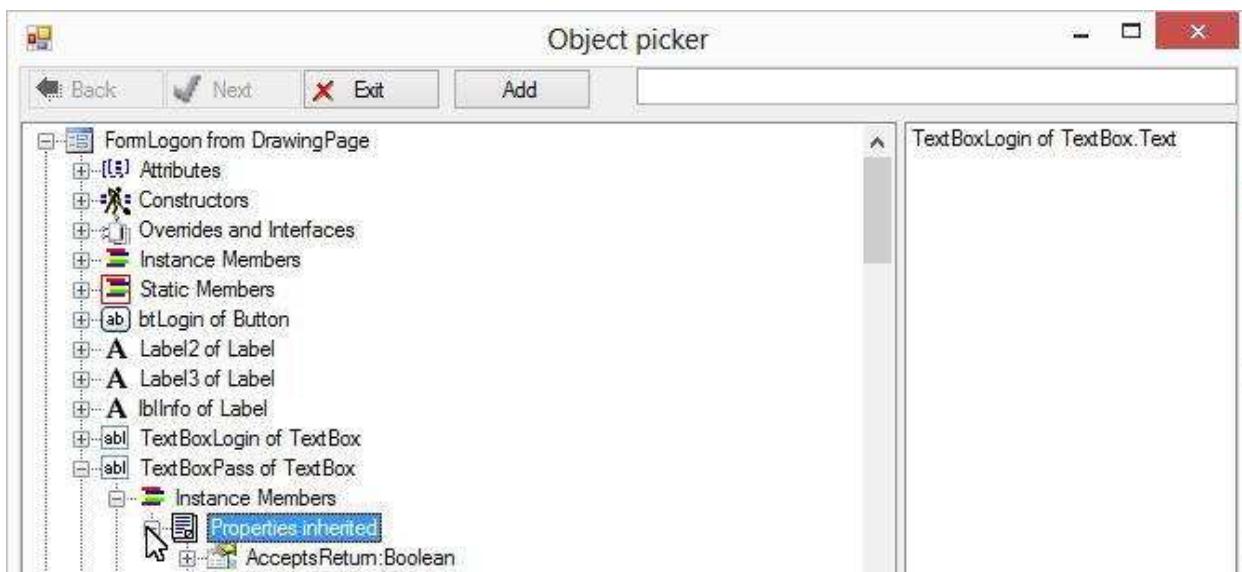
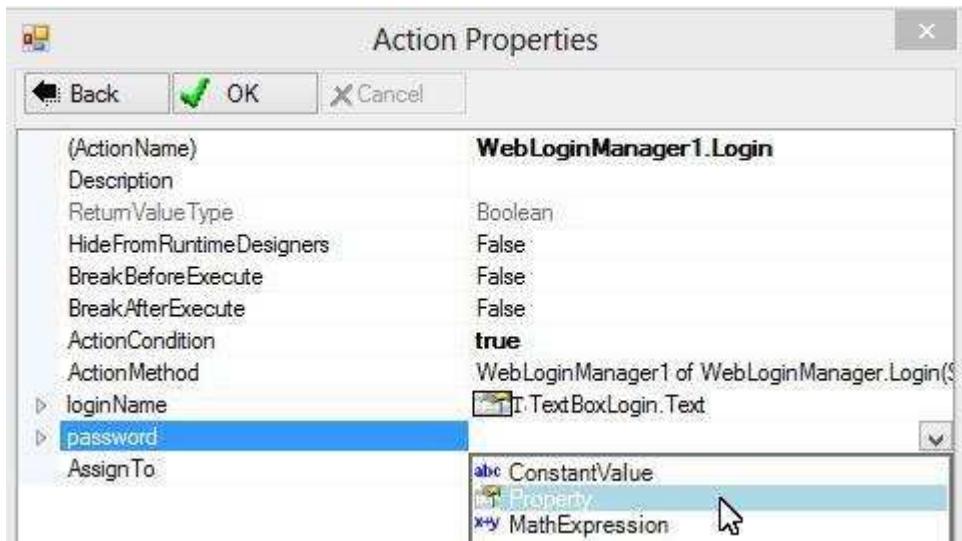
Object Browser (Top Window):

- `FormLogon from DrawingPage`
 - `Attributes`
 - `Constructors`
 - `Overrides and Interfaces`
 - `Instance Members`
 - `Static Members`
 - `btLogin of Button`
 - `A Label2 of Label`
 - `A Label3 of Label`
 - `A lblInfo of Label`
 - `sbl TextBoxLogin of TextBox`
 - `sbl TextBoxPass of TextBox`
 - `WebLoginManager1 of WebLoginManager`
 - `Instance Members`
 - `Properties inherited`
 - `Methods inherited` (highlighted)
- `ChangePassword(String, String, String)Boolean`

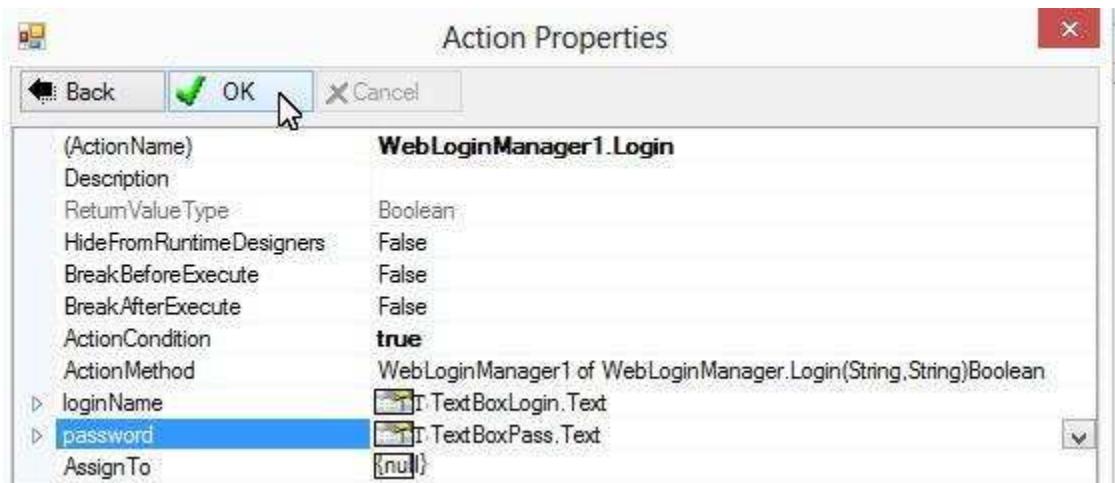
Select the `Text` property of login name text box for the `loginName`:



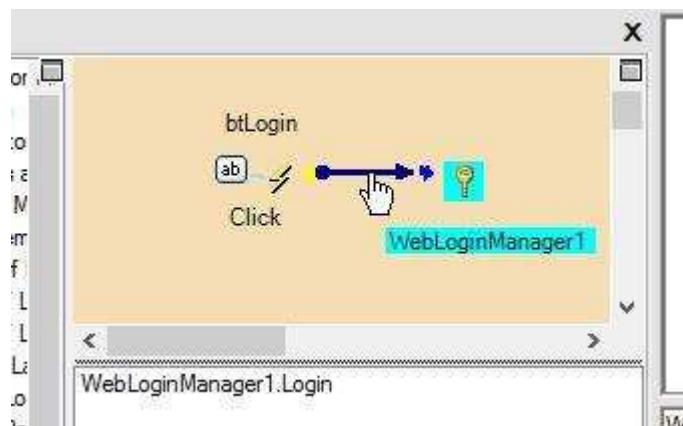
Select the Text property of the password text box for “password” parameter:



Click OK to finish creating this action:

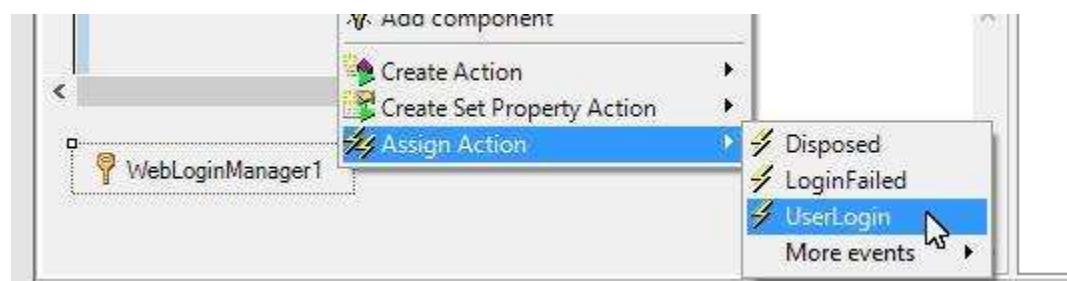


The action is created and assigned to the button:



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:



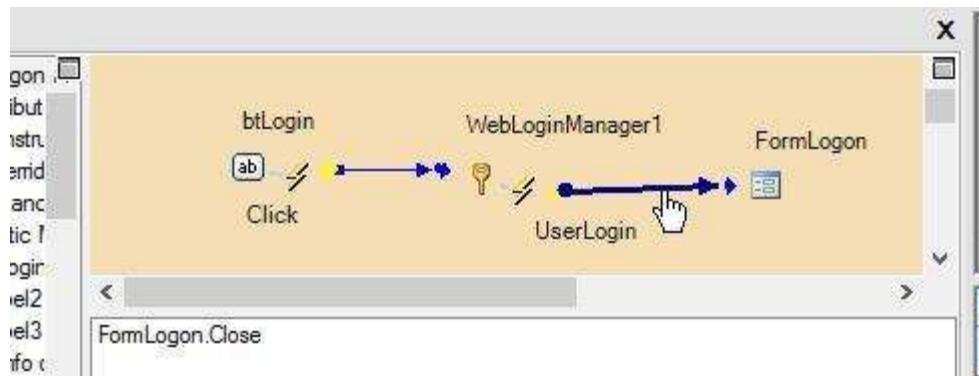
Choose Close method of the current form:



Click OK to finish creating this action:



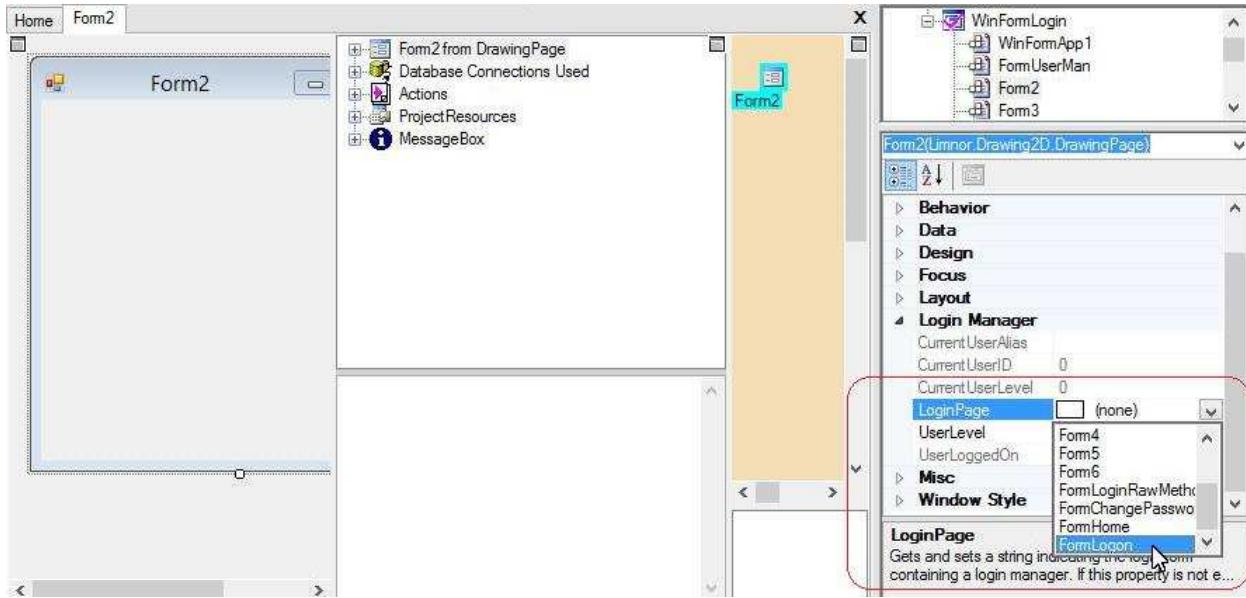
The action is created and assigned to the event:



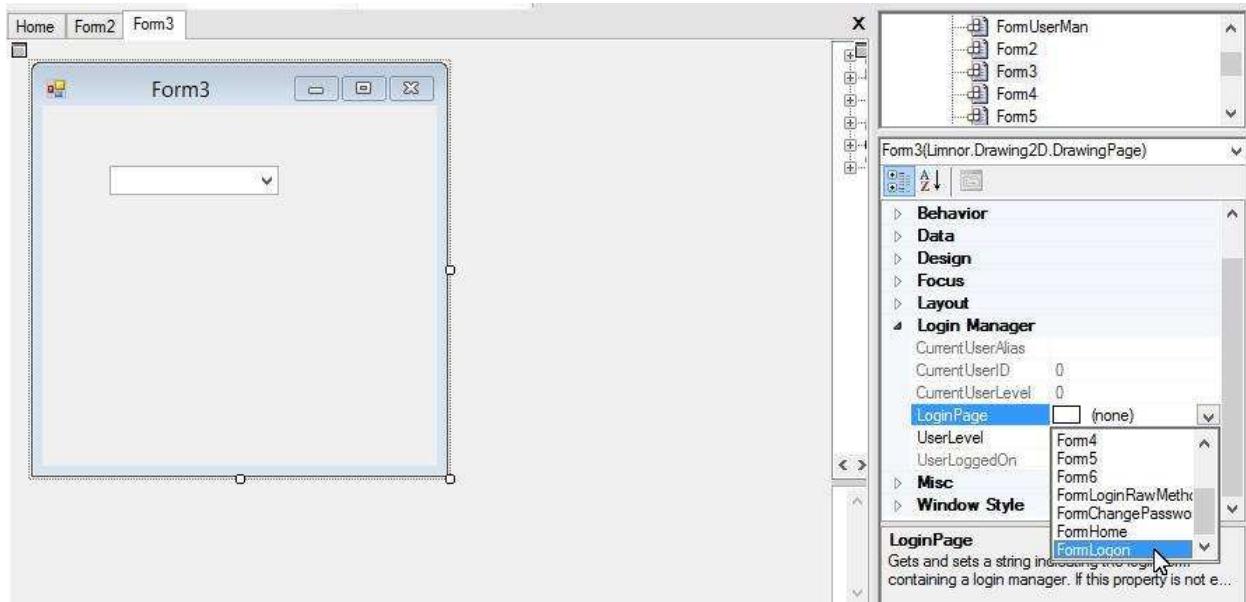
Note that this step of handling UserLogin event to close the login form is only needed for a Windows Form application. For a web application, this handling is not needed. For a web application, after logging on, target web page is automatically loaded.

Protect Forms

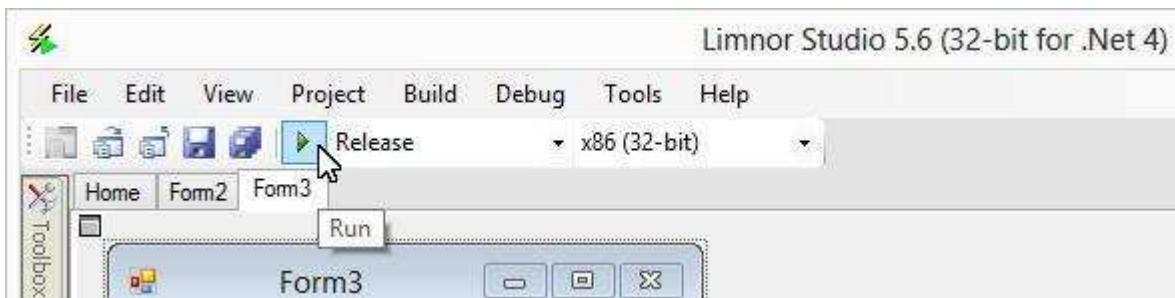
Once we have a login form, we may protect forms. For a form we want to protect, set its LoginPage property to the login form:



We may do it for all forms to be protected:



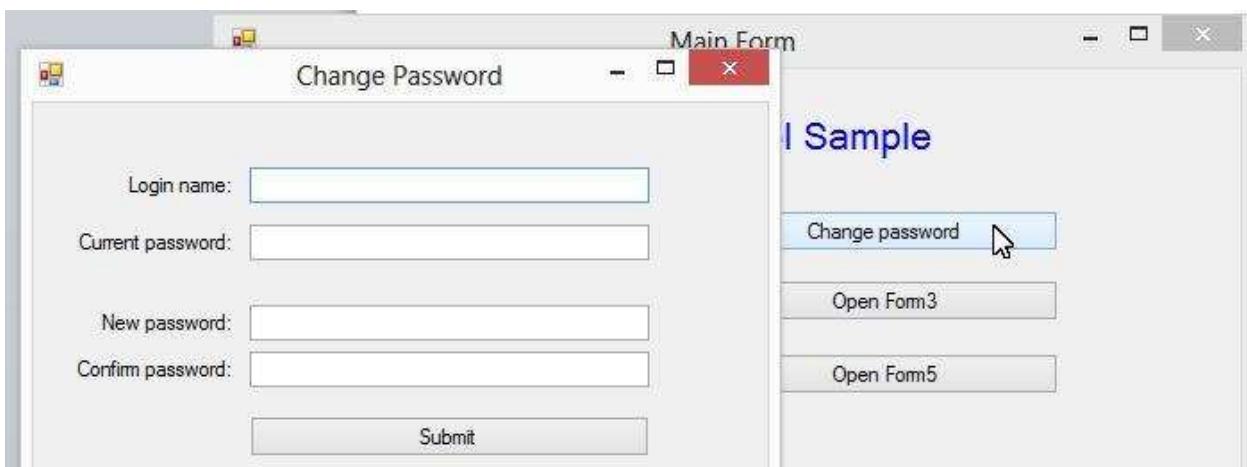
We are done adding permission control to the application. Let's see its effects. Run the project:



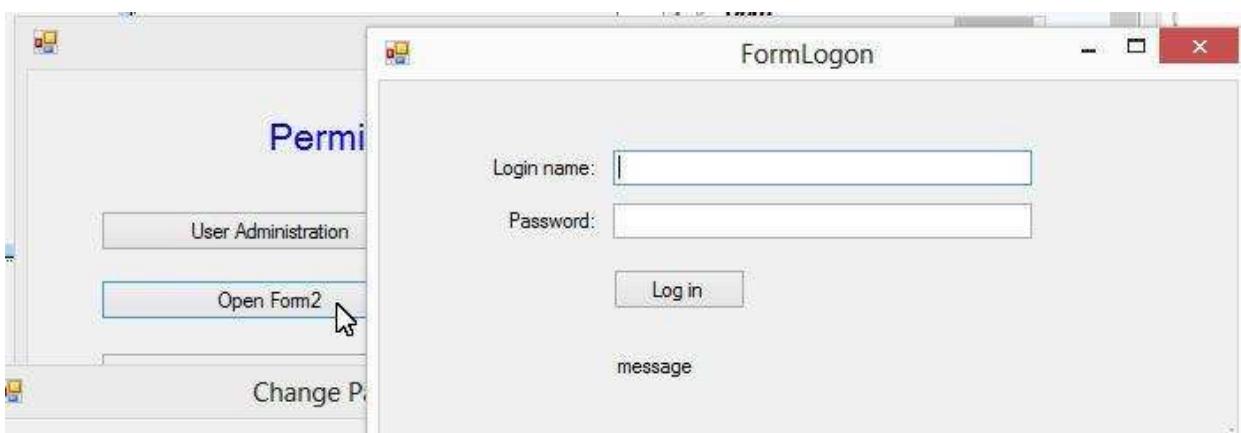
The Start Form appears:



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

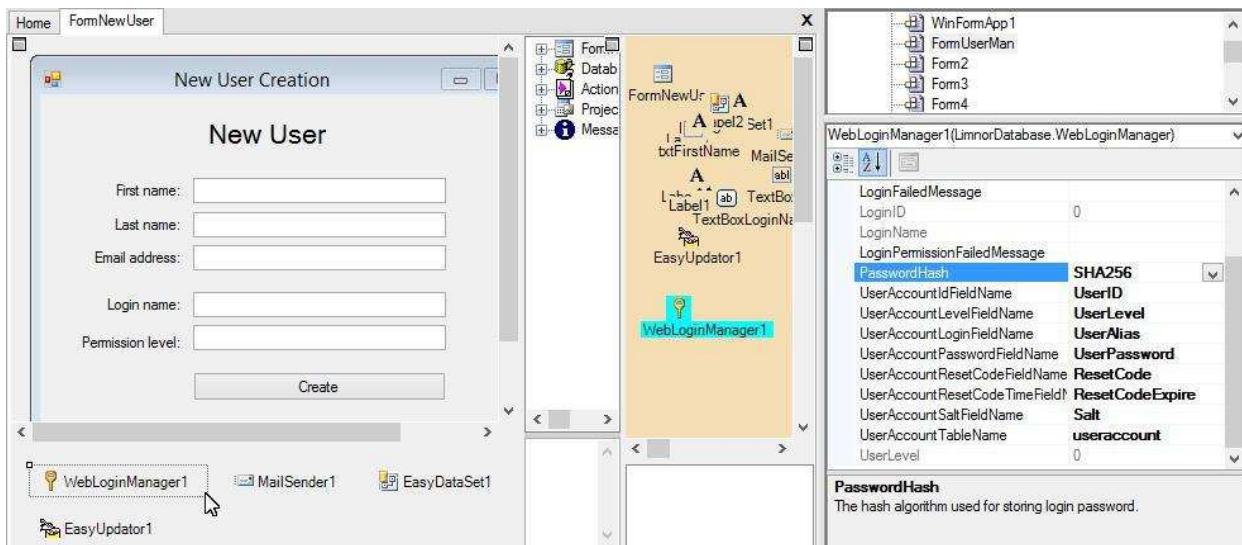
For developing data entry forms, see <http://www.limnor.com/support/Limnor%20Studio%20-%20User%20Guide%20-%20Part%20VI.pdf>. Chapter 5 “Data Binding” describes data entry using EasyDataSet. Chapter 8 “Data Entry Using EasyGrid” describes data entry using EasyDataGrid.

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.

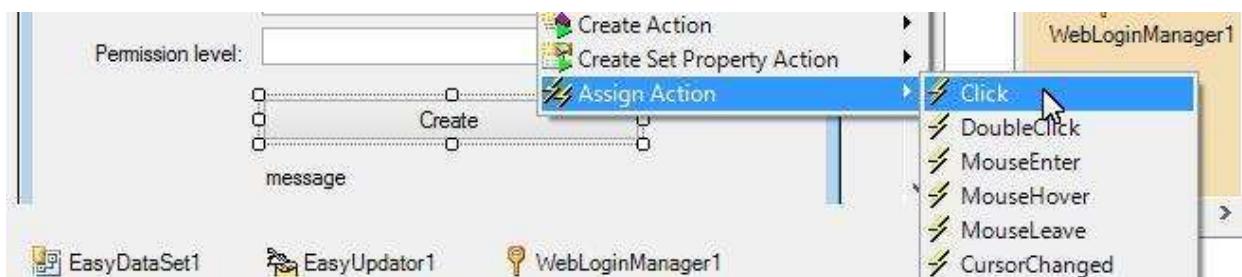
Note that the properties of the Login Manager Component must be set the same as we did for the Login Manager component in the login form.



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

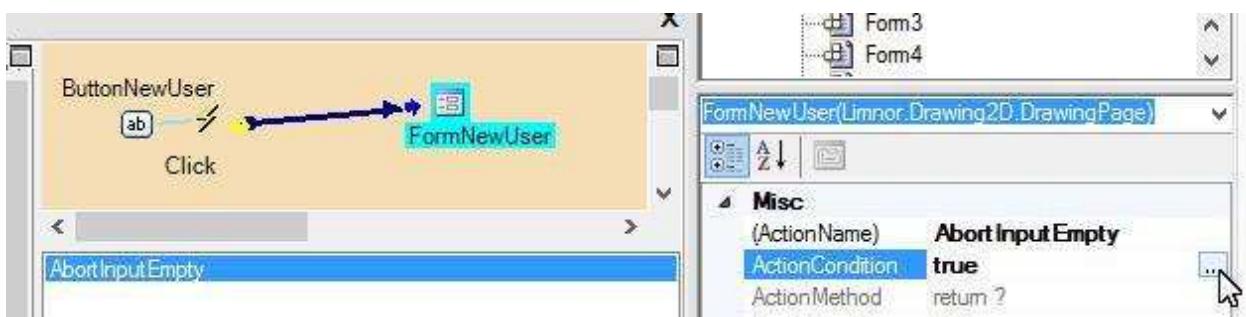
Each programming starts by right-clicking the button; choose “Assign Action”; choose “Click” event.



