

Programming with Visual HTML Editor

Last modify: Wednesday, March 12, 2014

Table of Contents

Introduction	2
Use Visual HTML Editor.....	2
Web Programming.....	2
Include Elements for Programming	2
Identifying Elements	4
User Input Validation	7
Use Abort in event handling	15
Insert Database Record.....	17
Send email.....	24
Set email recipients.....	24
Set email subject.....	26
Set email body.....	27
Send email.....	29
Assign actions to button	30
Save Error Message to Database	31
Show error message on web page.....	36
Save local ending time to database	39
Test.....	45
Data binding.....	48
File Upload	48
Feedback	49

Introduction

Limnor Studio provides two web page editors: Web Form Editor and Visual HTML Editor. See <http://www.limnor.com/support/webEditors.pdf>. This document shows programming samples using the Visual HTML Editor.

One sample is to develop a web page for sending emails. It uses text boxes for the visitor to enter email recipients, email subject and email body. It uses a button to send email. It uses a database to record the time used for sending the email.

One sample is to use data-binding to link databases to HTML elements. Data from database are automatically displayed on HTML elements. The visitor enters data into HTML elements and the data are saved back to the databases.

One sample is to use File Upload element to upload files to web server.

Use Visual HTML Editor

See <http://www.limnor.com/support/WebHtmlEditorProgramming1.pdf>

Web Programming

For this sample, we want to program it in the following way. When the visitor clicks the button, the web page will perform following actions:

1. checks that each email address in the text box for recipients is a valid email address; email addresses are separated by semicolons;
2. checks that the subject text box is not empty;
3. checks that the email body is not empty;
4. creates a new database record to record the web local time as the starting time, and also record email parameters;
5. sends an email to the recipients;
6. records the error message of email sending to the database;
7. shows the error message of email sending to the web page;
8. records the web page local time as the ending time to the database

Include Elements for Programming

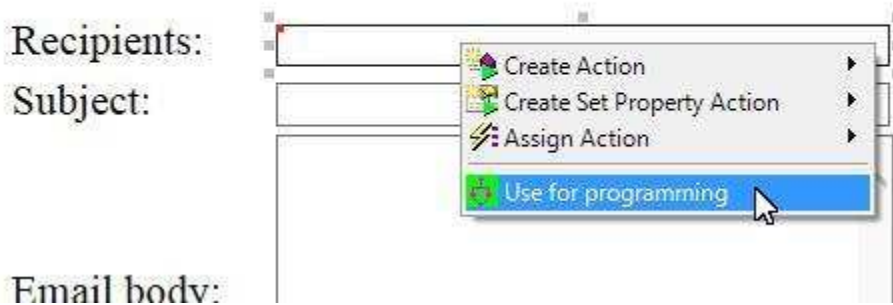
When you use an element in programming, for example, creating a method-execution action, creating property-setting action, or assigning actions to an event, the element is used in programming.

In some situations, you need to manually include an element for programming. For example, if you want to use an element's property in an action and the element has never being used then you need to manually include the element for programming first so that its properties can be accessed by your visual programming process.

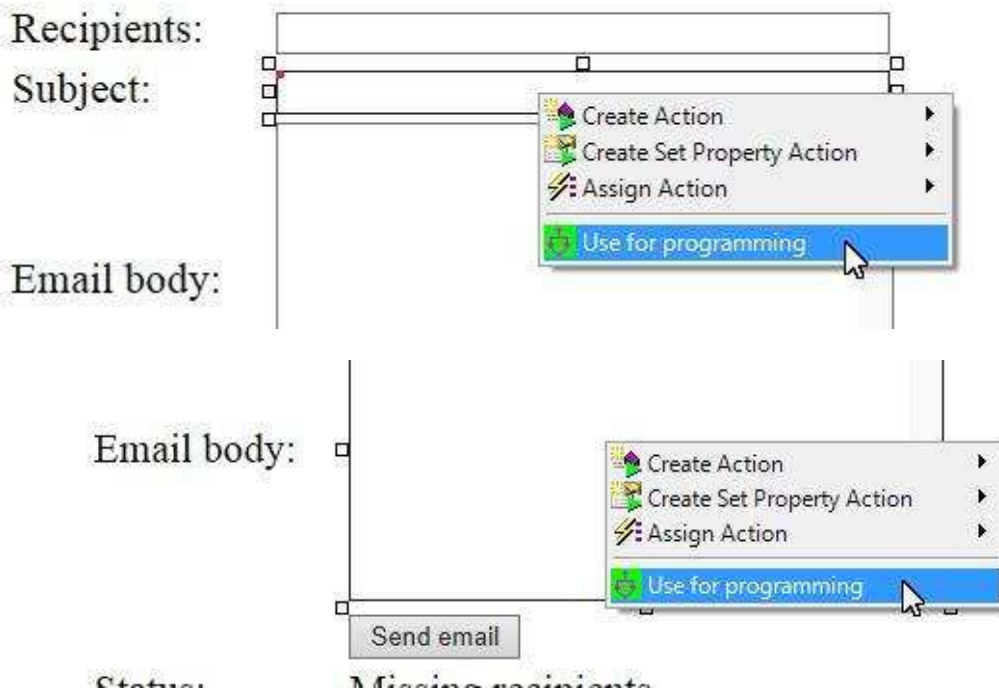
For example, we need to access “value” property of the text boxes for actions 1, 2, 3, 4, and 5.