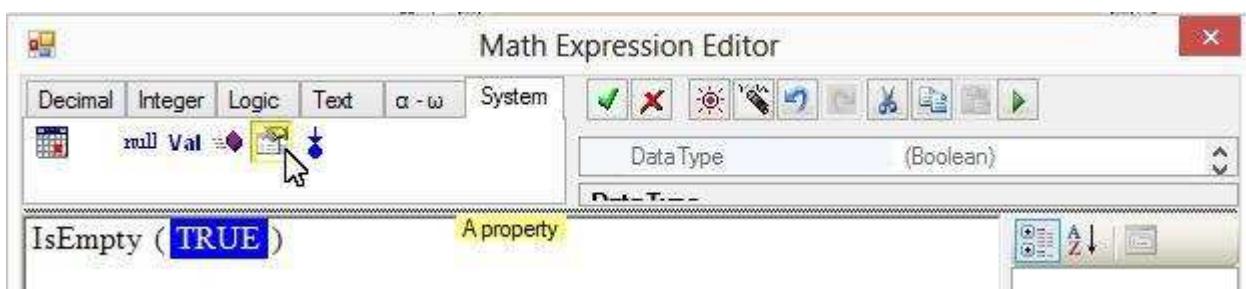
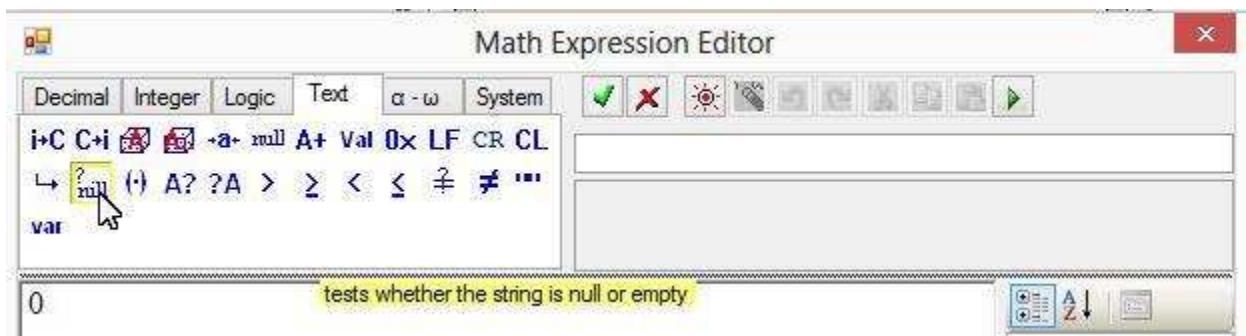
Select the Abort action:

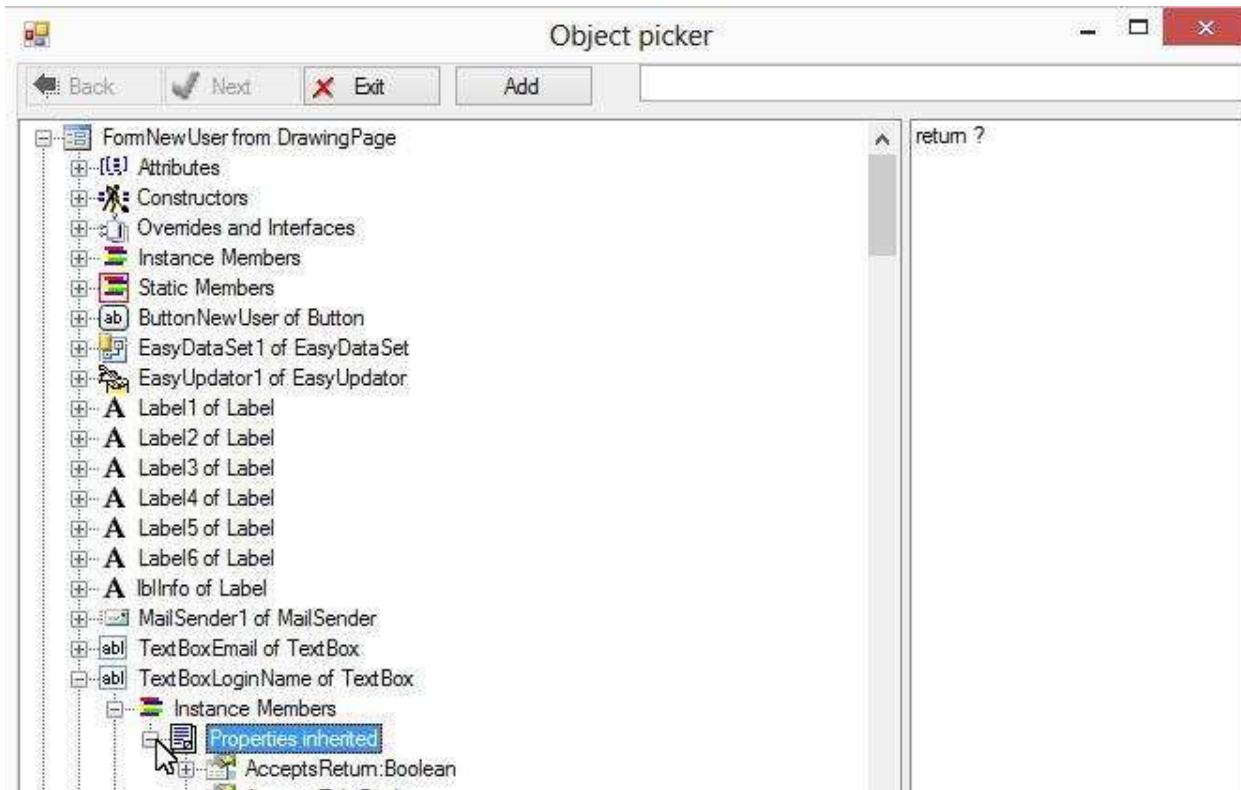


The abort action is assigned to the button. Rename it and set its ActionCondition to verify inputs:

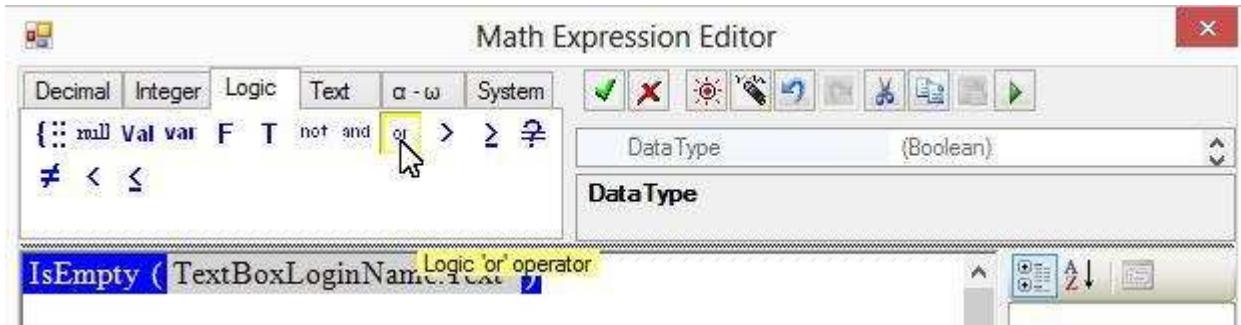


The conditions are that the login name or email address is empty:

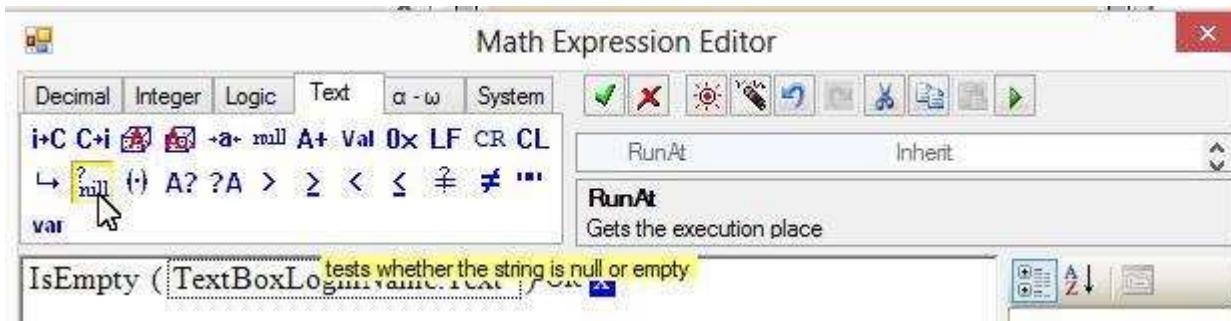




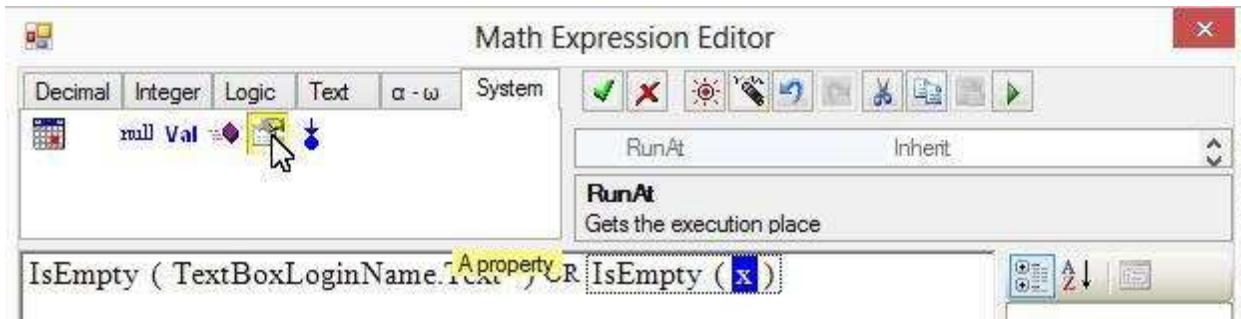
Click "IsEmpty" and then click "or":



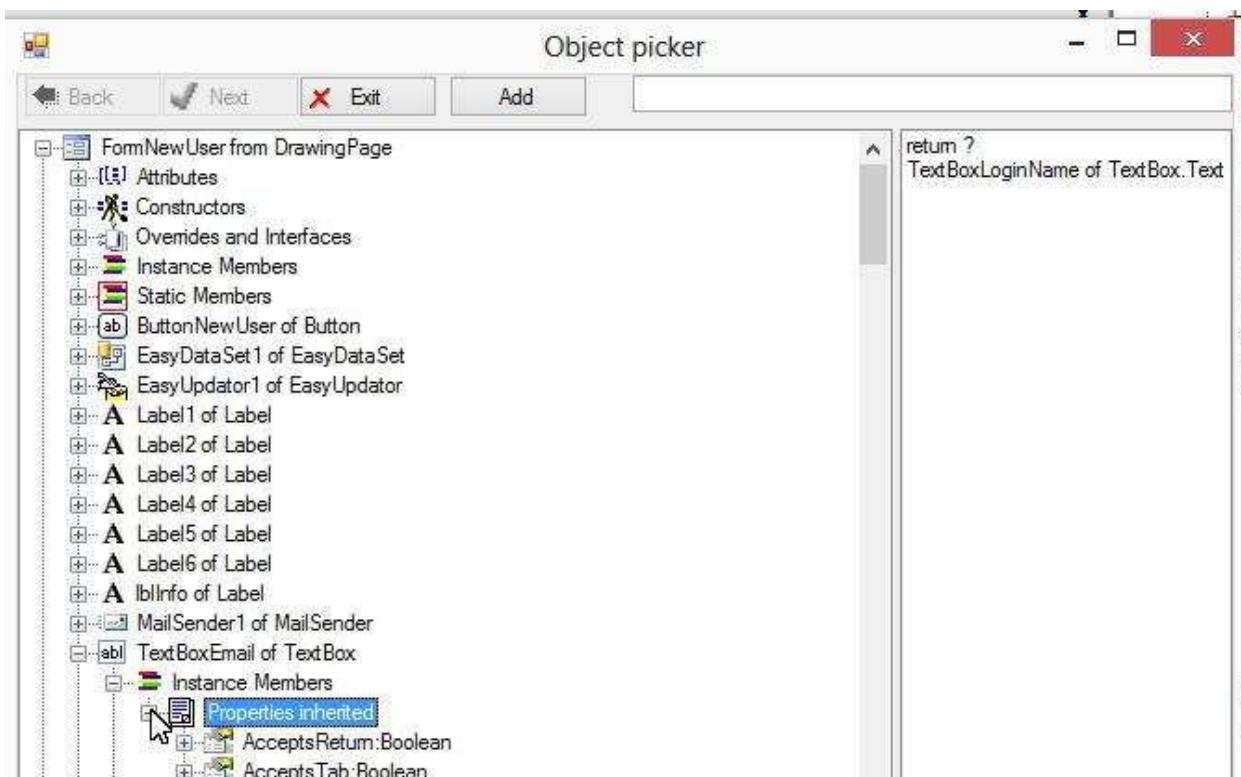
Click "x" and then click "?null":

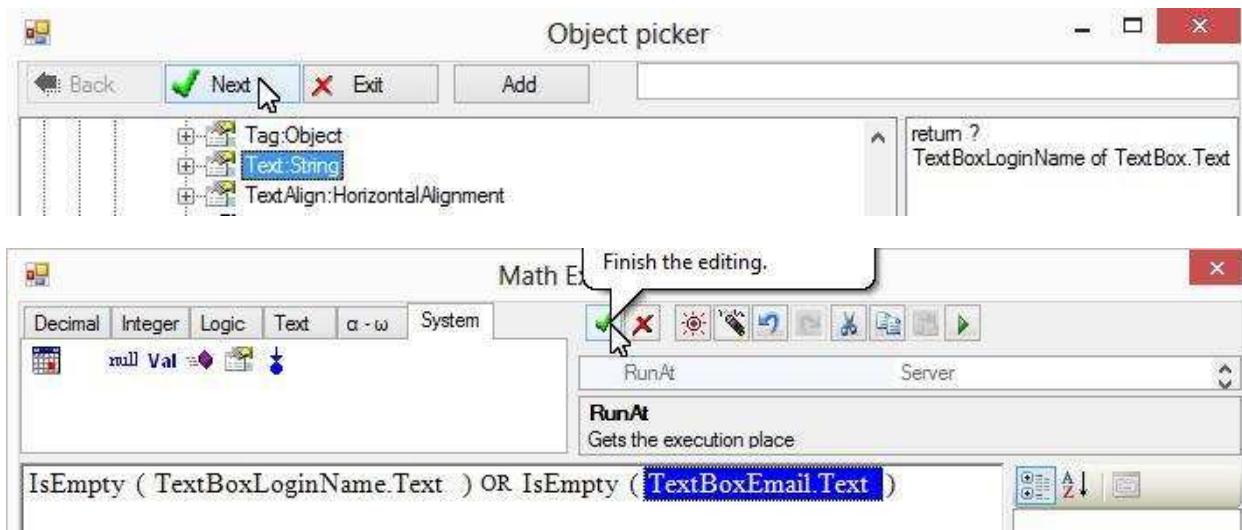


Click “x” inside “IsEmpty”, then click the property icon:



Select Text of the email address text box:



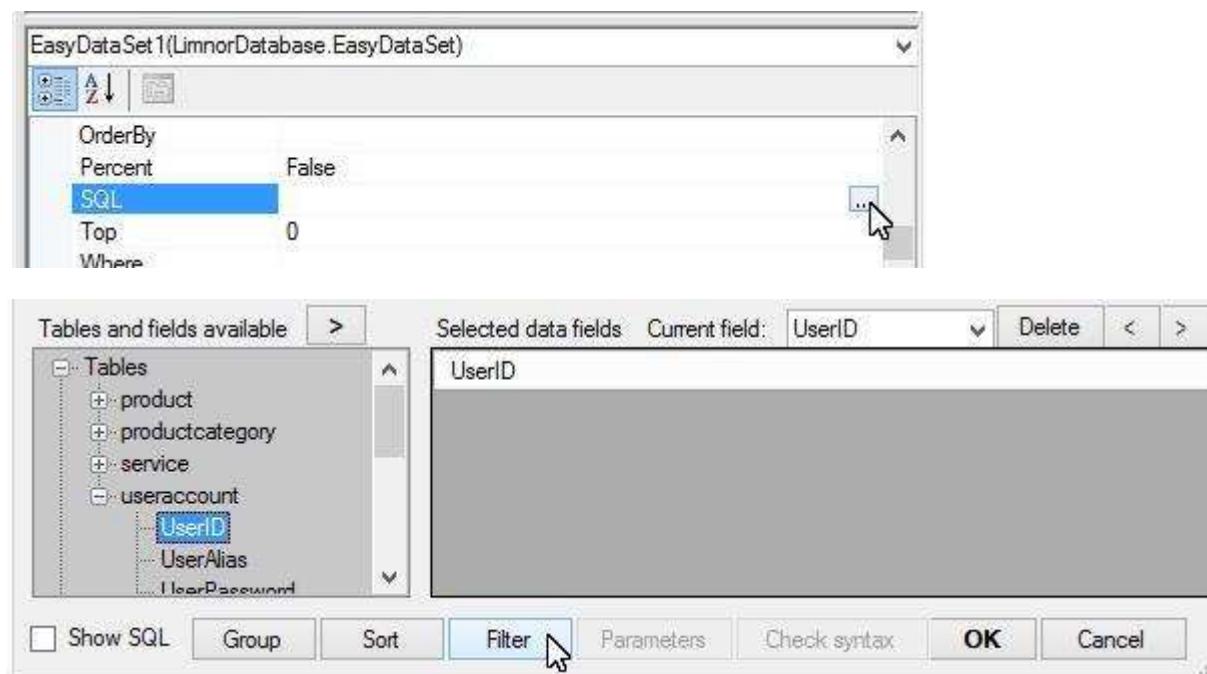


The input verification is done:



Login name usage checking

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



The screenshot shows the configuration of a WHERE clause for a database query. The top section displays the filter setup:

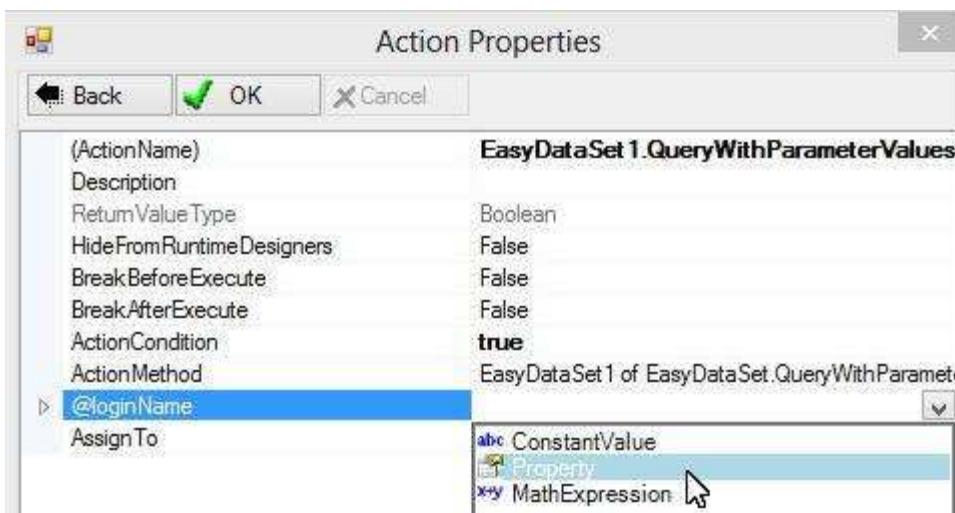
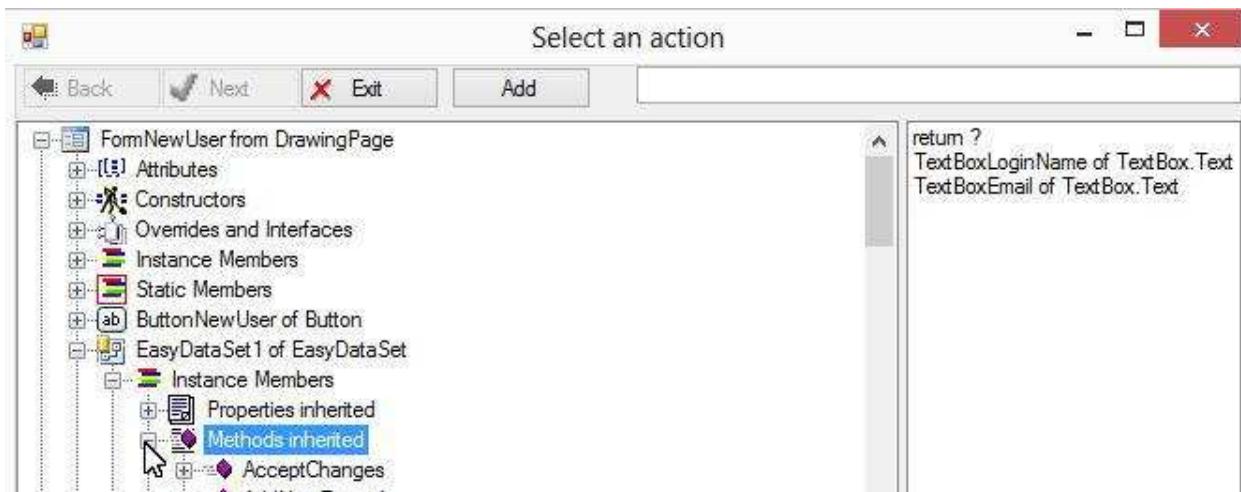
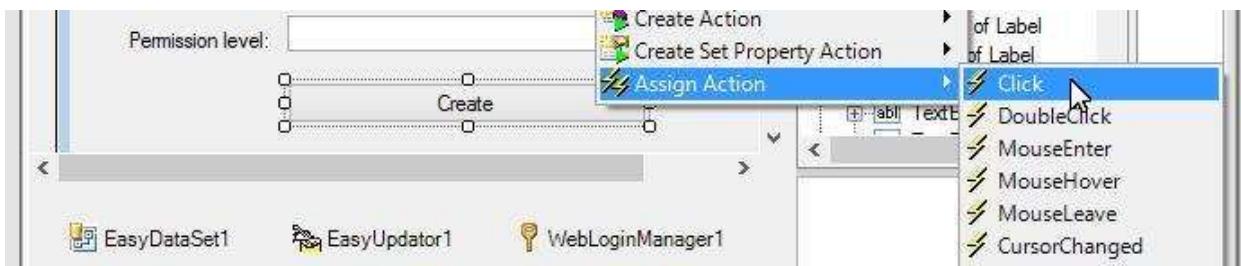
- Select field:** Table: useraccount, Field: UserID (selected), Operator: =, Value: Const.
- Select field:** Table: useraccount, Field: UserAlias, Operator: =, Value: Param (Name: @loginName).

Below this, the **Filters (WHERE clause)** section contains the generated SQL fragment: 'useraccount'.UserAlias' = @loginName. Buttons for **Add OR filter** and **Add AND filter** are available.

At the bottom, the **Tables and fields available** section shows 'useraccount' with fields UserID and UserAlias. The **Selected data fields** section shows 'Current field: UserID'. The **SQL statement** section displays the full query: `SELECT `useraccount`.`UserID` FROM `useraccount` WHERE `useraccount`.UserAlias = @loginName`. Buttons for **Show SQL**, **Group**, **Sort**, **Filter**, **Parameters**, **Check syntax**, **OK**, and **Cancel** are at the bottom.

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

Login Manager Windows Form Sample – Part 1 | 2013



Object picker

Back Next Exit Add

- FormNewUser from DrawingPage
 - Attributes
 - Constructors
 - OVERRIDES AND INTERFACES
 - Instance Members
 - Static Members
 - ab ButtonNewUser of Button
 - EasyDataSet1 of EasyDataSet
 - EasyUpdator1 of EasyUpdator
 - A Label1 of Label
 - A Label2 of Label
 - A Label3 of Label
 - A Label4 of Label
 - A Label5 of Label
 - A Label6 of Label
 - A lblInfo of Label
 - MailSender1 of MailSender
 - abl TextBoxEmail of TextBox
 - abl TextBoxLoginName of TextBox
 - Instance Members
 - Properties inherited

return ?
TextBoxLoginName of TextBox.Text
TextBoxEmail of TextBox.Text

Object picker

Back Next Exit Add

- Tag: Object
- Text: String
- TextAlign: HorizontalAlignment
- TextLength: Int32

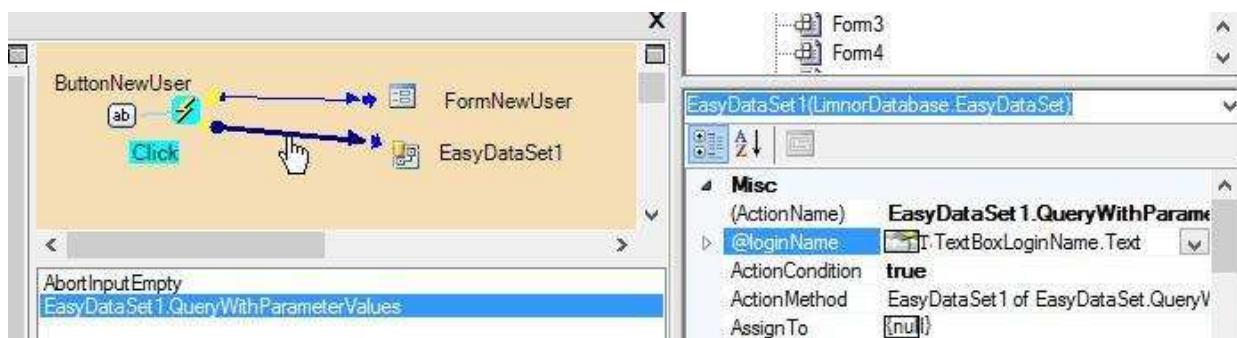
return ?
TextBoxLoginName of TextBox.Text
TextBoxEmail of TextBox.Text

Action Properties

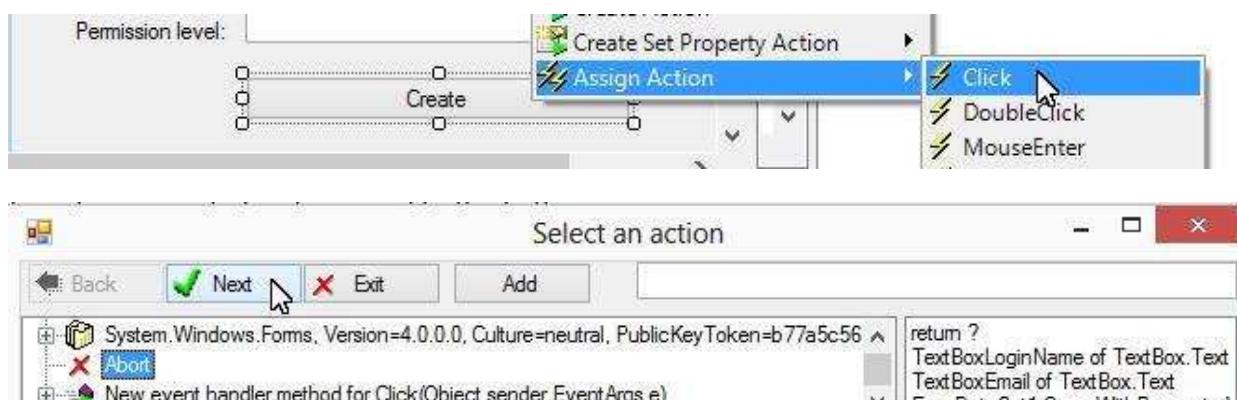
Back OK Cancel

(ActionName)	EasyDataSet1.QueryWithParameterValues
Description	
ReturnValueType	Boolean
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	EasyDataSet1 of EasyDataSet.QueryWithParameterValues()
@loginName	TextBoxLoginName.Text
AssignTo	{null}

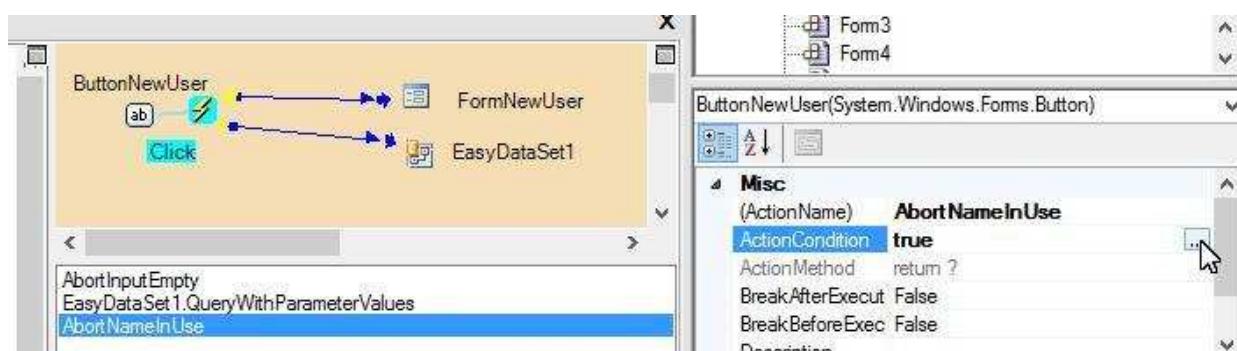
The action is created and assigned to the button:



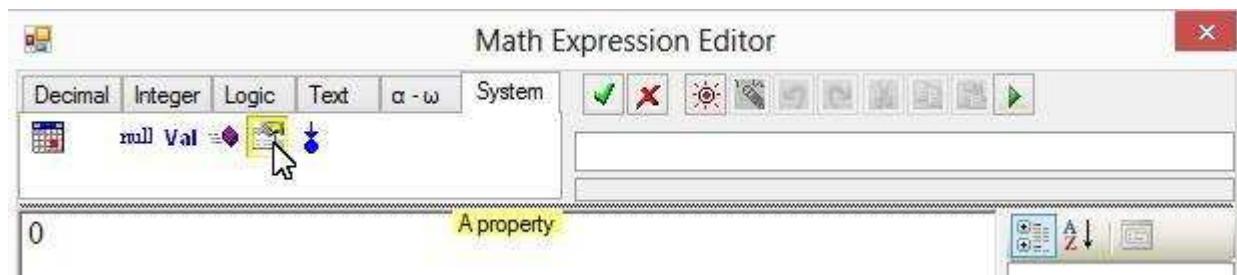
Assign an abort action after searching the database:

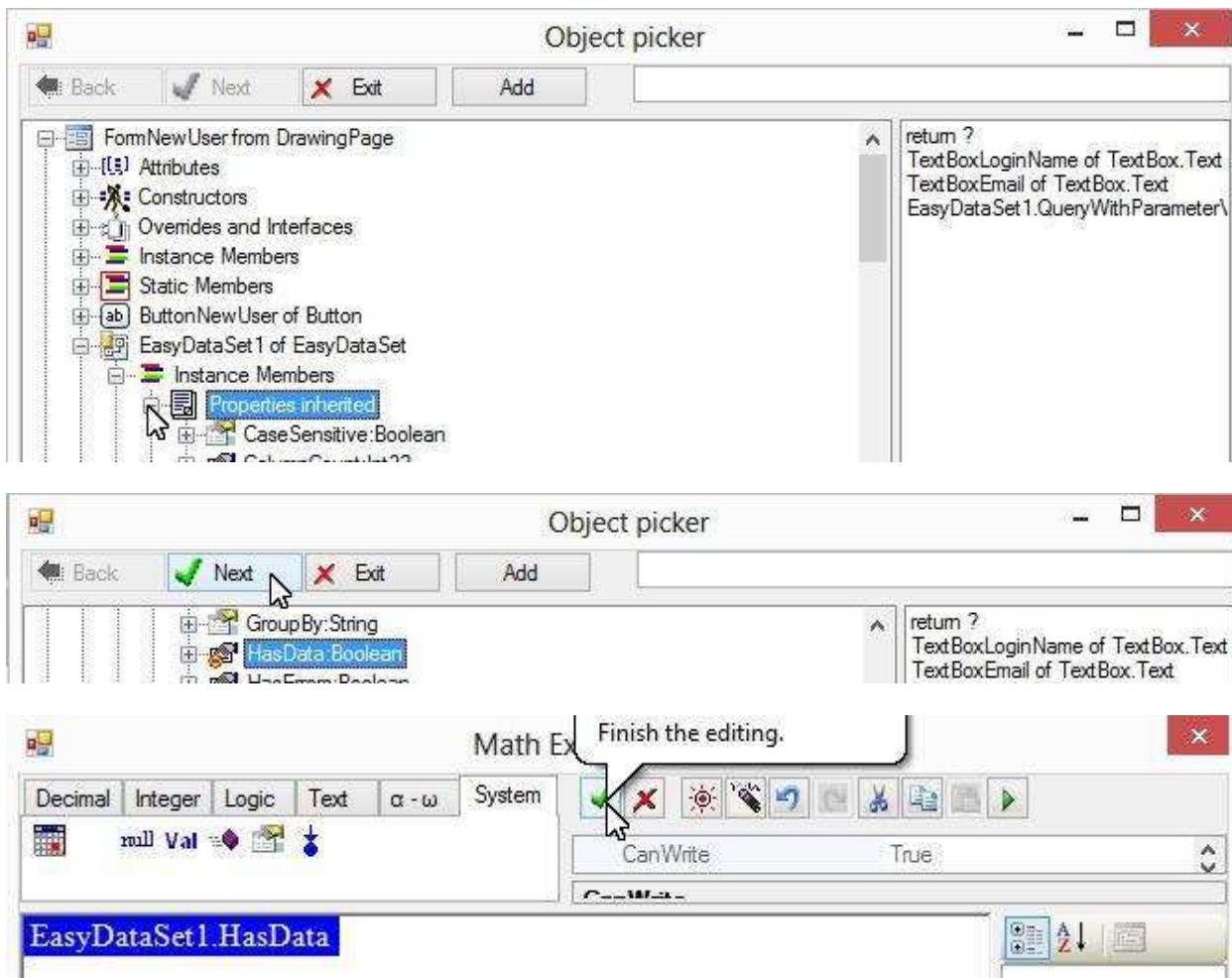


An abort action is assigned to the button. Rename it and set its ActionCondition:

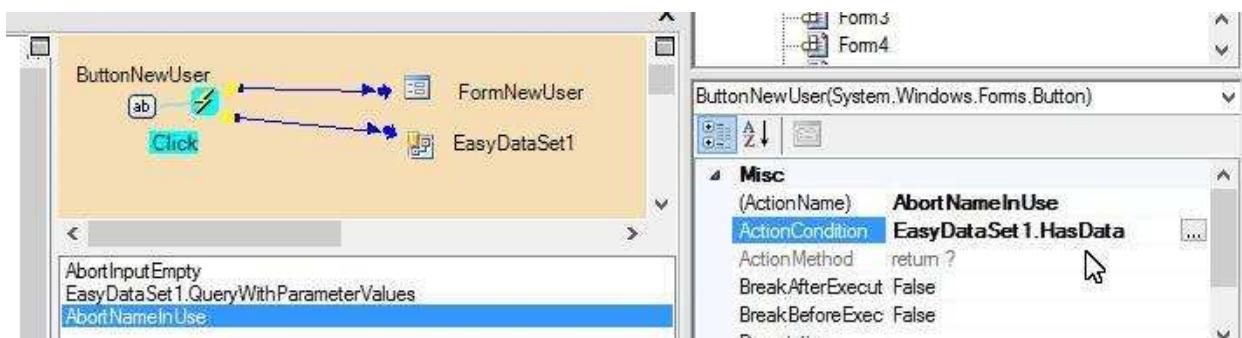


Select HasData property of the EasyDataSet:



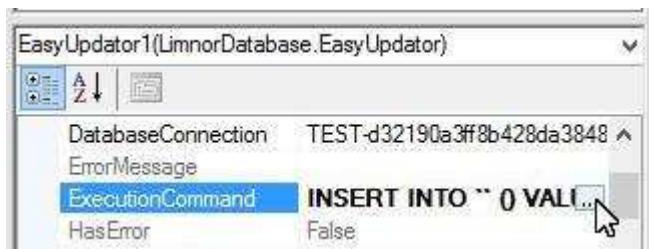


The login name usage checking is done:



Create new user record

The ExecuteCommand property of the EasyUpdater 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:

Command type:

Insert

Select fields to be changed:

FieldName	Include	Value
UserID	<input type="checkbox"/>	
UserAlias	<input checked="" type="checkbox"/>	@loginName
UserPassword	<input type="checkbox"/>	
Salt	<input type="checkbox"/>	
UserLevel	<input checked="" type="checkbox"/>	@level
ResetCode	<input type="checkbox"/>	

Database command (SQL statement):

```
INSERT INTO `useraccount` ('UserAlias', 'UserLevel', 'FirstName', 'LastName', 'Email')
VALUES (@loginName, @level, @firstname, @lastname, @email)
```

Table name:

product
productcategory
service
useraccount

Value to be set to the field:

@loginName

Filter (FROM/WHERE clause):

Use single quotes to include string literals in the value the filter, for example, 'English'

Database command (SQL statement):

```
INSERT INTO `useraccount` ('UserAlias', 'UserLevel', 'FirstName', 'LastName', 'Email')
VALUES (@loginName, @level, @firstname, @lastname, @email)
```

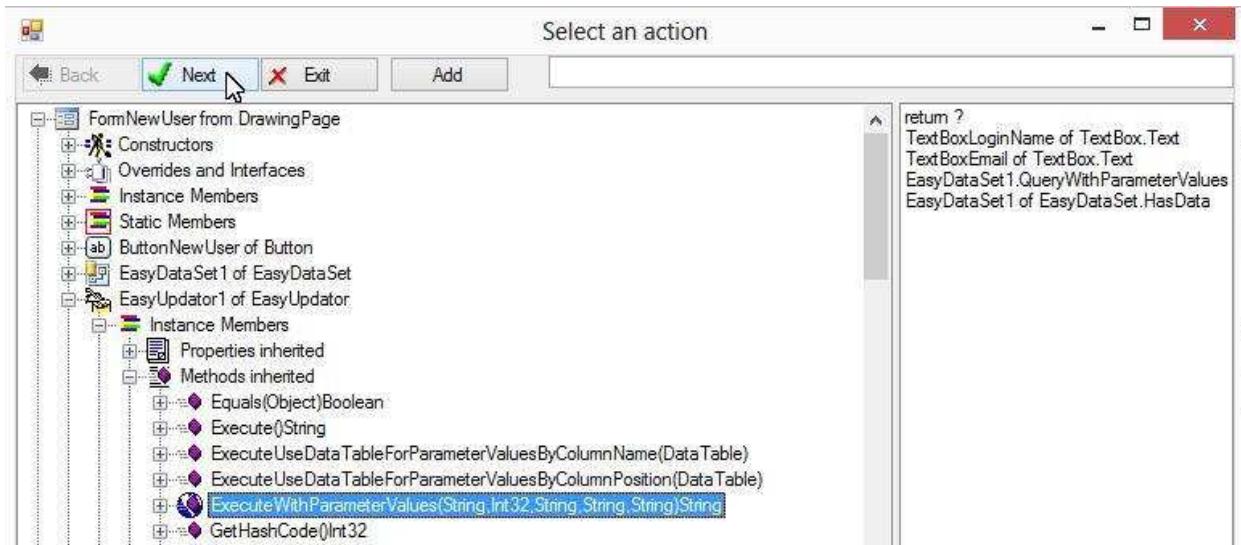
Use single quotes to include string literals in the value the filter, for example, 'English'

Use @<parameter name> to include a parameter in the value or filter. For example, @Notes.

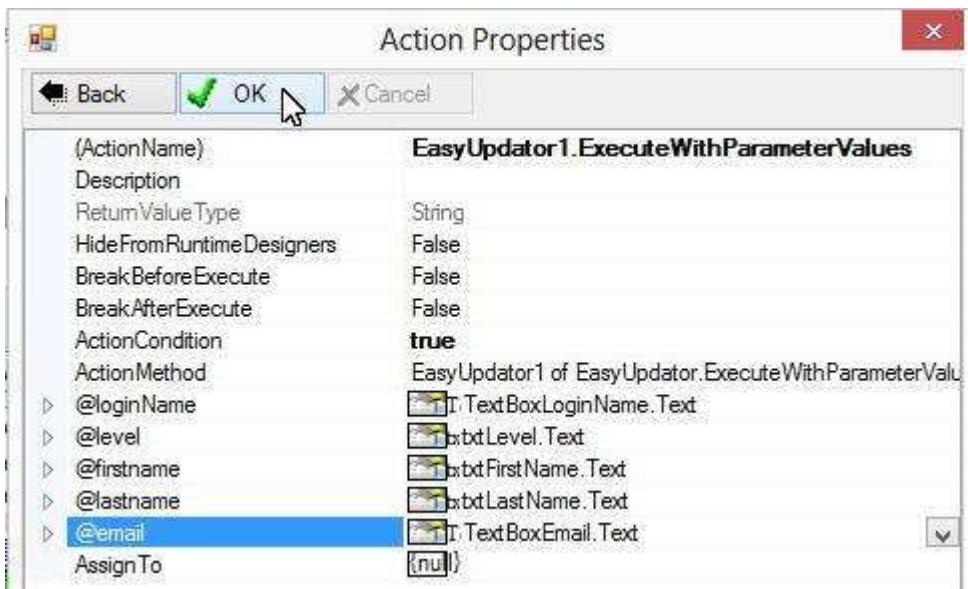
OK **Cancel**

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

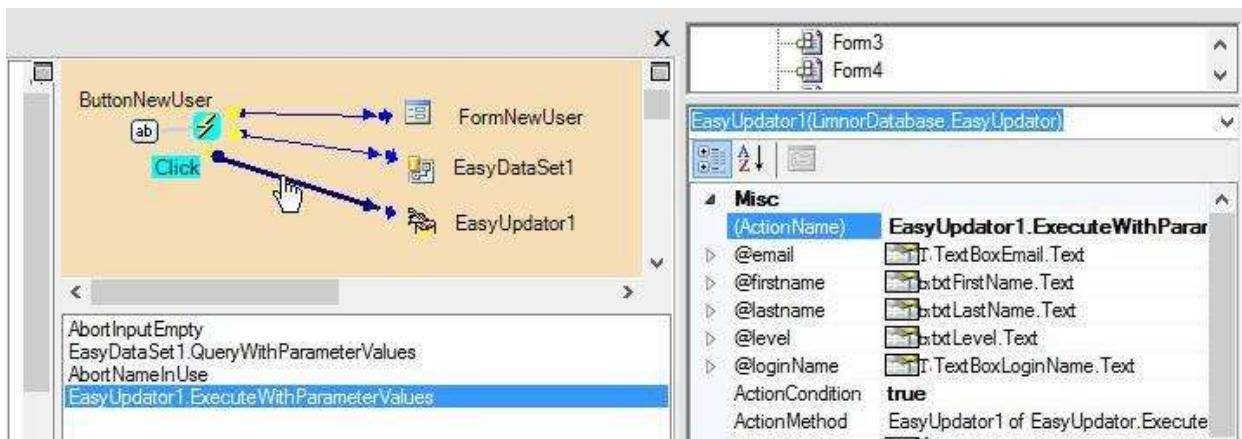




Pass Text property of the text boxes to the action parameters:

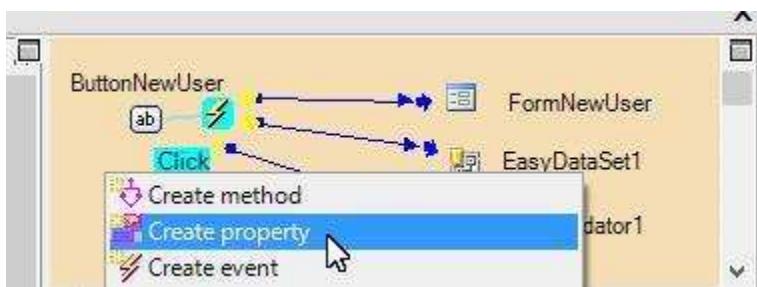


The action is created and assigned to the button:

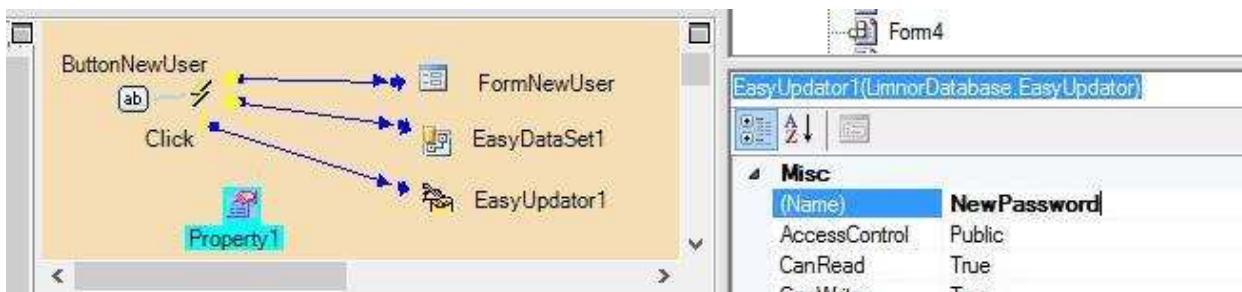


Create new password

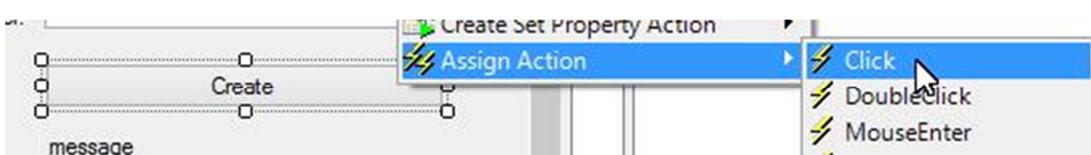
Create a property to hold the new password so that we may send it to the new user:



Rename it to “NewPassword”:

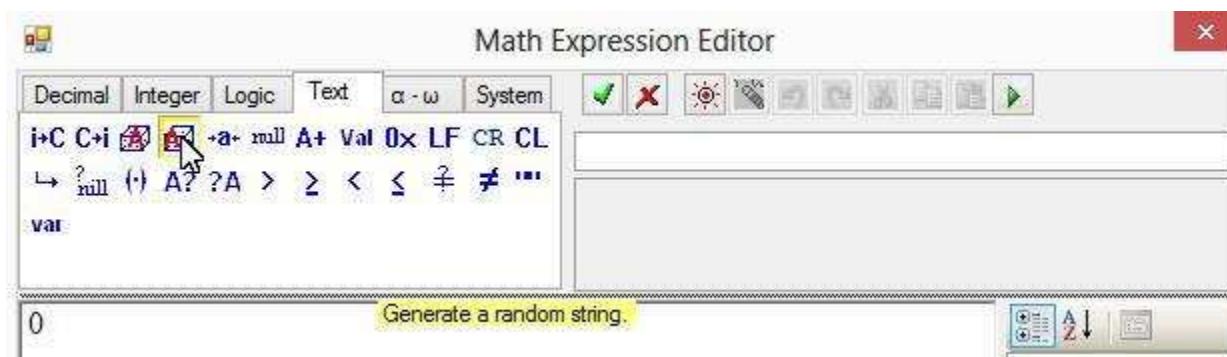
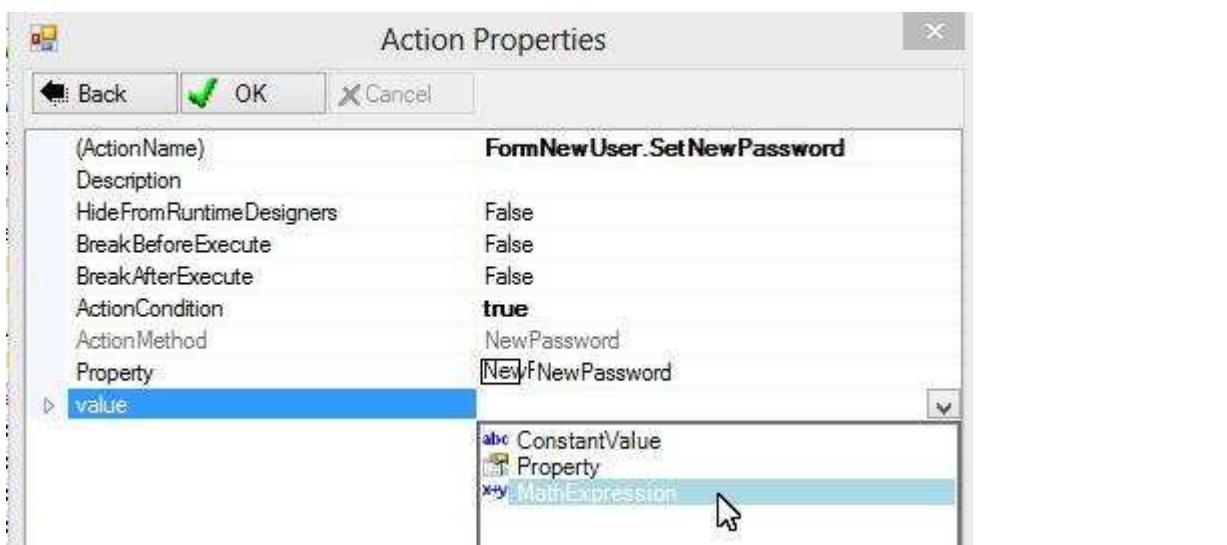


Set the property to a random string of 8 characters:

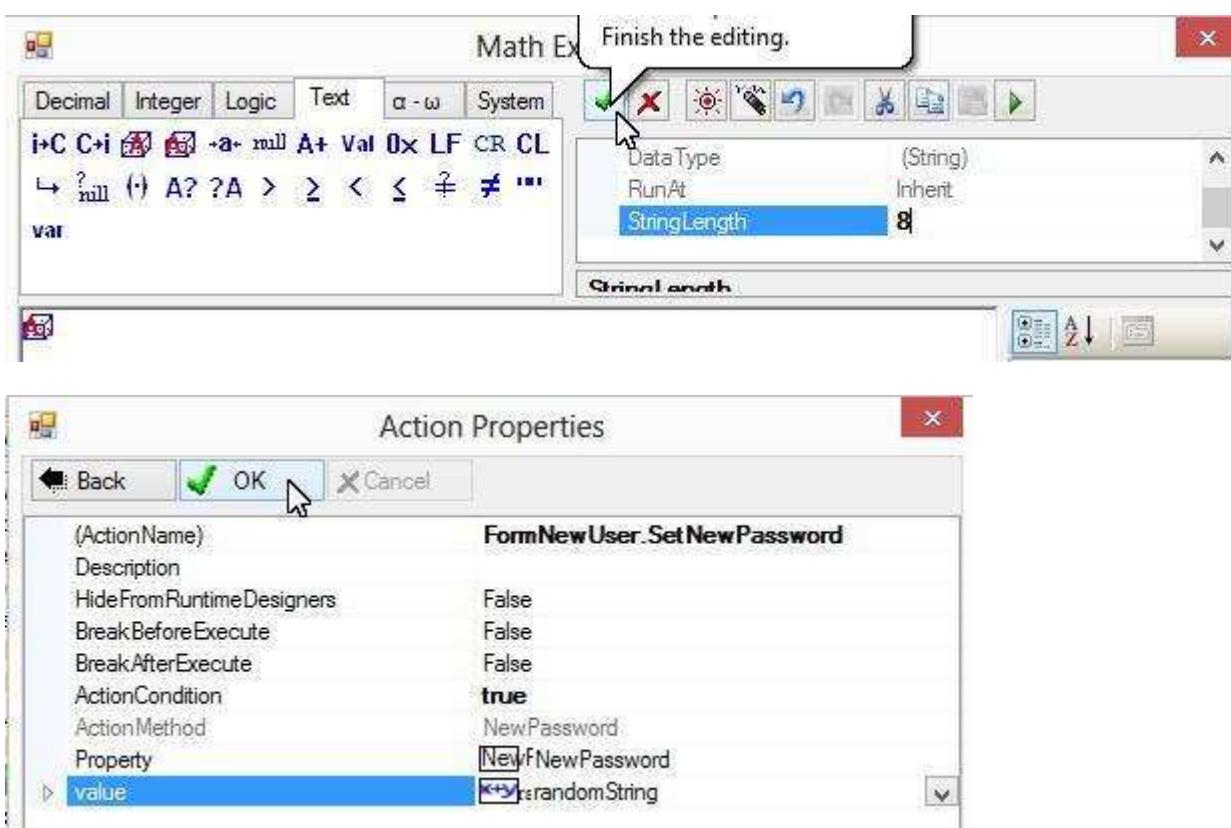




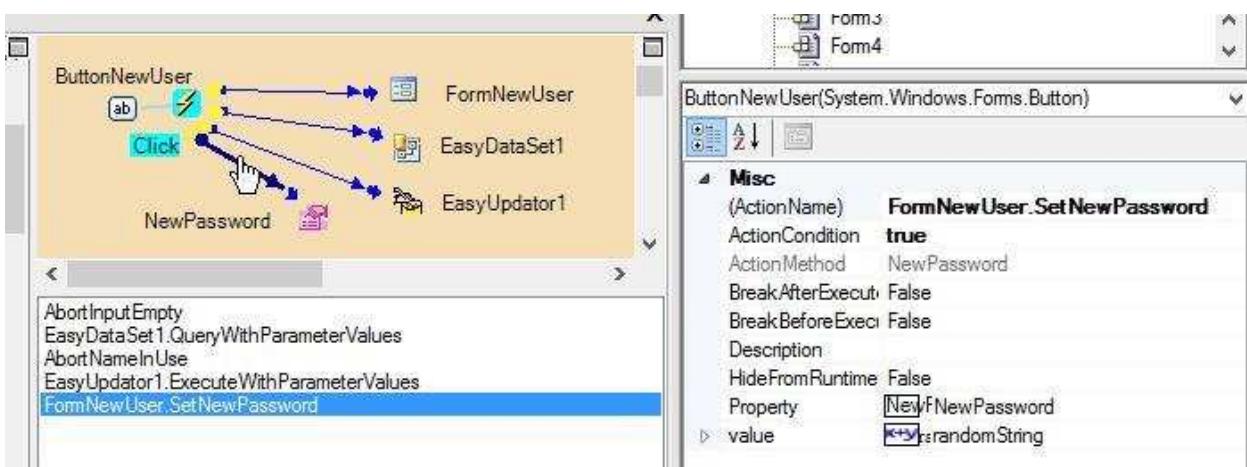
Set the value to a random string:



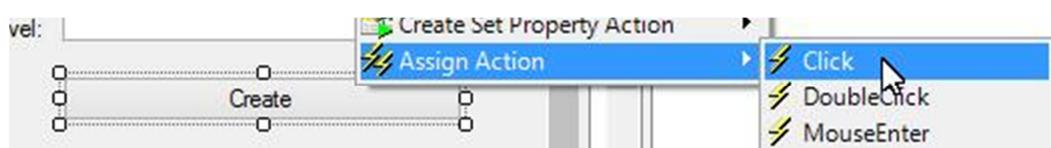
Set length to 8:



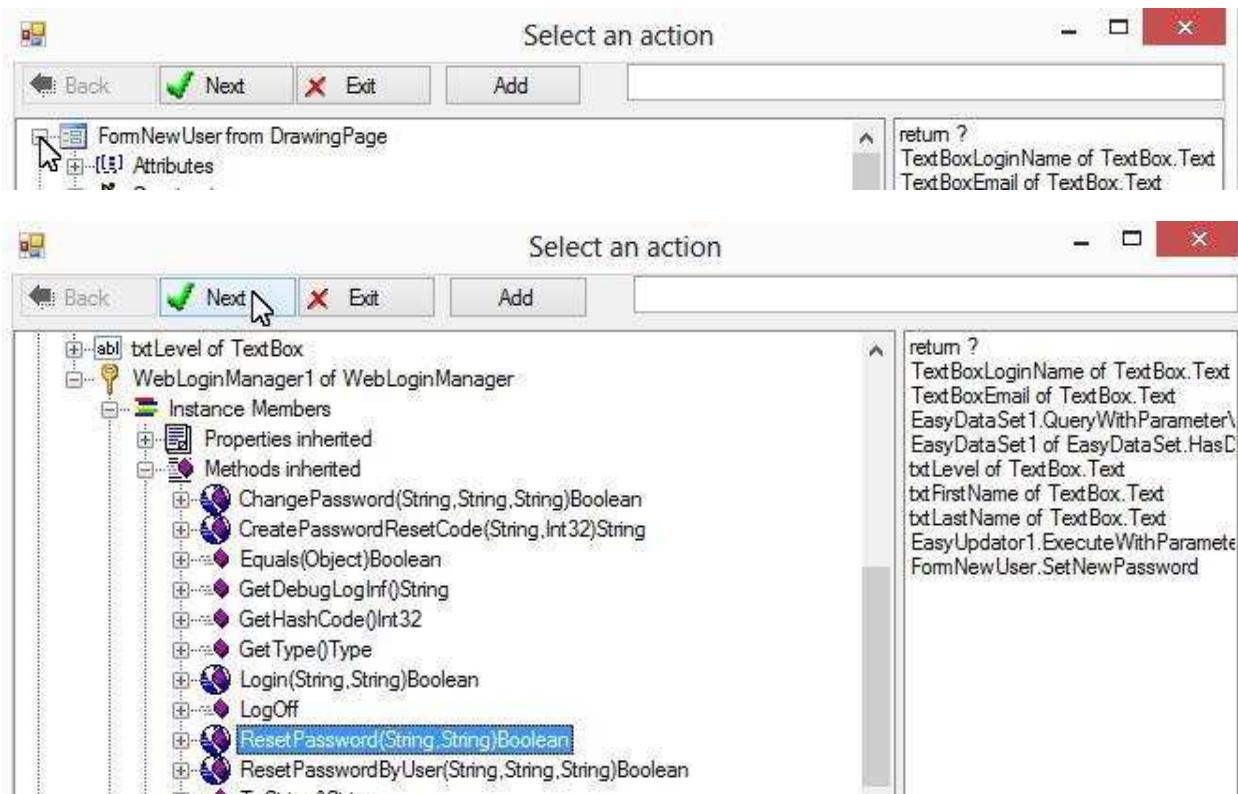
The action is created and assigned to the button:



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

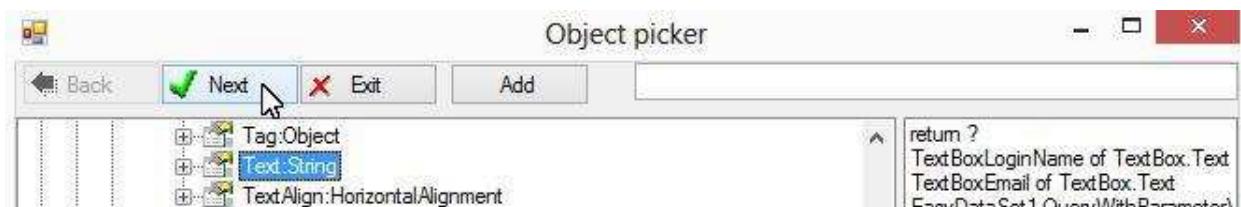


Select ResetPassword method of the Login Manager component:

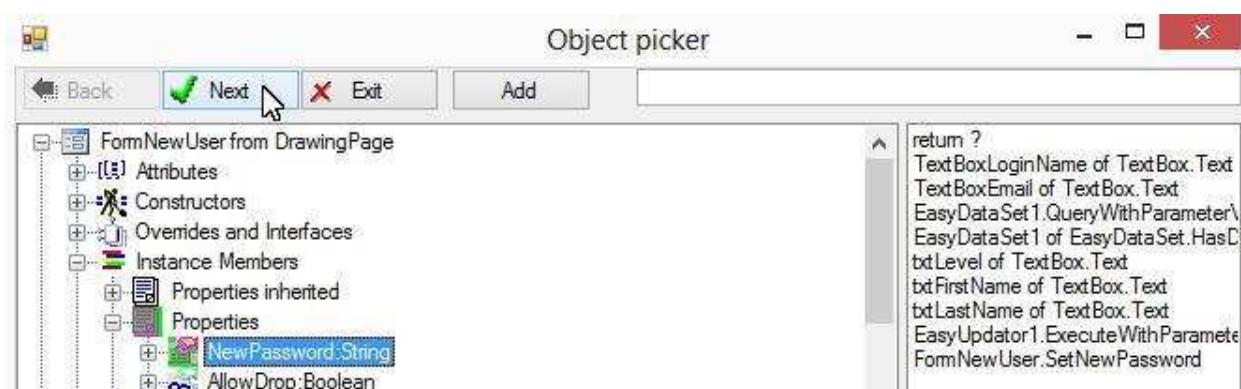
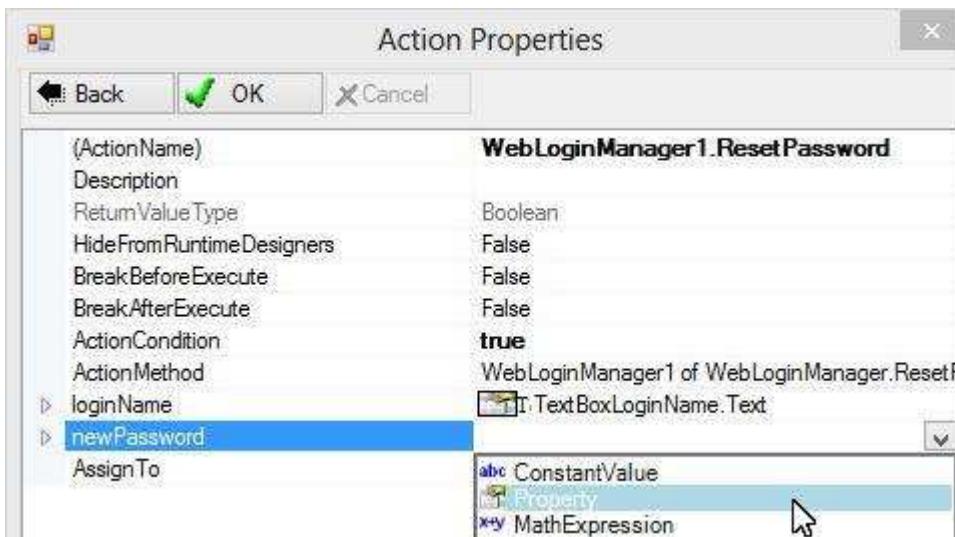


Pass the Text property of the login name text box to the action:

(ActionName)	WebLoginManager1.ResetPassword
Description	
ReturnValueType	Boolean
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	WebLoginManager1 of WebLoginManager.ResetP
loginName	<input type="button" value="ConstantValue"/> <input type="button" value="Property"/> <input type="button" value="MathExpression"/>
newPassword	
AssignTo	



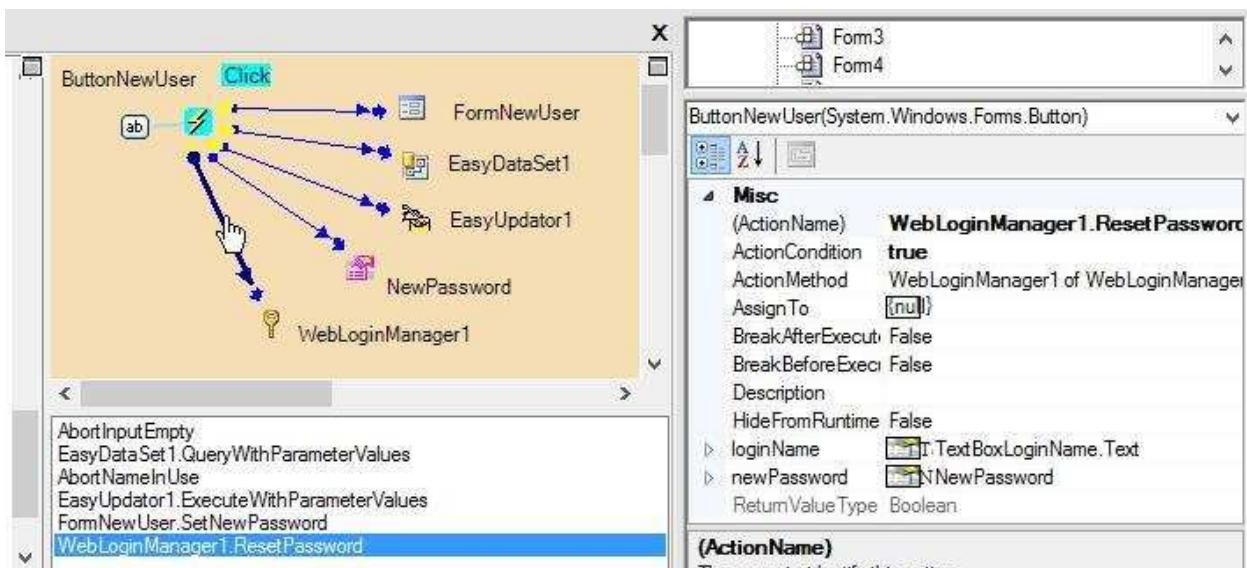
Pass the NewPassword property to the action:



Click OK to create the action:



The action is created and assigned to the button:



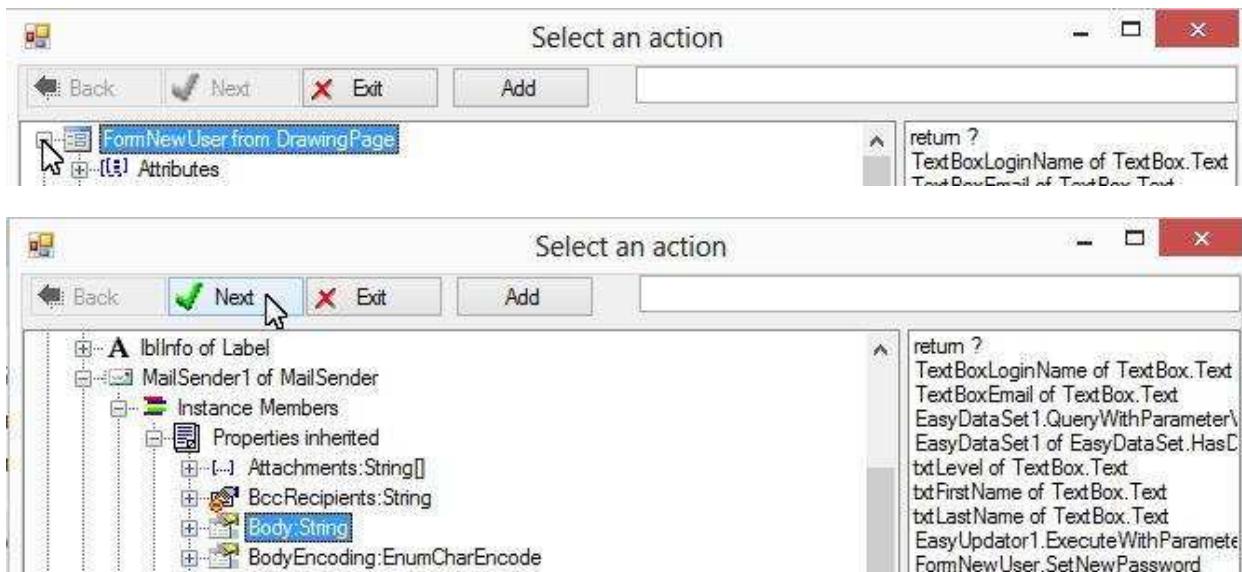
Note that you should never let an end user directly execute a ResetPassowrd action!

Send password to user

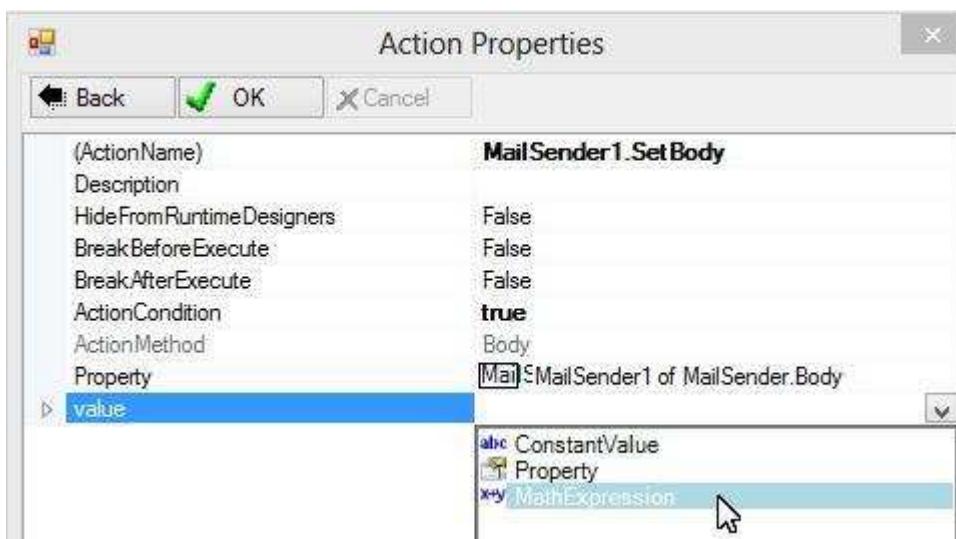
In this sample, we use a MailSender component to send the password to the user. Set the email body to include the new password:



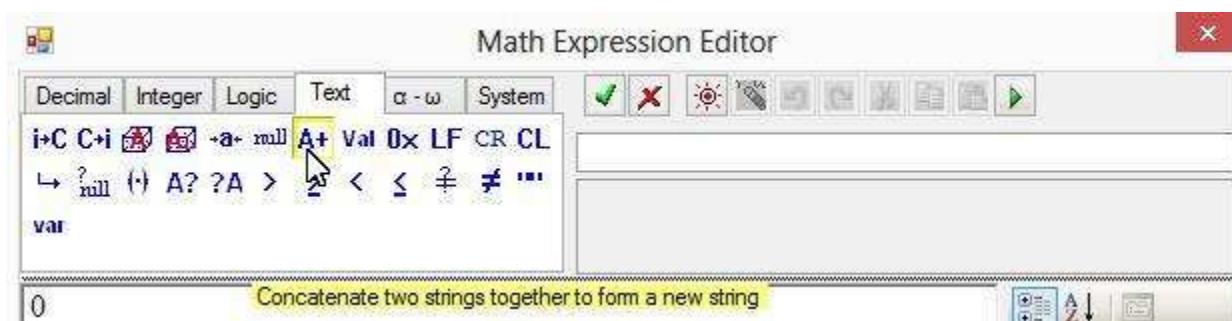
Login Manager Windows Form Sample – Part 1 | 2013