To manually include an element for programming is easy. Right-click it and then choose “Use for programming”.

Web Mail Sample

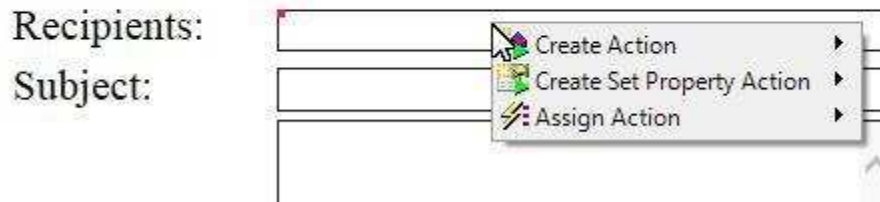


Web Mail Sample

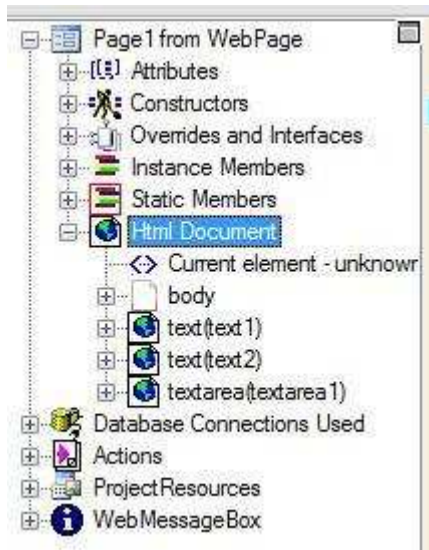
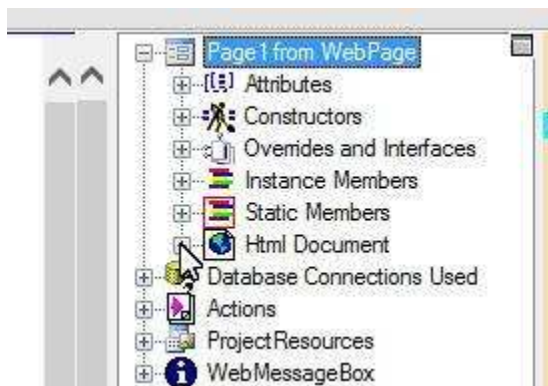


The menu item “Use for programming” just needs to be used once. Right-click a text box again, you can see that the menu item is no longer available:

Web Mail Sample



In the Object Explorer, elements used for programming are listed under Html Document:



Identifying Elements

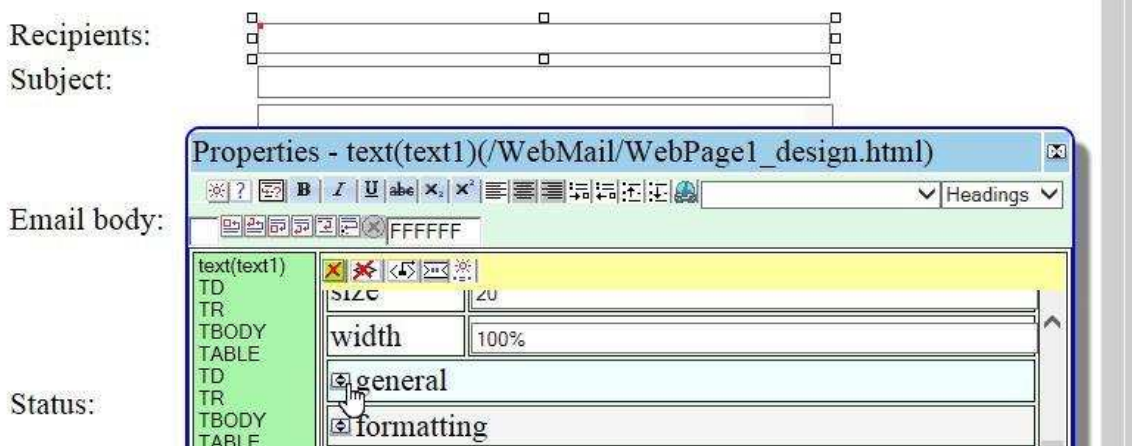
If you are familiar with standalone Form programming or Web Form Editor then you know that every control has a Name property uniquely identifying the control, the Name property cannot be empty. This is not the case for the Visual HTML Editor. In the Visual HTML Editor, identification is only required when an element is involved in programming.

In the Visual HTML Editor, each element can be identified by its “id” attribute and its “name” attribute. “id” and “name” can be empty. If you set “id” to non-empty then the value for “id” must be unique in the whole web page. But “name” can be duplicated.