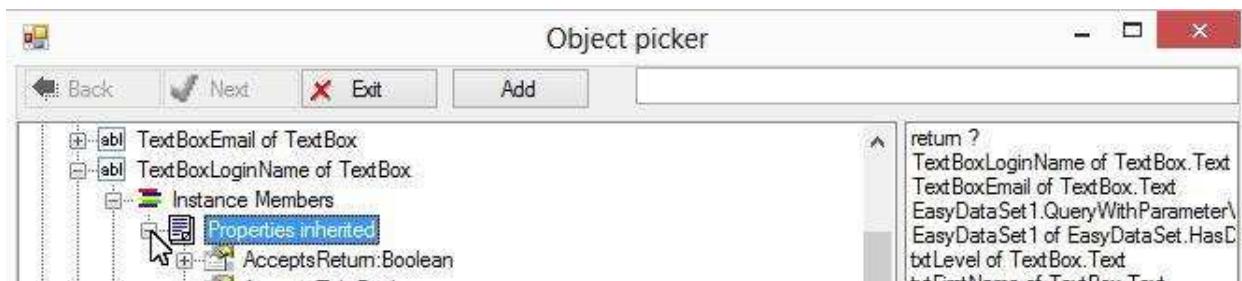
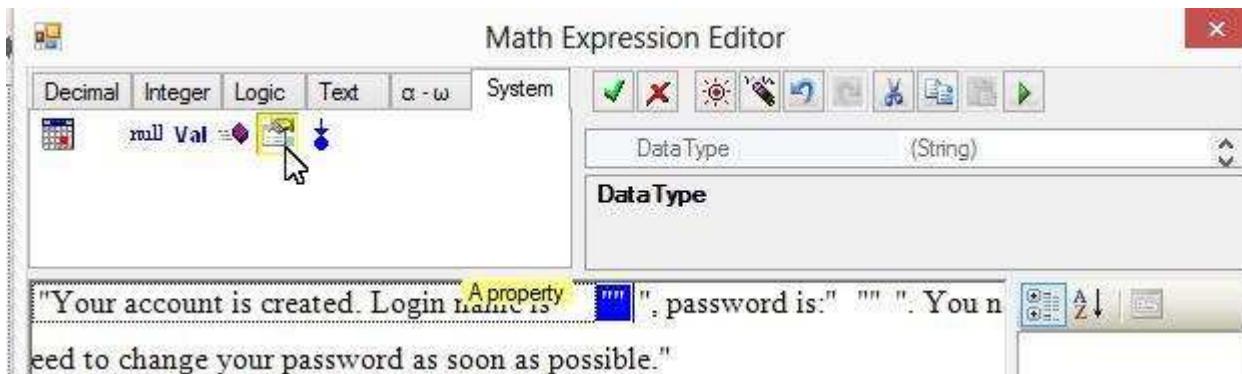
Use an expression to form email body:



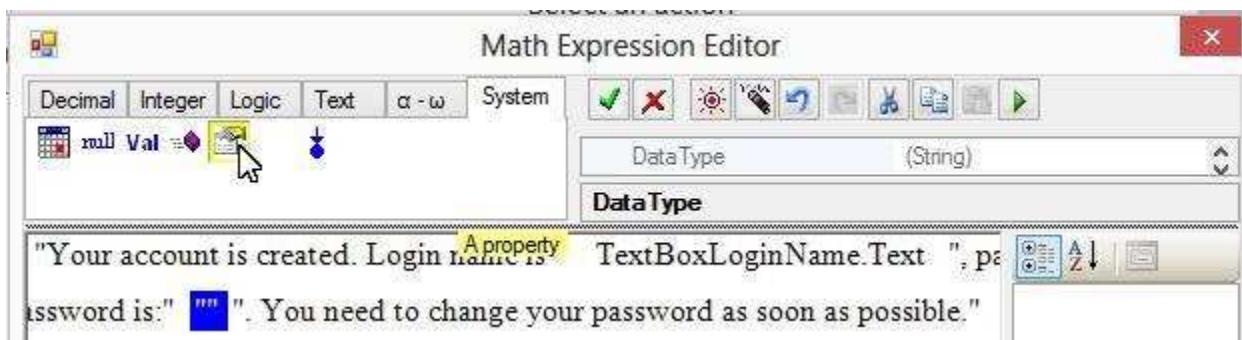
Click A+ a few times to create 5 parts of strings:



Select login name:



Select the new password into the expression:



Object picker

FormNewUser from DrawingPage

- Attributes
- Constructors
- Overrides and Interfaces
- Instance Members
- Properties inherited
- Properties
- NewPassword String
- AllowDrop:Boolean

return ?
 TextBoxLoginName of TextBox.Text
 TextBoxEmail of TextBox.Text
 EasyDataSet1.QueryWithParameter
 EasyDataSet1 of EasyDataSet.HasD
 btLevel of TextBox.Text
 txtFirstName of TextBox.Text
 txtLastName of TextBox.Text
 EasyUpdator1.ExecuteWithParamete
 FormNewUser.SetNewPassword
 NewPassword

Math Editor

Finish the editing.

Decimal Integer Logic Text α - ω System

Value: Double

Data Type

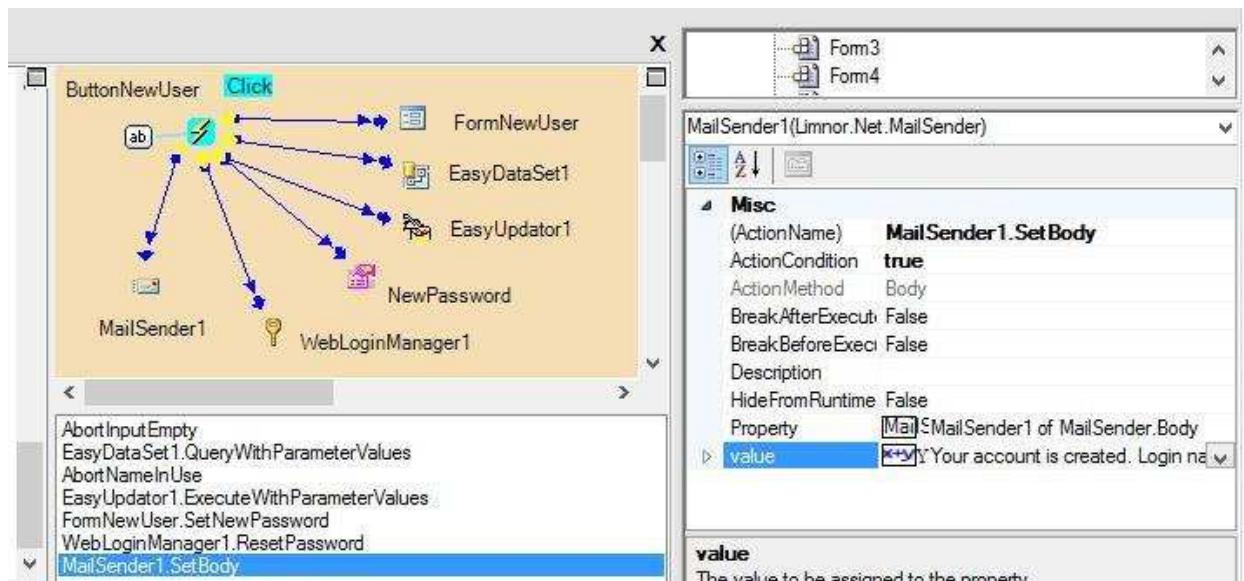
"Your account is created. Login name is " TextBoxLoginName.Text ", password is:" NewPassword ". You need to change your password as soon as possible."

Action Properties

Action Name: MailSender1.SetBody

Description: HideFromRuntime: False
 BreakBeforeExecute: False
 BreakAfterExecute: False
 ActionCondition: true
 ActionMethod: Body
 Property: MailSEmailSender1 of MailSender.Body
 value: Your account is created. Login name is TextBoxLoginName.Text, password is:NewPassword. You need to change your password as soon as possible.

The action is created and assigned to the button:



Execute a Send action to send the email:

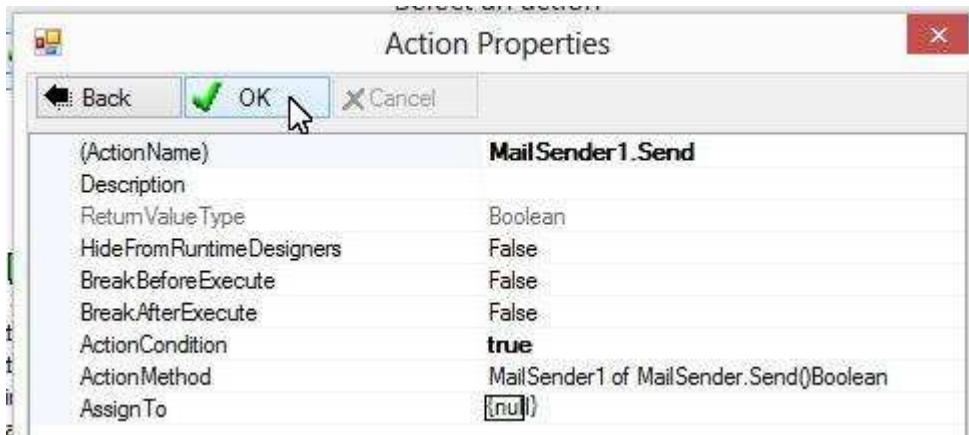
Assign Action

Click

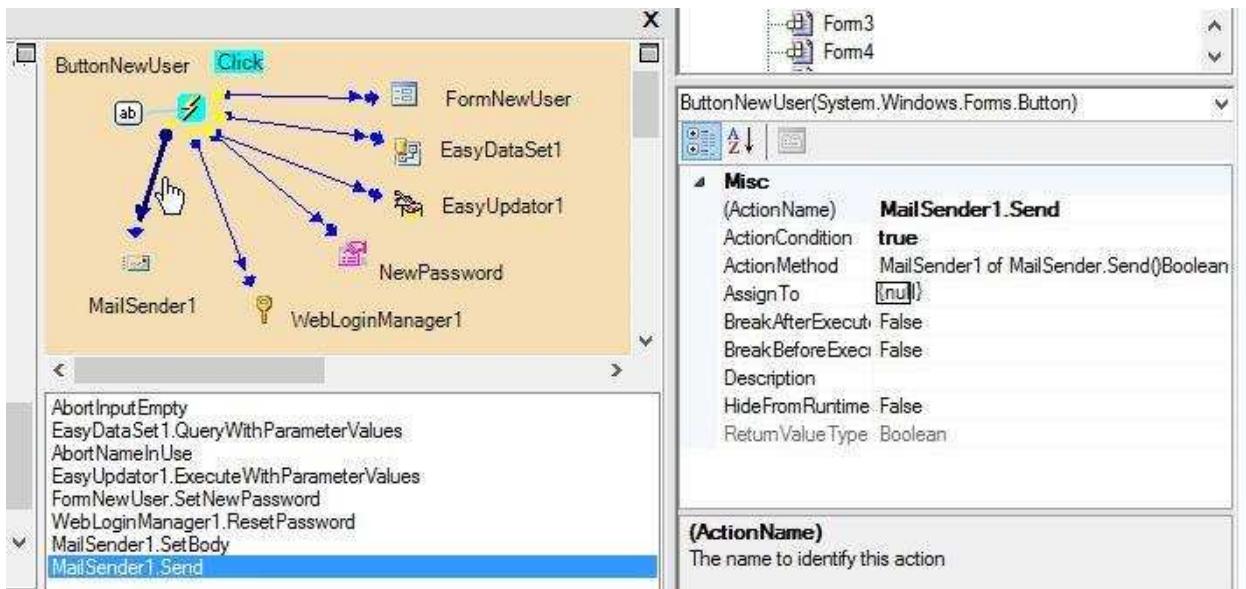
MailSender1

AddAttachment(String)

Send()



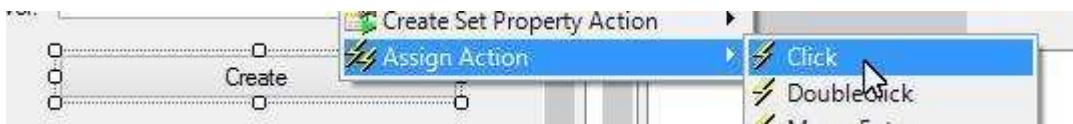
The action is created and assigned to the button:



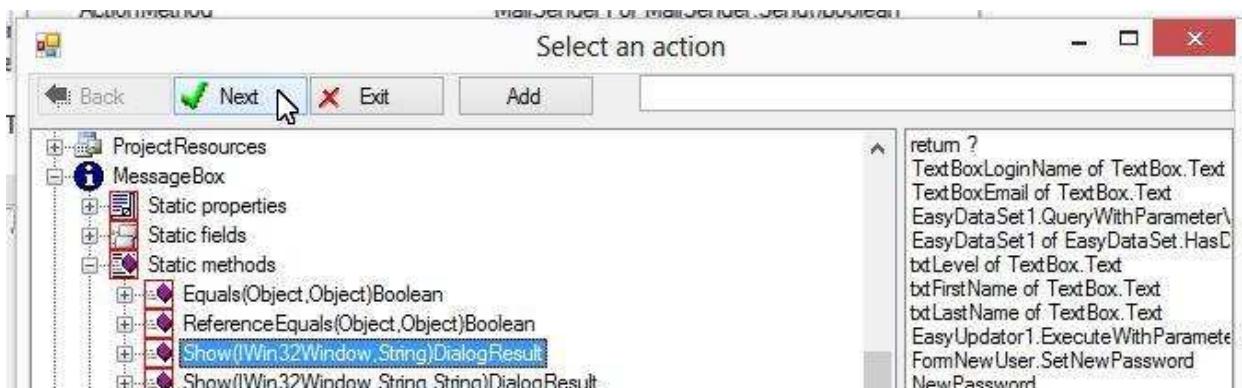
We did not show how to set properties of the MailSender.

Show notification message

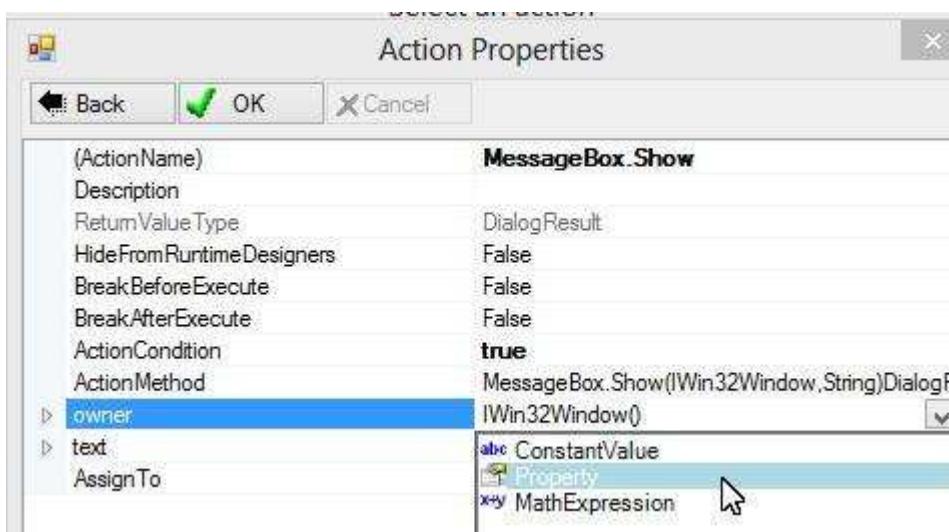
We may show a message box after all actions are executed:



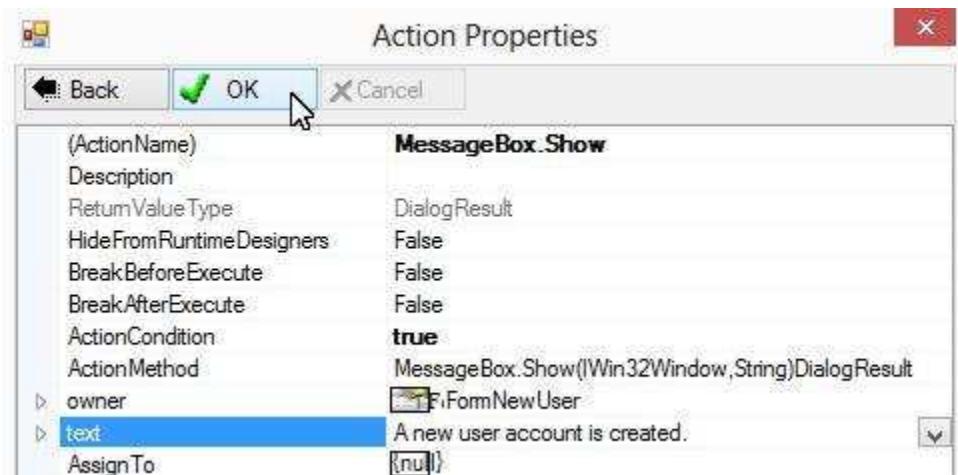
Select a Show method of the MessageBox:



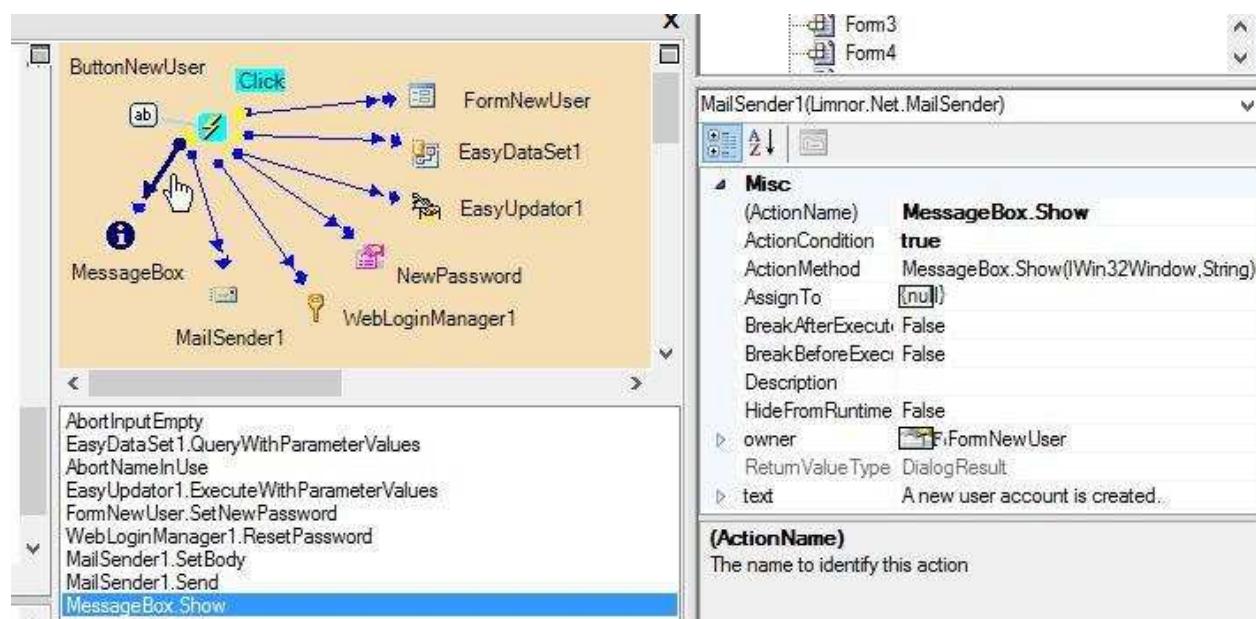
Select the current form as the message box owner:



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.



Three screenshots of a Windows application interface are shown, illustrating the configuration of a form action.

Screenshot 1: Action Selection

The window title is "Select an action". It contains buttons: Back, Next, Exit, Add, and a text input field. Below these is a tree view showing the current node: "FormNewUser from DrawingPage". Under this node, there are "Attributes" and "Constructors".

Screenshot 2: Action Properties

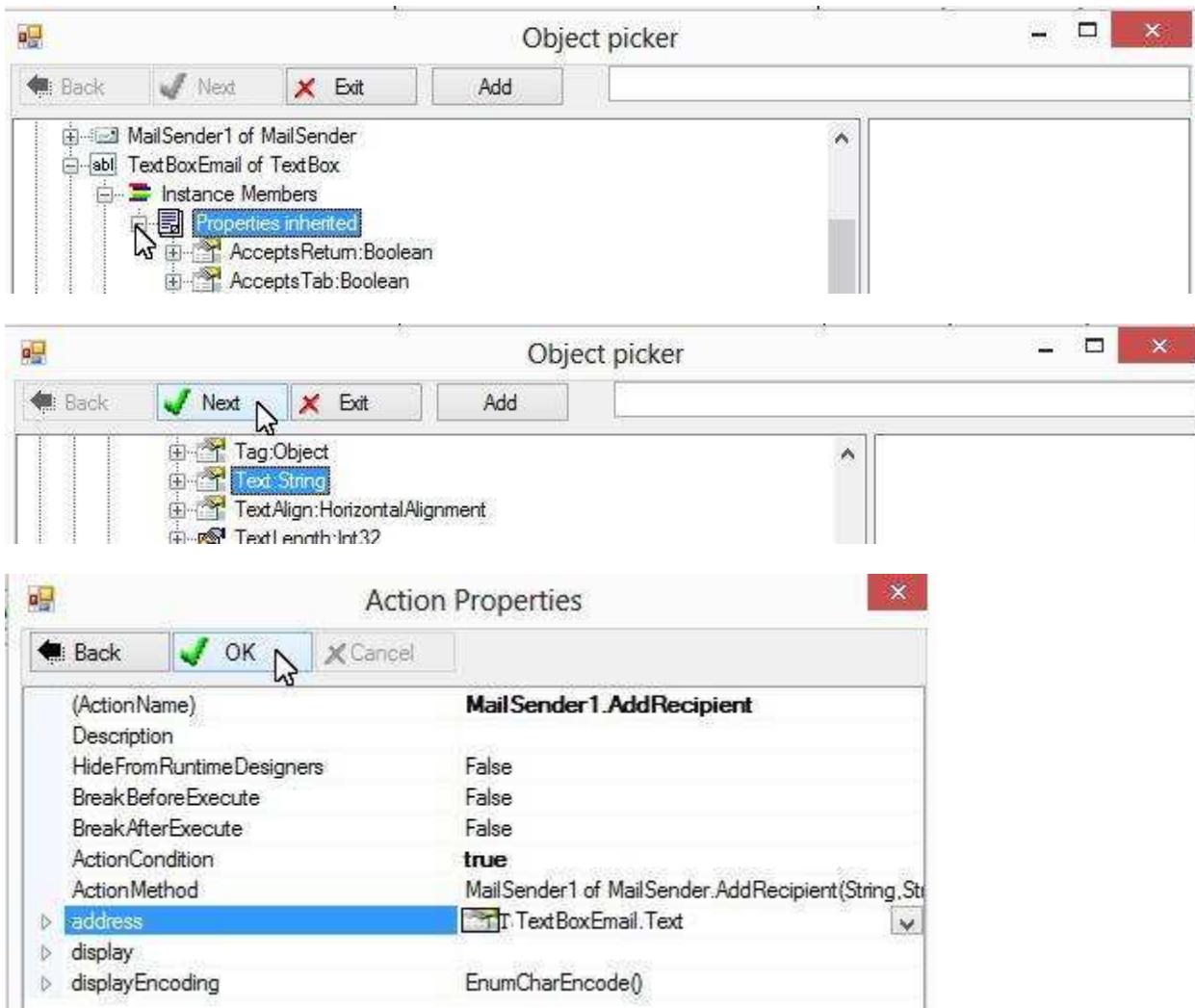
The window title is "Action Properties". It has buttons: Back, OK, Cancel. The "ActionName" field is set to "MailSender1.AddRecipient". Other properties listed include:

- Description
- HideFromRuntimeDesigners: False
- BreakBeforeExecute: False
- BreakAfterExecute: False
- ActionCondition: true
- ActionMethod: MailSender1 of MailSender.AddRecipient(String, String)

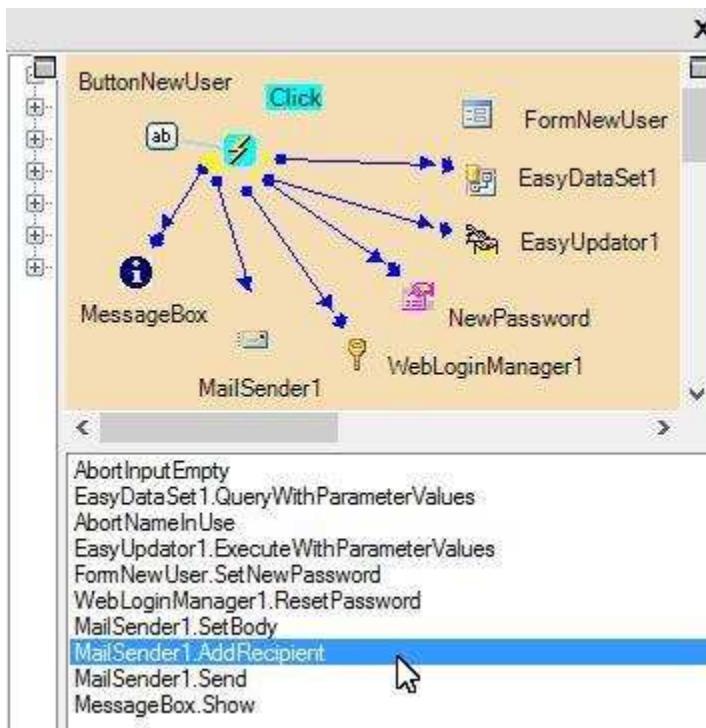
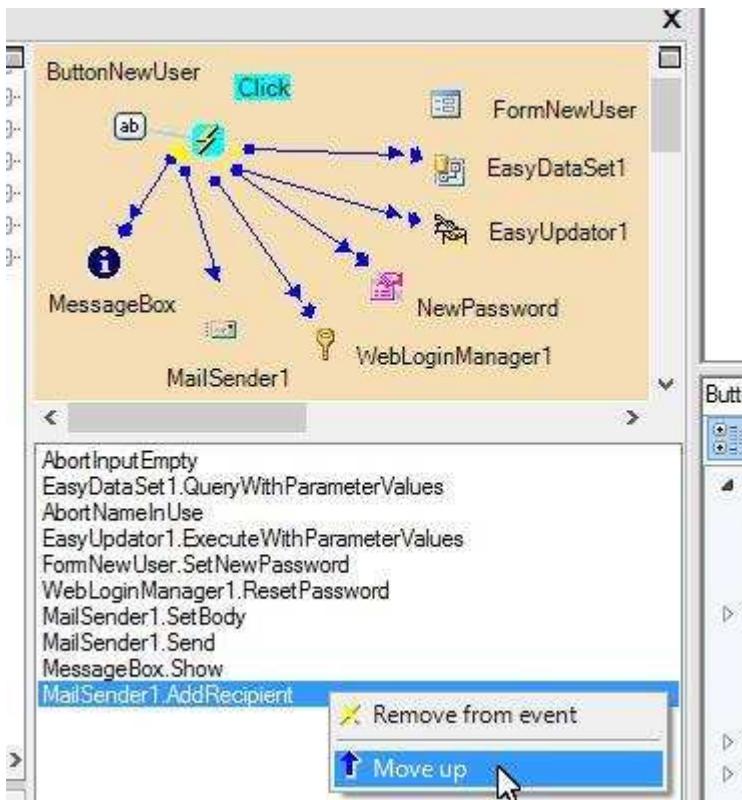
The "address" property is currently selected. A dropdown menu shows three options: Constant/Value, Property (selected), and MathExpression.

Screenshot 3: Object Picker

The window title is "Object picker". It contains buttons: Back, Next, Exit, Add. Below these is a tree view showing the current node: "FormNewUser from DrawingPage". Under this node, there are "Attributes" and "Constructors".

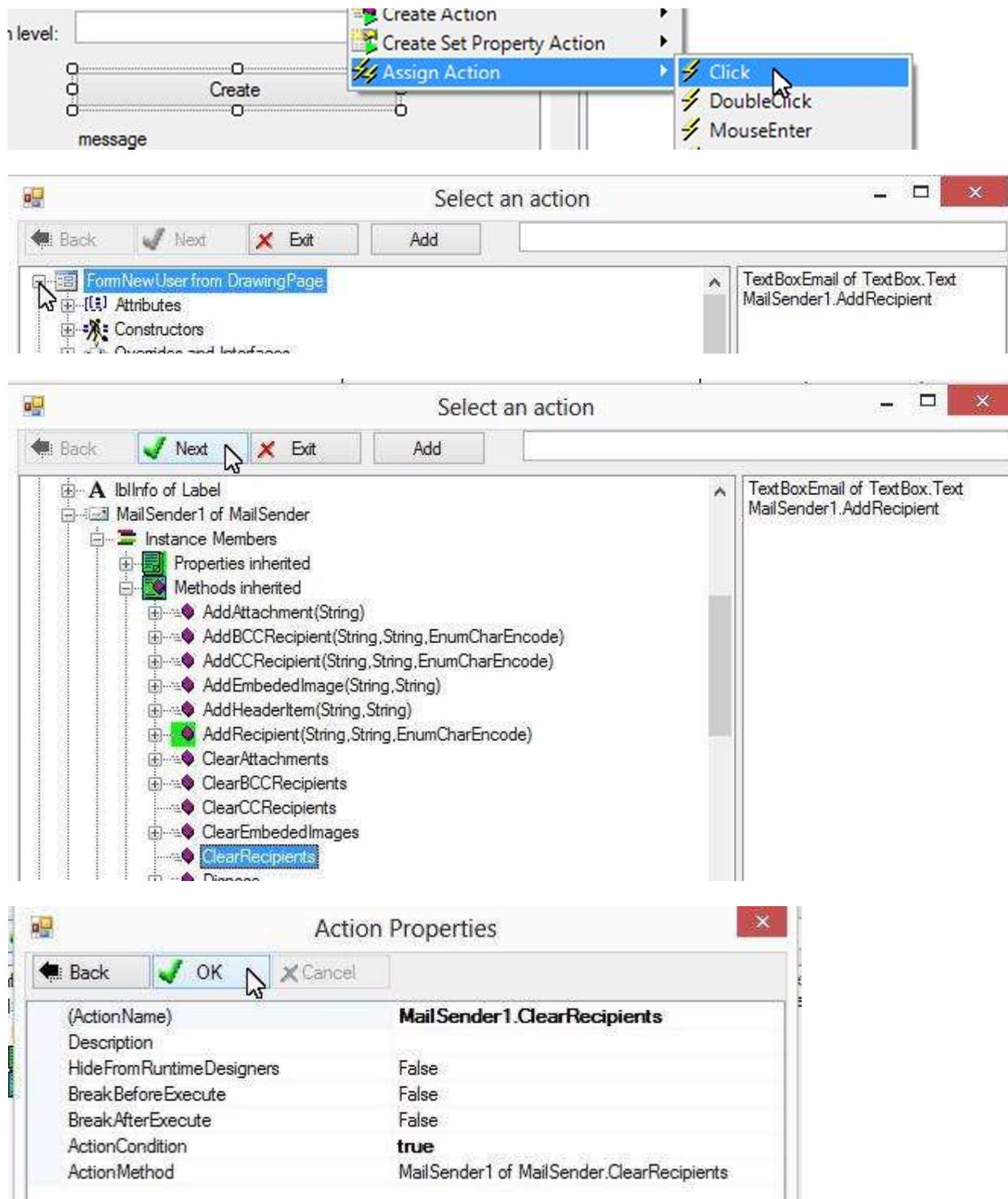


The action is created and assigned to the button. Move it up to before Send action:

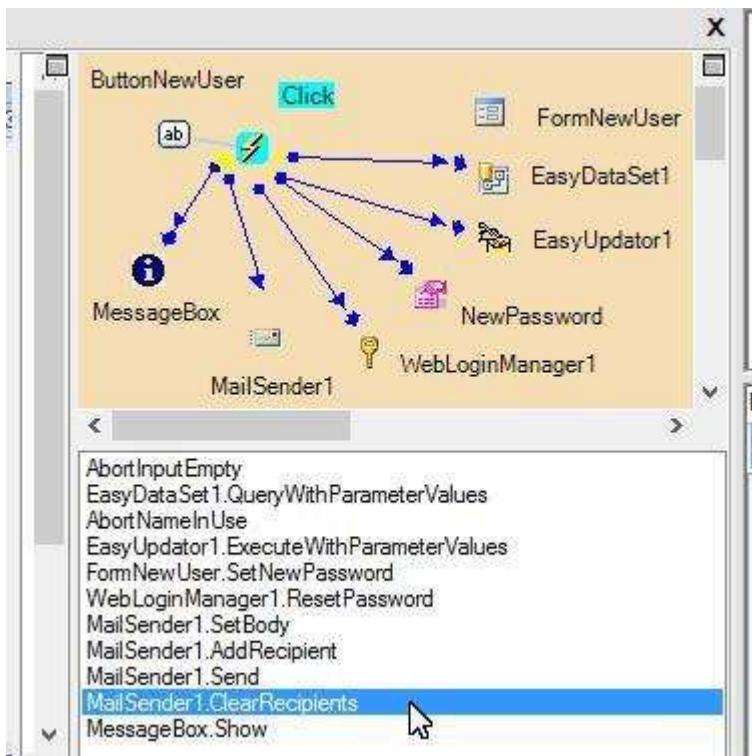
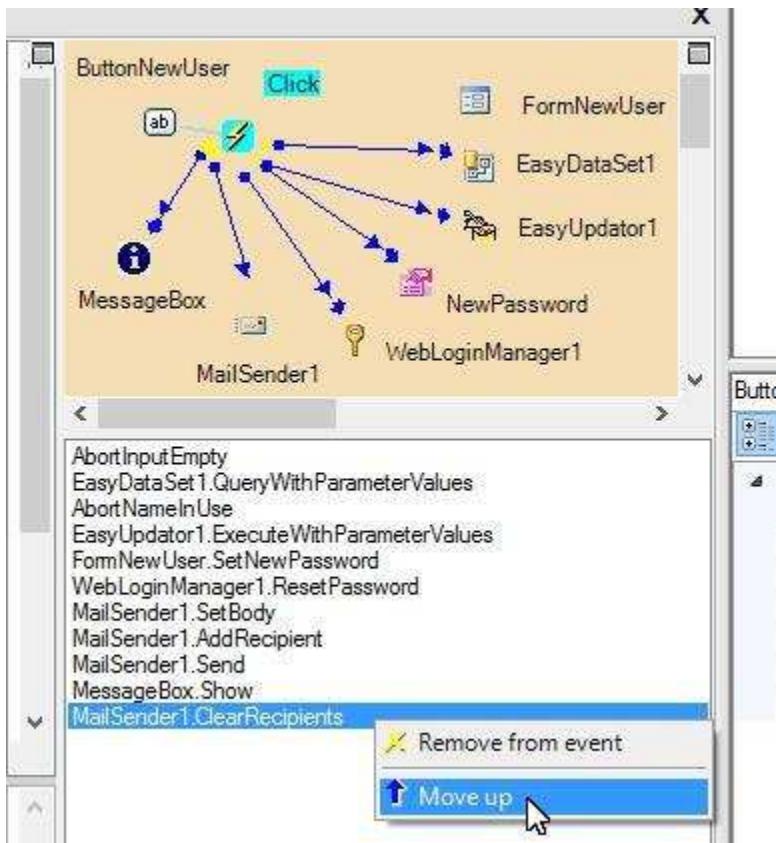


Remove email address

After sending email, the email address should be removed:



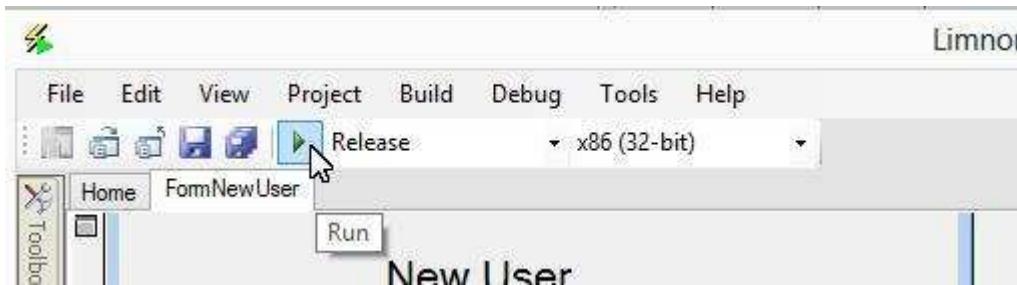
The action is created and assigned to the button. Move it up before the message box:



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

Protect new user form

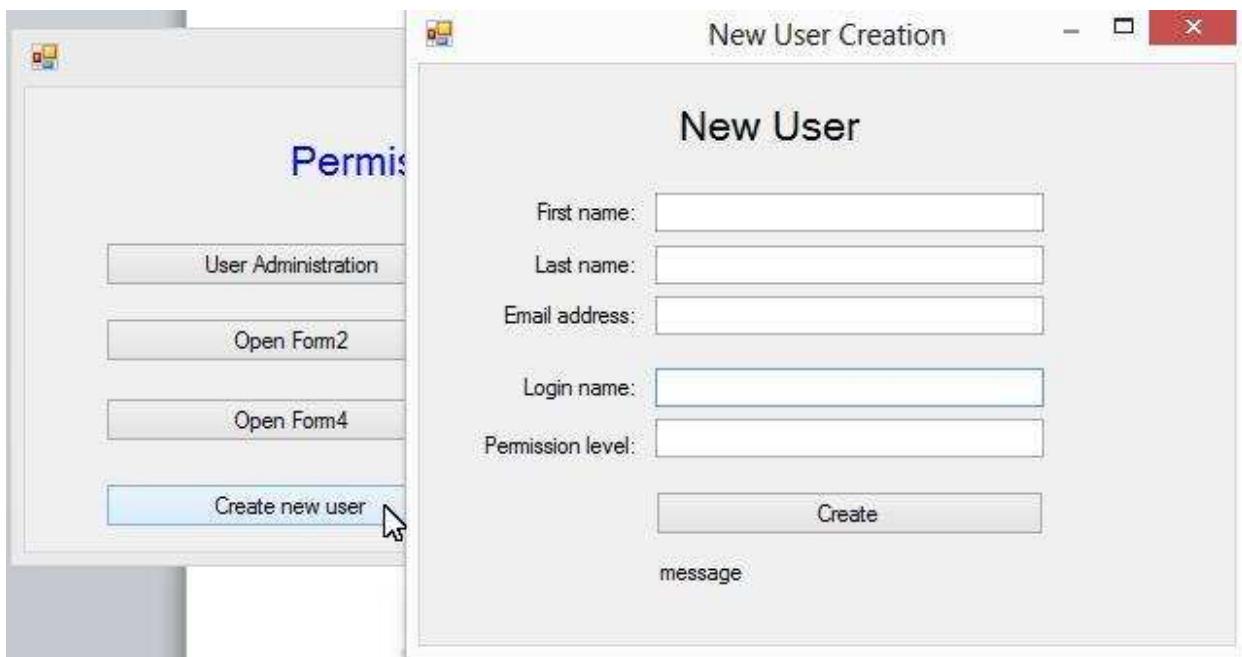
We want to add access restriction to this new user form. Before we do that, let's use it to create the first user record in the user table so that we may later log in using this first user to create other users. Run the project:



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:



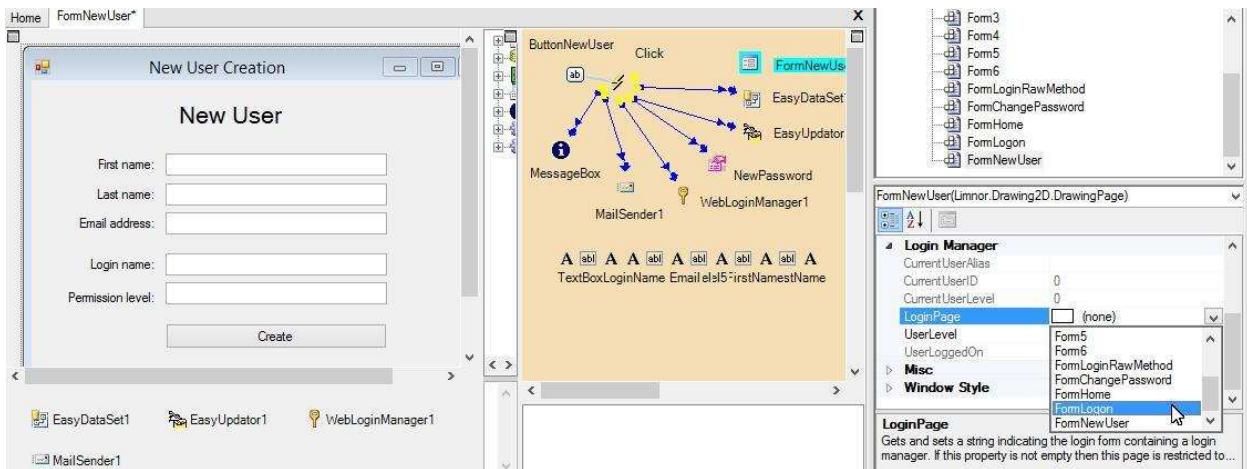
From the email message, we get the password for the new user: "Y9jjNsxGi+". The user table has its first record:

```
mysql> select * from useraccount;
+-----+-----+-----+-----+-----+-----+-----+
| UserID | UserAlias | UserPassword | ResetCode | ResetCodeExpire | FirstName | LastName | Email           | Phone | Salt
+-----+-----+-----+-----+-----+-----+-----+
|     0  | admin     | de34d88f3e3c60a63f1e20d9eb22b7634d035b675555dc42e1ee8a24be29c4b5 | NULL   | NULL          |          |          | info@mydomain.com | NULL  | EwicDT6uE8yDlqGSTtnMuLashvXuRpF2BNuB9OvbHSSg12i8vnMkO/aF4UE1bBgV
+-----+-----+-----+-----+-----+-----+-----+
1 row in set <0.00 sec
mysql> -
```

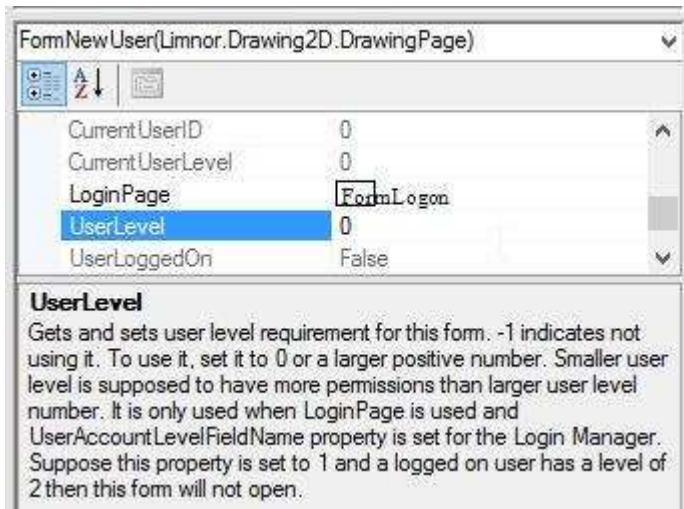
Note that the UserPassword field is

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:



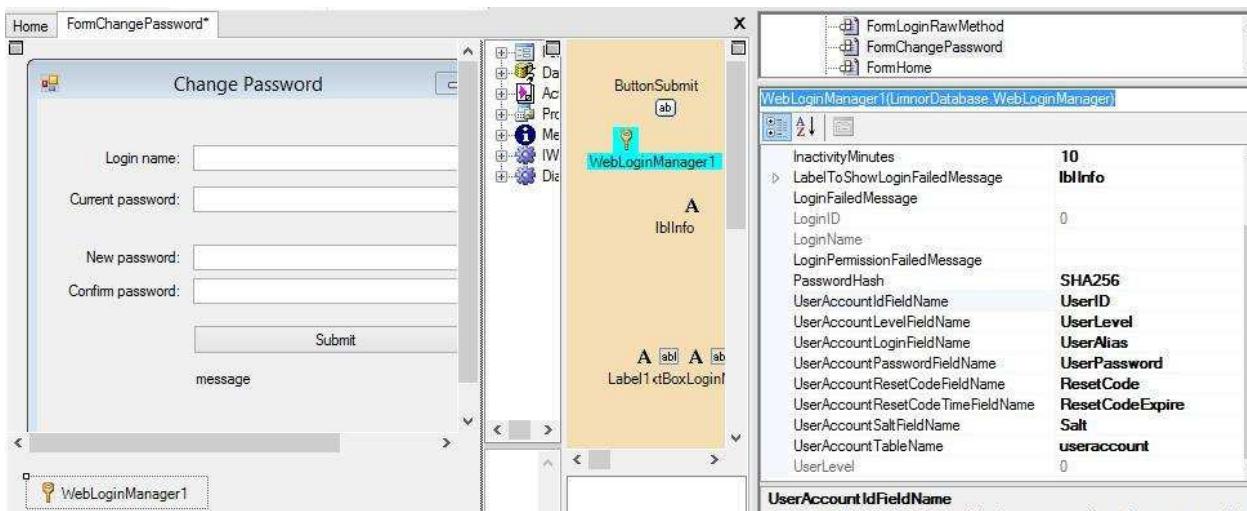
Keep the UserLevel to be 0 so that the maximum permission is required to use this form:



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

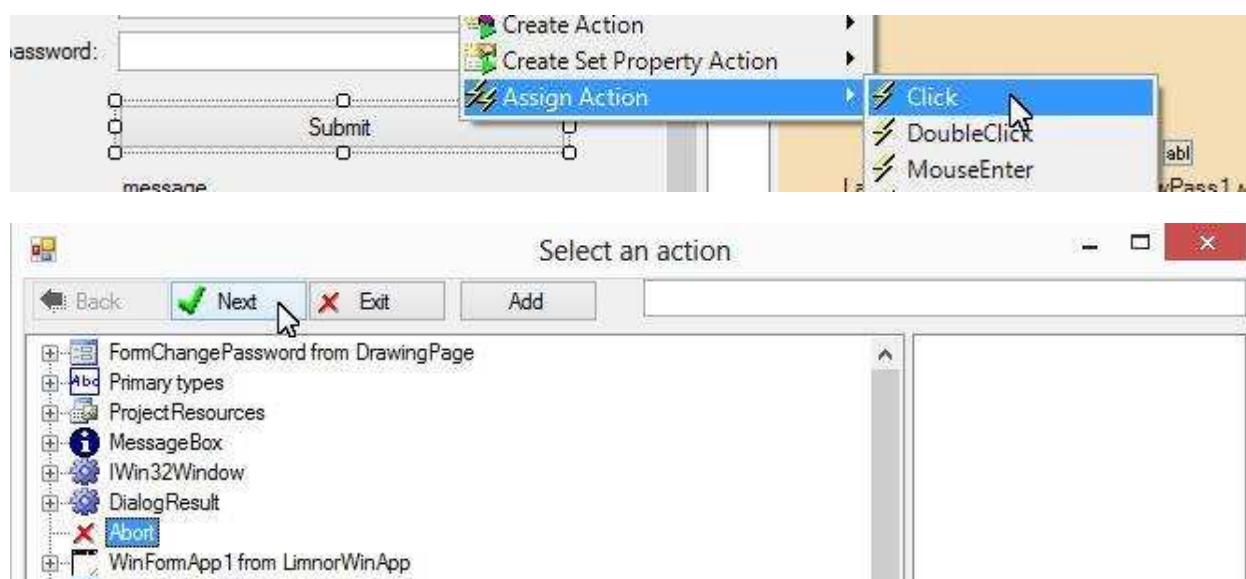
The ChangePassword method of the Login Manager Component can be used to allow a user to change his/her password. Let's use a form to give users this capability. Note that the properties of the Login Manager Component must be set the same as we did for the Login Manager component in the login form.



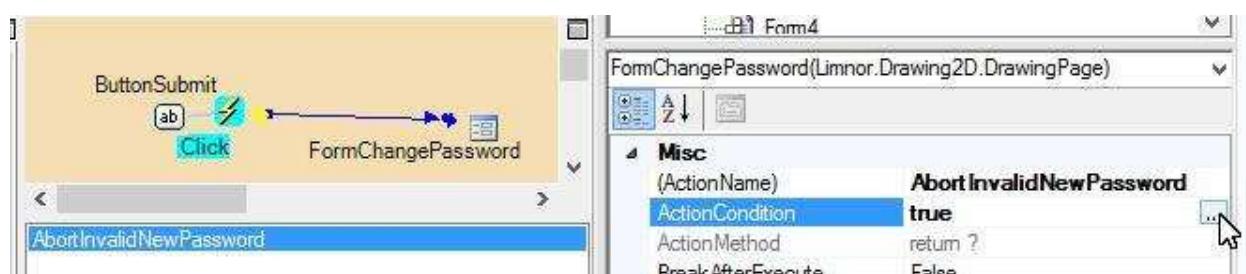
Input verification

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

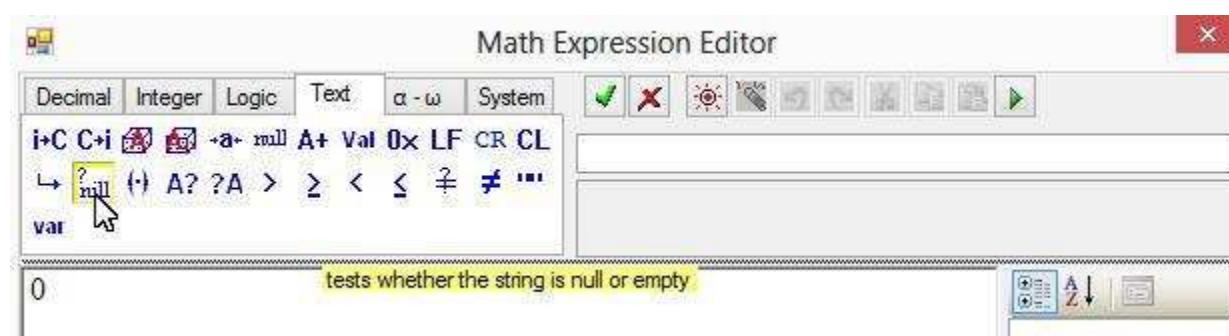
Login Manager Windows Form Sample – Part 1 | 2013



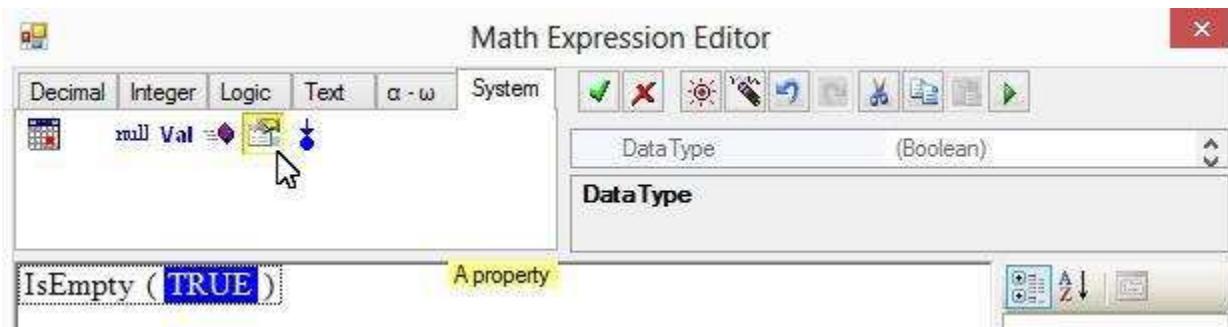
Rename the action and set the ActionCondition:



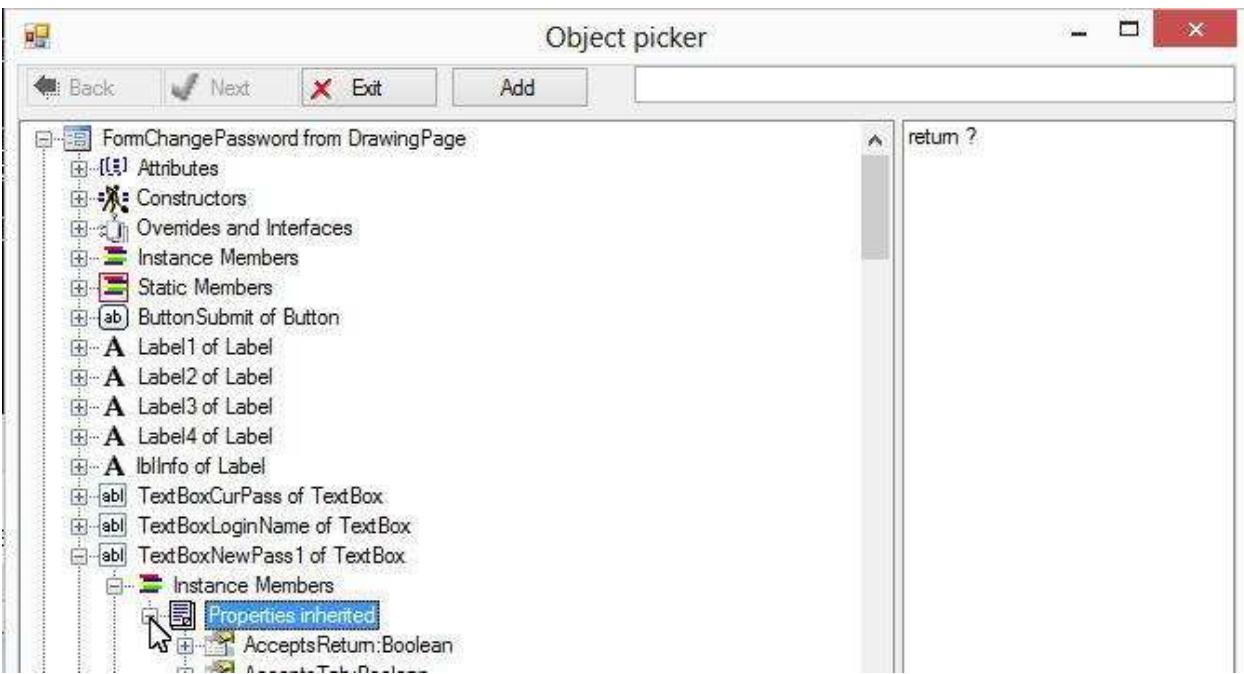
Click “?null”:



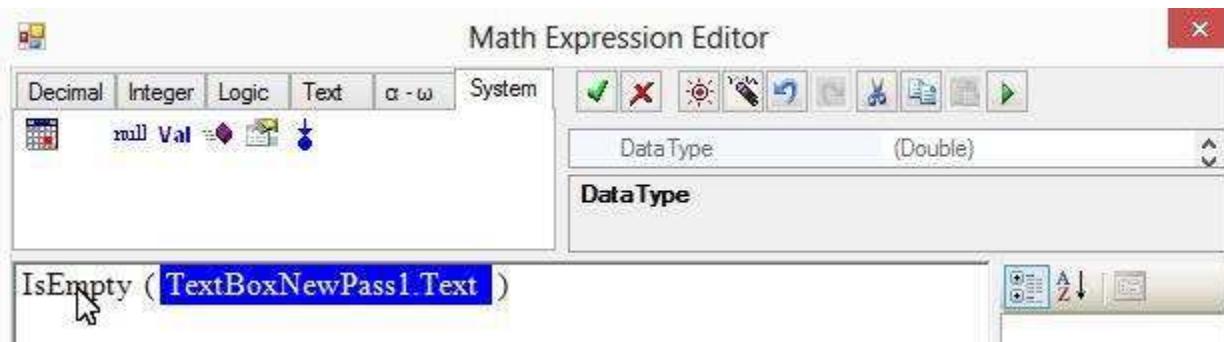
Click “TRUE” inside IsEmpty and click the Property icon:



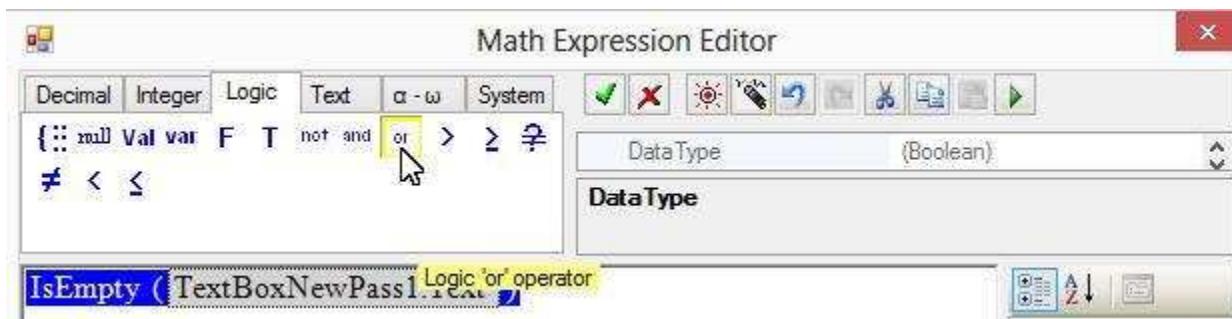
Select the Text property of the new password text box:



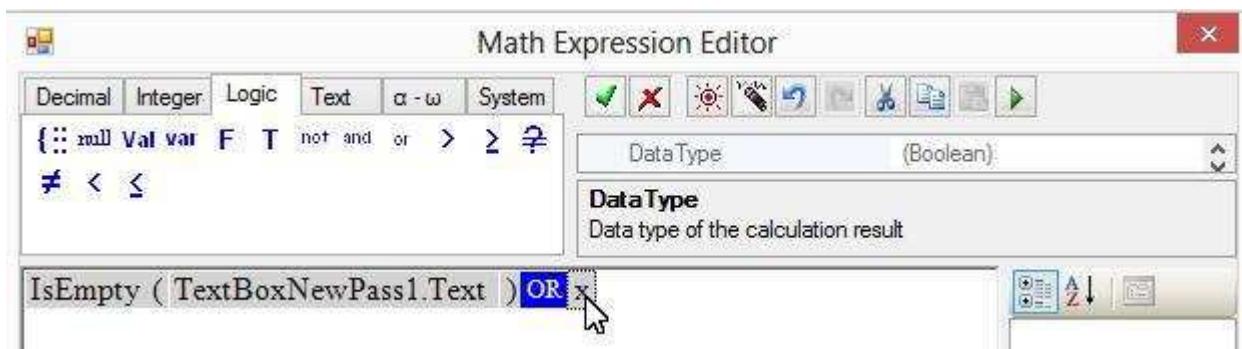
Click IsEmpty:



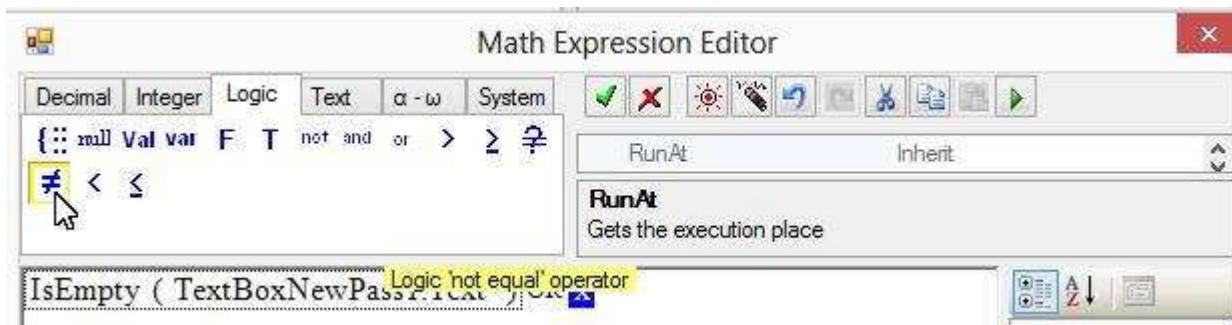
Click or:



Click x:



Click “=/”:



Login Manager Windows Form Sample – Part 1 | 2013

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

The screenshot displays four windows related to configuring a condition in a flowchart:

- Math Expression Editor**: Shows the expression `IsEmpty (TextBoxNewPass1.Text) OR TextBoxNewPass1.Text = TRUE`. The cursor is over the `TextBoxNewPass1.Text` part of the second term.
- Object picker**: Shows the path `TextBoxNewPass1 of TextBox.Text` selected in the tree view.
- Object picker**: Shows the path `TextBoxNewPass1 of TextBox.Text` selected in the tree view.
- Math Expression Editor**: Shows the expression `IsEmpty (TextBoxNewPass1.Text) OR TextBoxNewPass1.Text = TRUE`.
- Math Expression Editor**: Shows the expression `IsEmpty (TextBoxNewPass1.Text) AND TextBoxNewPass1.Text = TRUE`. The cursor is over the `TextBoxNewPass1.Text` part of the second term.

Login Manager Windows Form Sample – Part 1 | 2013

Object picker

FormChangePassword from DrawingPage

- Attributes
- Constructors
- Overrides and Interfaces
- Instance Members
- Static Members
- ButtonSubmit of Button
- Label1 of Label
- Label2 of Label
- Label3 of Label
- Label4 of Label
- LblInfo of Label
- TextBoxCurPass of TextBox
- TextBoxLoginName of TextBox
- TextBoxNewPass1 of TextBox
- TextBoxNewPass2 of TextBox
- Instance Members
- Properties inherited
- AcceptsReturn: Boolean
- AcceptsTab: Boolean

return ?
TextBoxNewPass1 of TextBox.Text

Object picker

Tag:Object

- Text String
- TextAlign:HorizontalAlignment
- TextLength:Int?

return ?
TextBoxNewPass1 of TextBox.Text

Math Editor

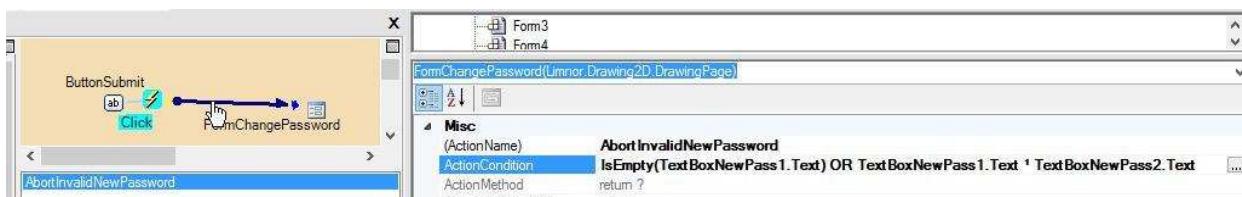
Finish the editing.

RunAt

Gets the execution place

IsEmpty (TextBoxNewPass1.Text) OR TextBoxNewPass1.Text = TextBoxNewPass2.Text

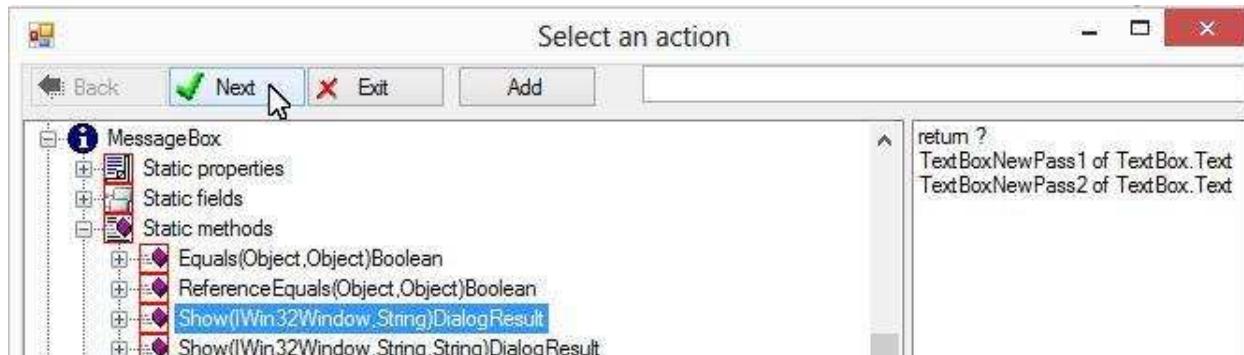
This is our input verification:



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:

Action Properties

(ActionName)	MessageBox.Show
Description	
ReturnValueType	DialogResult
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	MessageBox.Show(IWin32Window, String)DialogR
owner	IWin32Window()
text	abc ConstantValue
AssignTo	Property

Object picker

FormChangePassword from DrawingPage	return ? TextBoxNewPass1 of TextBox.Text TextBoxNewPass2 of TextBox.Text
Primary types	

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

(ActionName)	MessageBox.Show
Description	
ReturnValueType	DialogResult
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	MessageBox.Show(IWin32Window, String) DialogResult
owner	FormChangePassword

(ActionName)	MessageBox.ShowVerifyFail
Description	
ReturnValueType	DialogResult
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	IsEmpty(TextBoxNewPass1.Text) OR TextBoxNewPass1.Text != TextBoxNewPass2.Text
ActionMethod	MessageBox.Show(IWin32Window, String) DialogResult
owner	FormChangePassword
text	New password cannot be empty and must match confirm password

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

AbortInvalidNewPassword

MessageBox.ShowVerifyFail

Remove from event

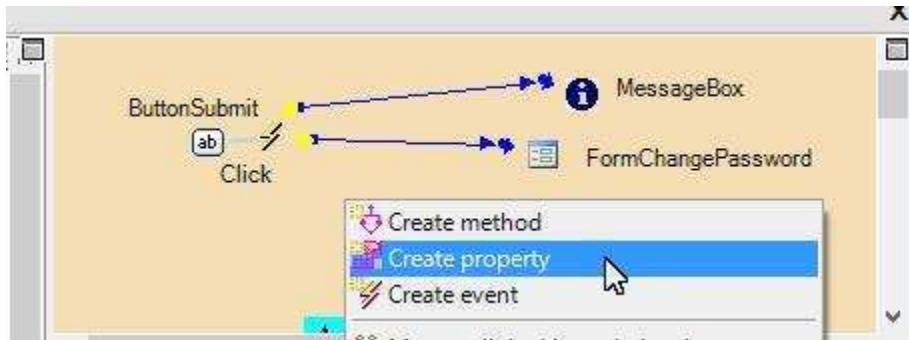
Move up

(ActionName)	MessageBox.ShowVerifyFail
ActionCondition	IsEmpty(TextBoxNewPass1.Text) OR TextBoxNewPass1.Text != TextBoxNewPass2.Text
ActionMethod	MessageBox.Show(IWin32Window, String) DialogResult
AssignTo	[null]
BreakAfterExecute	False
BreakBeforeExecute	False

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

Execute password change

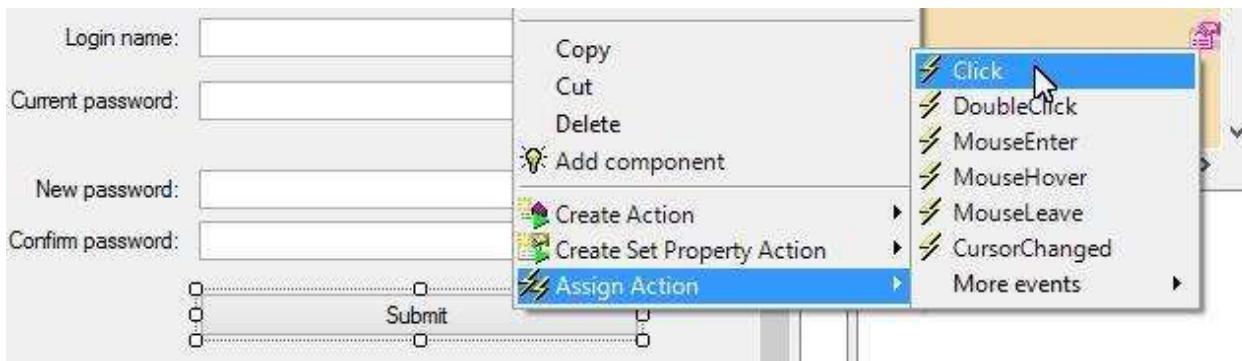
We use a property to indicate whether password changing is successful:



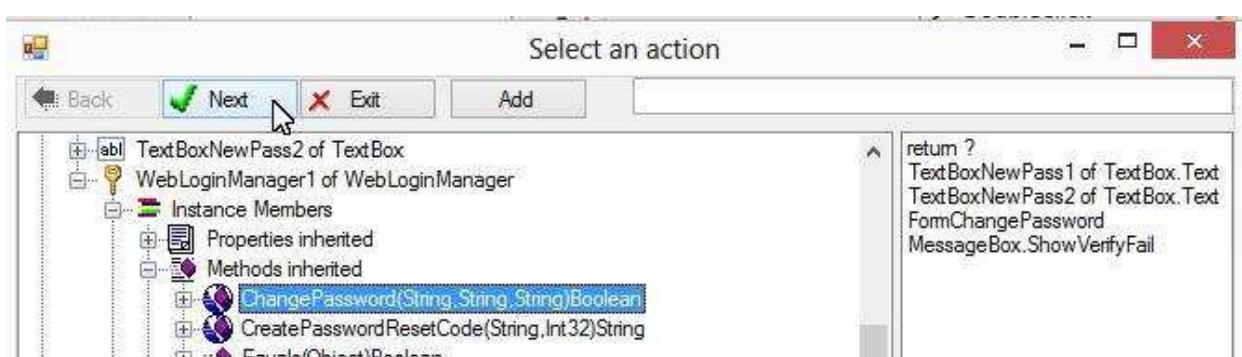
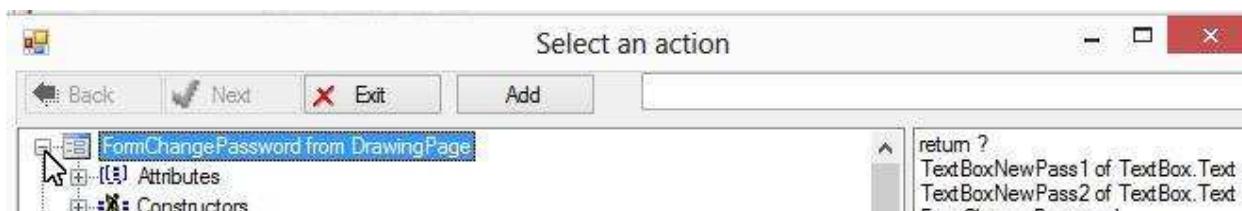
Rename the property and change its type to be Boolean:

The WinForm Designer interface is shown again. On the left, a label control is selected. In the properties pane, under the 'Misc' section, the 'Name' is 'lblInfo', 'AccessControl' is 'Public', 'CanRead' is 'True', 'CanWrite' is 'True', 'DefaultValue' is empty, 'Description' is empty, 'Getter' is 'Get property value of Property1', 'IsReadOnly' is 'False', and 'PropertyName' is 'Property1'. The 'PropertyType' is currently set to 'String'. At the bottom, an 'Object picker' dialog is open, showing the 'Primary types' section with 'Boolean' selected.

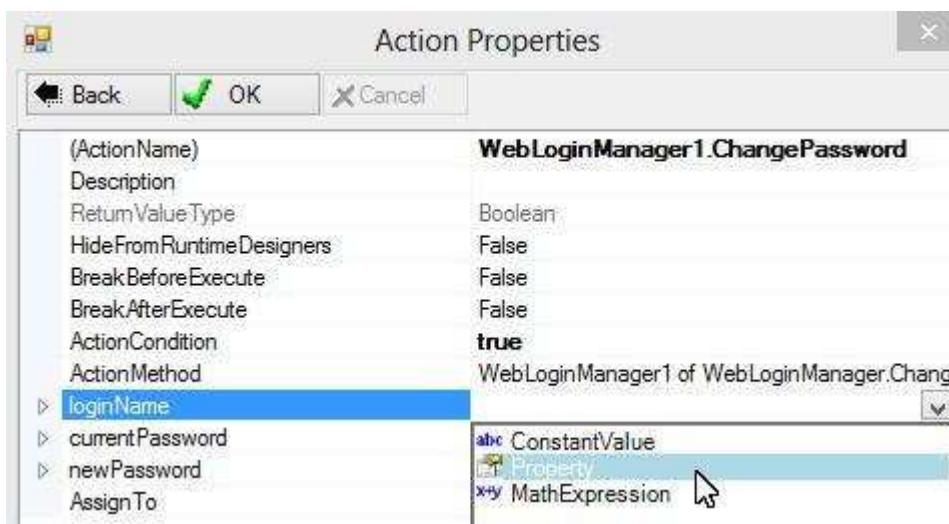
Now we may execute a ChangePassword action:



Select ChangePassword method:



Pass Text property of the text boxes to the action:



Object picker

FormChangePassword from DrawingPage

Attributes Constructors

return ?
TextBoxNewPass1 of TextBox.Text
TextBoxNewPass2 of TextBox.Text
FormChangePassword

Object picker

Back Next Exit Add

Properties inherited

AcceptsReturn: Boolean
AcceptsTab: 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
TextLength: Int32

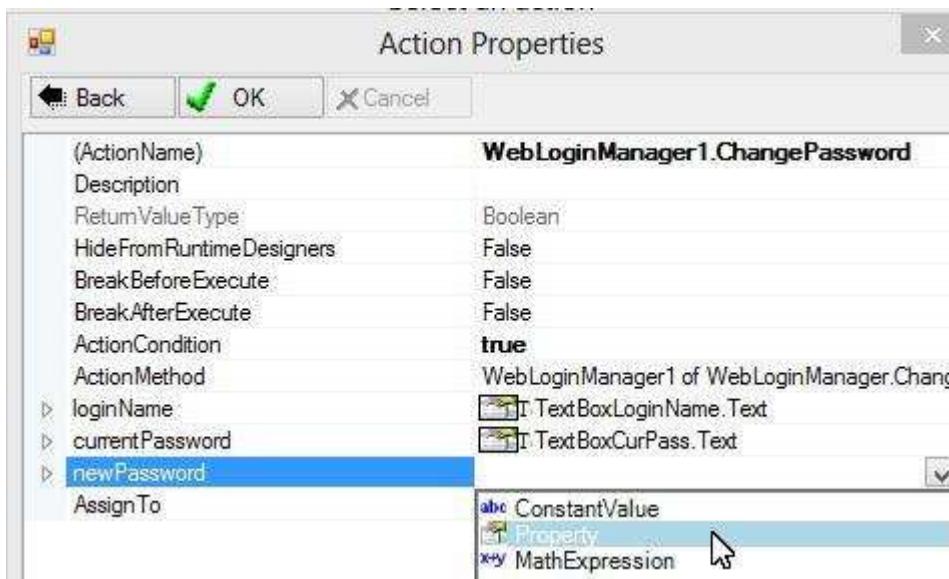
return ?
TextBoxNewPass1 of TextBox.Text
TextBoxNewPass2 of TextBox.Text
FormChangePassword
MessageBox.ShowVerifyFail

Action Properties

(ActionName)
Description
ReturnValueType
HideFromRuntimeDesigners
BreakBeforeExecute
BreakAfterExecute
ActionCondition
ActionMethod
loginName
currentPassword
newPassword
AssignTo

WebLoginManager1.ChangePassword

Boolean
False
False
False
true
WebLoginManager1 of WebLoginManager.ChangePassword
TextBox.LoginName.Text
ConstantValue
Property
MathExpression



Set "AssignTo" to PasswordChanged property:

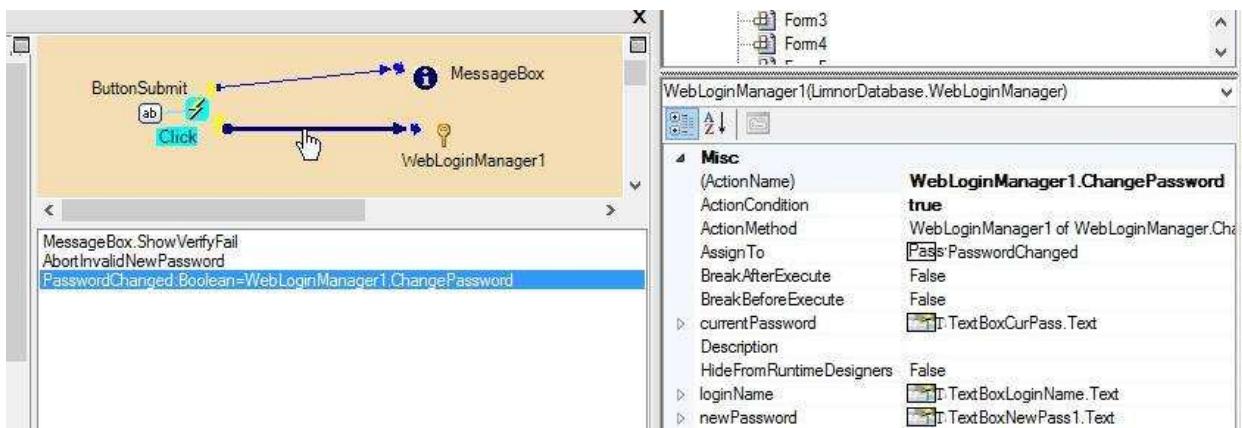
The figure consists of three vertically stacked windows from a software interface:

- Action Properties** window (top):

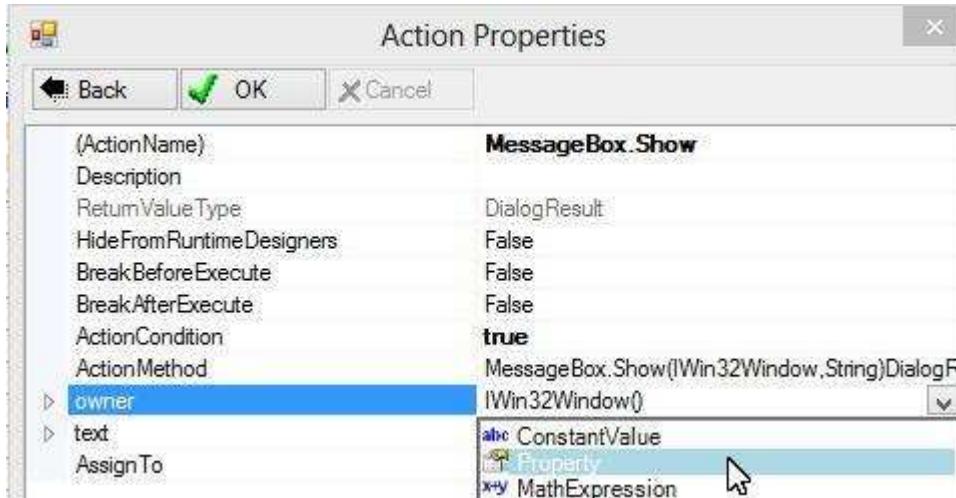
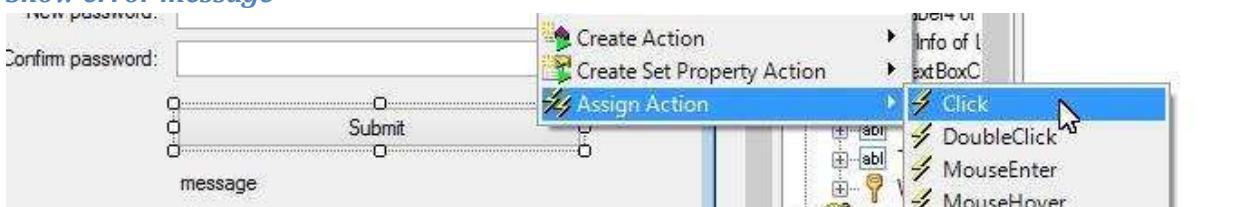
(ActionName)	WebLoginManager1.ChangePassword
Description	
ReturnValueType	Boolean
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	WebLoginManager1 of WebLoginManager.ChangePassword
▷ loginName	TextBoxLoginName.Text
▷ currentPassword	TextBoxCurPass.Text
▷ newPassword	TextBoxNewPass1.Text
AssignTo:	{null}
- Object picker** window (middle):
 - Shows a tree view of class members for "FormChangePassword from DrawingPage".
 - The "Properties" node is expanded, showing "PasswordChanged:Boolean" and "AllowDrop:Boolean".
 - The "PasswordChanged:Boolean" item is selected.
 - A preview pane on the right shows the selected property: "TextBoxLoginName of TextBox.Text", "TextBoxCurPass of TextBox.Text", and "TextBoxNewPass1 of TextBox.Text".
- Action Properties** window (bottom):

(ActionName)	WebLoginManager1.ChangePassword
Description	
ReturnValueType	Boolean
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	WebLoginManager1 of WebLoginManager.ChangePassword
▷ loginName	TextBoxLoginName.Text
▷ currentPassword	TextBoxCurPass.Text
▷ newPassword	TextBoxNewPass1.Text
AssignTo:	Pass.PasswordChanged

The action is created and assigned to the button:



Show error message



Object picker

FormChangePassword from DrawingPage

Attributes Constructors

TextBoxLoginName of TextBox.Text
TextBoxCurPass of TextBox.Text
TextBoxNewPass1 of TextBox.Text

Action Properties

(ActionName) MessageBox.ShowError

Description

ReturnValueType DialogResult

HideFromRuntimeDesigners False

BreakBeforeExecute False

BreakAfterExecute False

ActionCondition true

ActionMethod MessageBox.Show(IWin32Window, String) DialogResult

owner FormChangePassword

text Invalid login name

Assign To [null]

Action Properties

(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

0 A property

Object picker

Back Exit Add

- FormChangePassword from DrawingPage
 - Attributes
 - Constructors
 - Overrides and Interfaces
 - Instance Members
 - Properties
 - PasswordChanged: Boolean**
 - AllowDrop: Boolean

TextBoxLoginName of TextBox.Text
 TextBoxCurPass of TextBox.Text
 TextBoxNewPass1 of TextBox.Text
 PasswordChanged
 WebLoginManager1.ChangePasswo
 FormChangePassword
 MessageBox.ShowError

Math Expression Editor

Decimal Integer Logic Text α - ω System

{ :: null Val var F T **not** and or > ≥ ? ≠ < ≤

CanWrite True

CanWrite

PasswordChanged Logic 'not' operator

Math Expression Editor

Finish the editing.

Decimal Integer Logic Text α - ω System

{ :: null Val var F T **not** and or > ≥ ? ≠ < ≤

Name math

Name

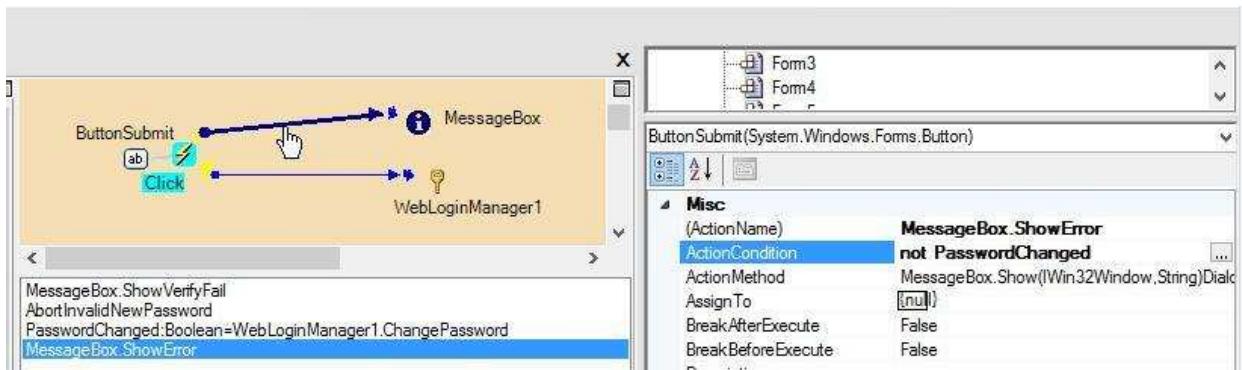
NOT PasswordChanged

Action Properties

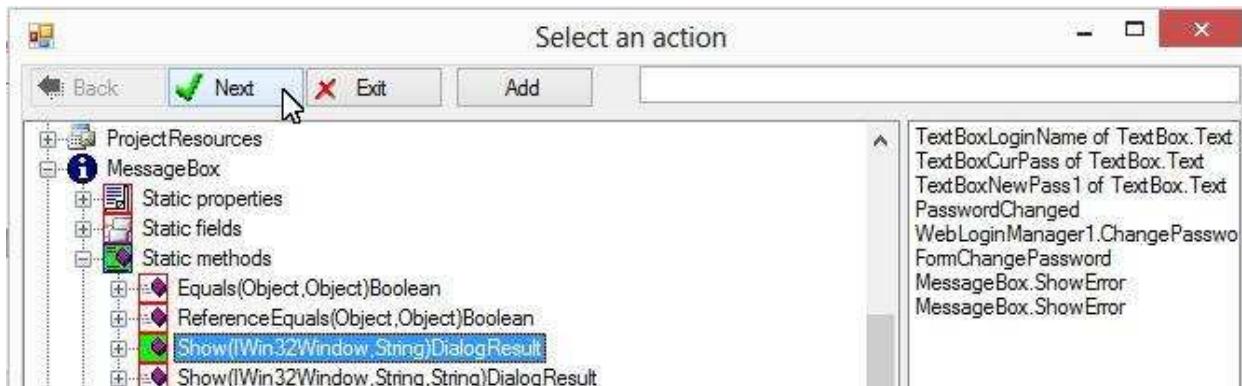
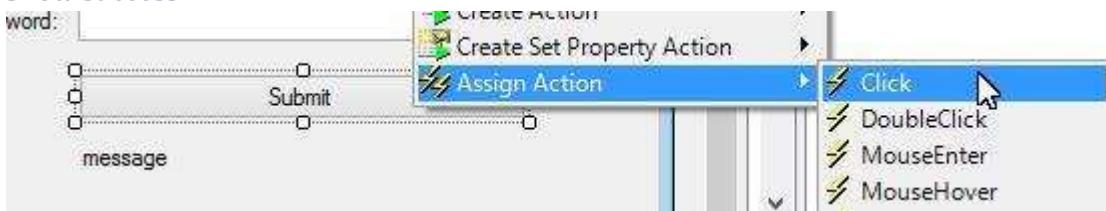
Back Cancel

(ActionName)	MessageBox.ShowError
Description	
ReturnValueType	DialogResult
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	not PasswordChanged
ActionMethod	MessageBox.Show(IWin32Window, String)DialogResult
owner	FormChangePassword
text	Invalid user credential
AssignTo	Null

The action is created and assigned to the button:



Show success



Action Properties

(ActionName)	MessageBox.Show
Description	
ReturnValueType	DialogResult
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	MessageBox.Show(IWin32Window, String) DialogResult
owner	IWin32Window()
text	ConstantValue
AssignTo	Property

Object picker

FormChangePassword from DrawingPage	TextBoxLoginName of TextBox.Text
Primary types	TextBoxCurPass of TextBox.Text
Project Resources	TextBoxNewPass1 of TextBox.Text

Action Properties

(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

Toolbar:

- Decimal
- Integer
- Logic
- Text
- $\alpha - \omega$
- System
- Checkmark icon
- X icon
- Lightbulb icon
- Calculator icon
- Clipboard icon
- Green arrow icon

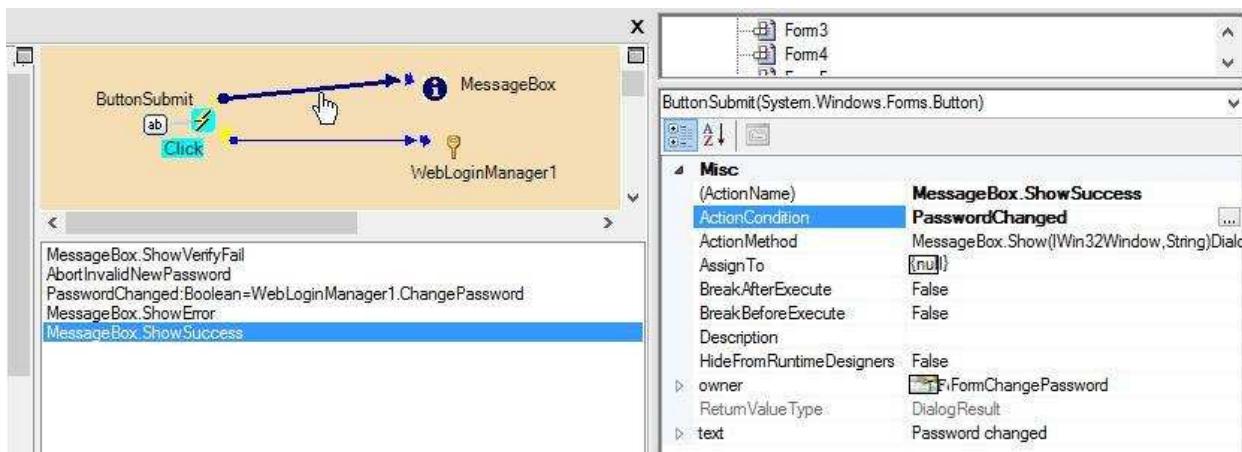
Text Input:

0 A property

The screenshots illustrate the configuration of a Windows Form action:

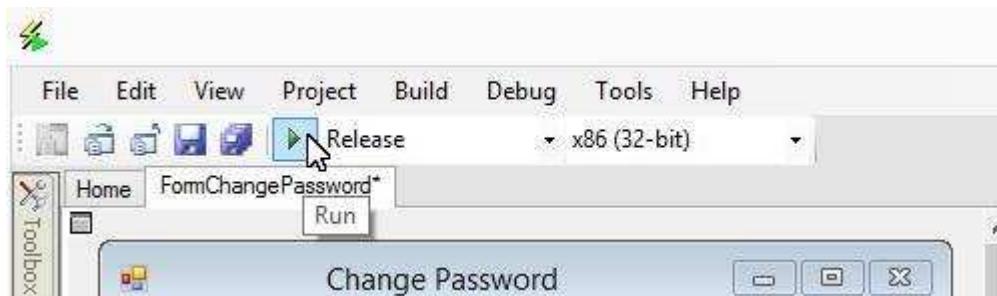
- Object picker:** Shows the selection of `FormChangePassword from DrawingPage`. The `PasswordChanged` property is highlighted.
- Math Expression:** Shows the expression `CanWrite` set to `True`.
- Action Properties:** Shows the creation of an action named `MessageBox.ShowSuccess` with the condition `PasswordChanged`.

The action is created and assigned to the button:



Test

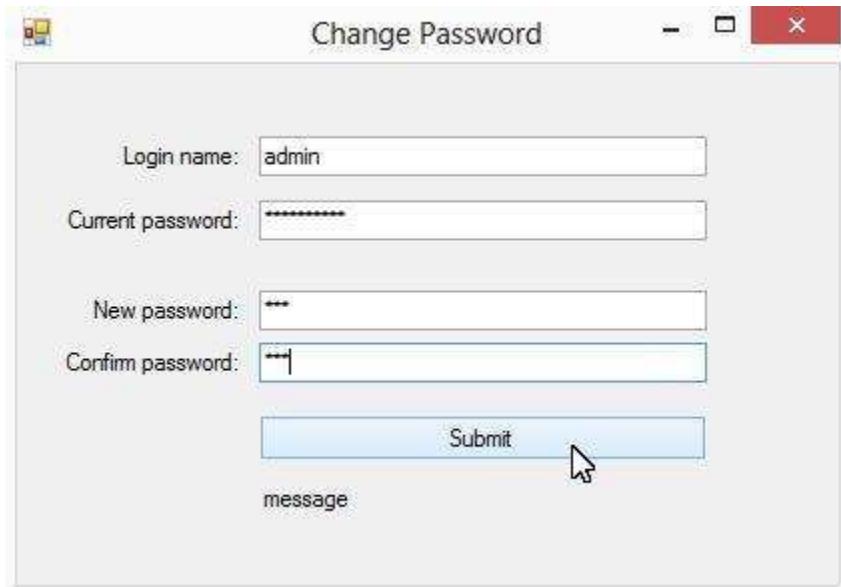
Now we may change the password of the first user record.



Click "Change password":



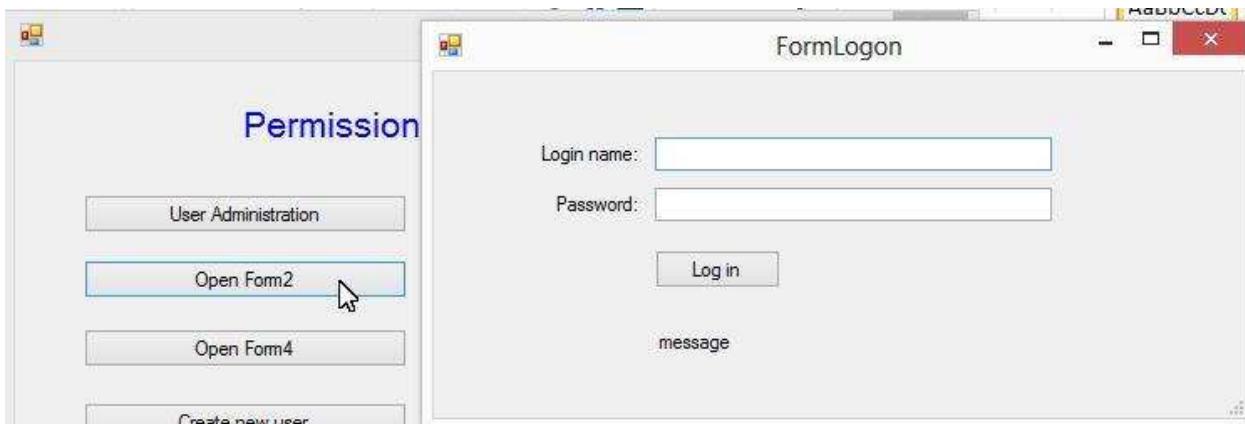
Enter the login name and the password; enter new password; click Submit:



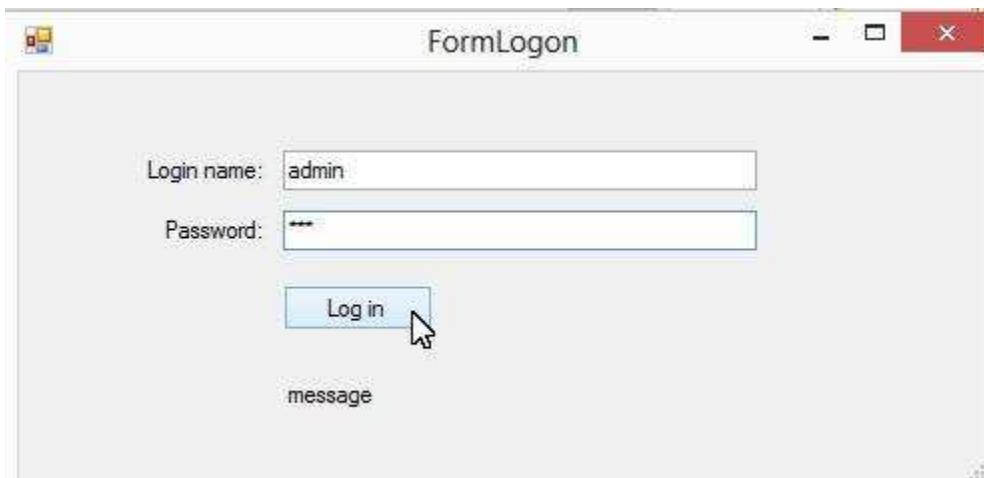
We get message “Password changed”:



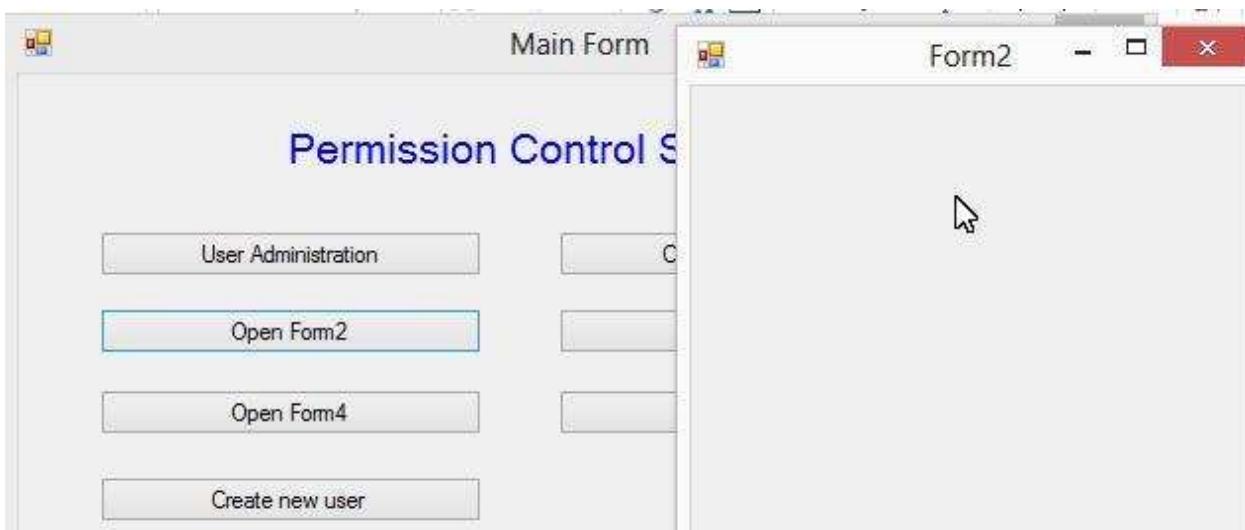
Let's try to use our new password. Click “Open Form 2”, the login form appears:



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>