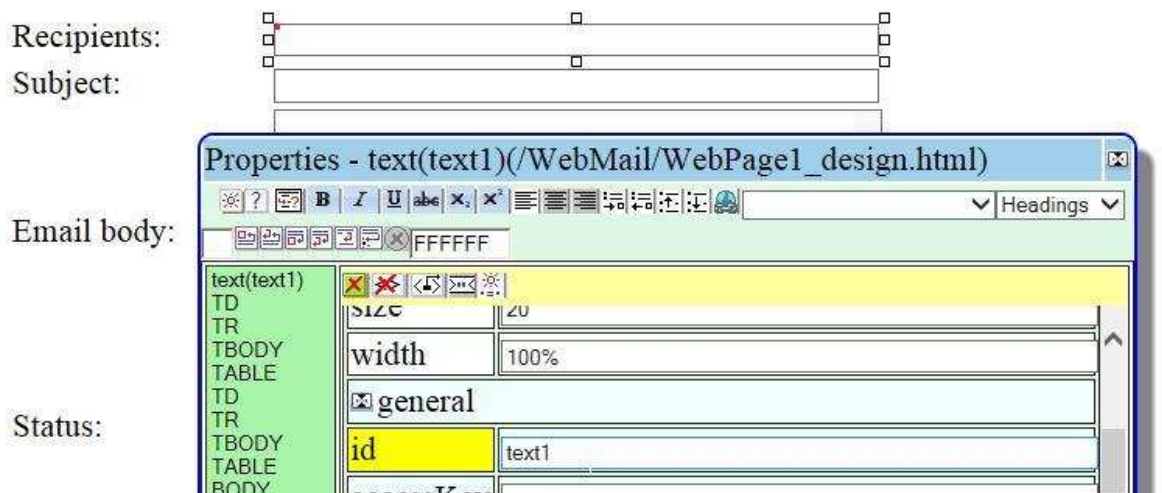
The first character of “id” must be an English letter. Other characters must be alphanumeric or underscores. Do not use spaces. “id” is case-sensitive.

When you use an element in programming, explicitly or implicitly, if the element’s “id” is empty then a unique “id” is automatically created for it.

For example, after choosing menu “Use for programming”, an “id” is created for the element. Open “general” group to find “id” attribute:



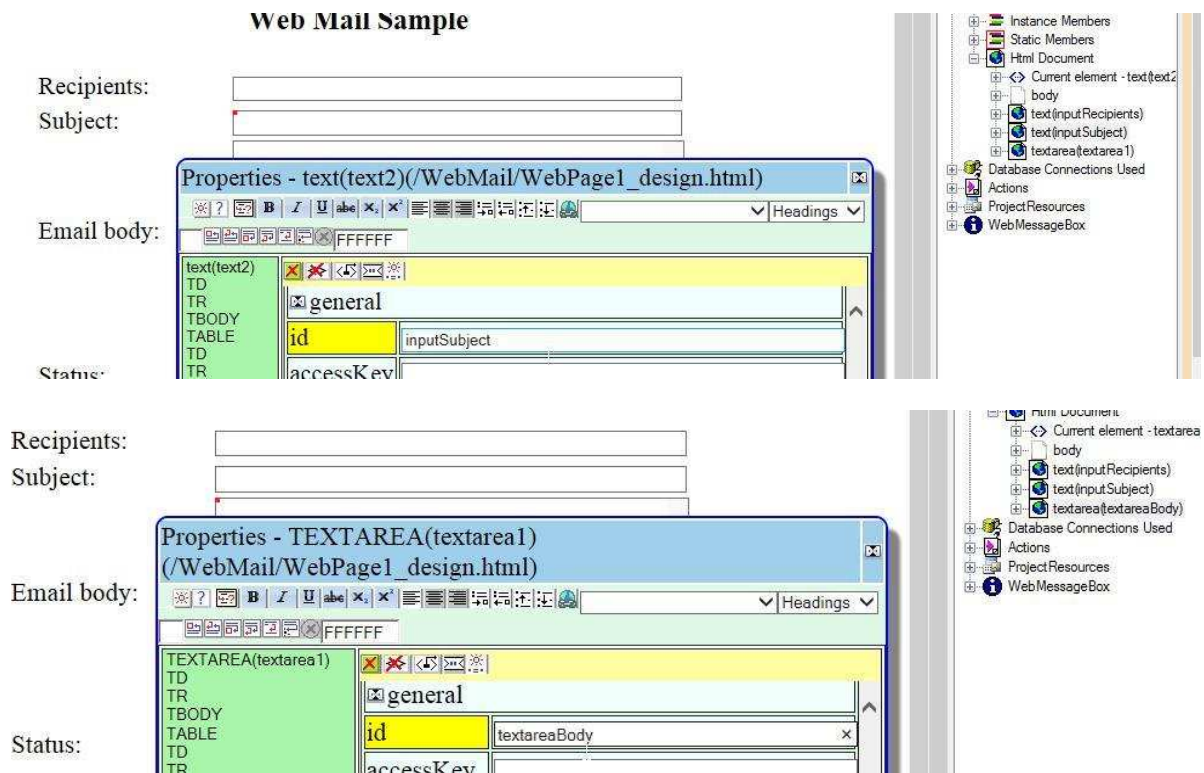
We can see that the “id” is “text1”:



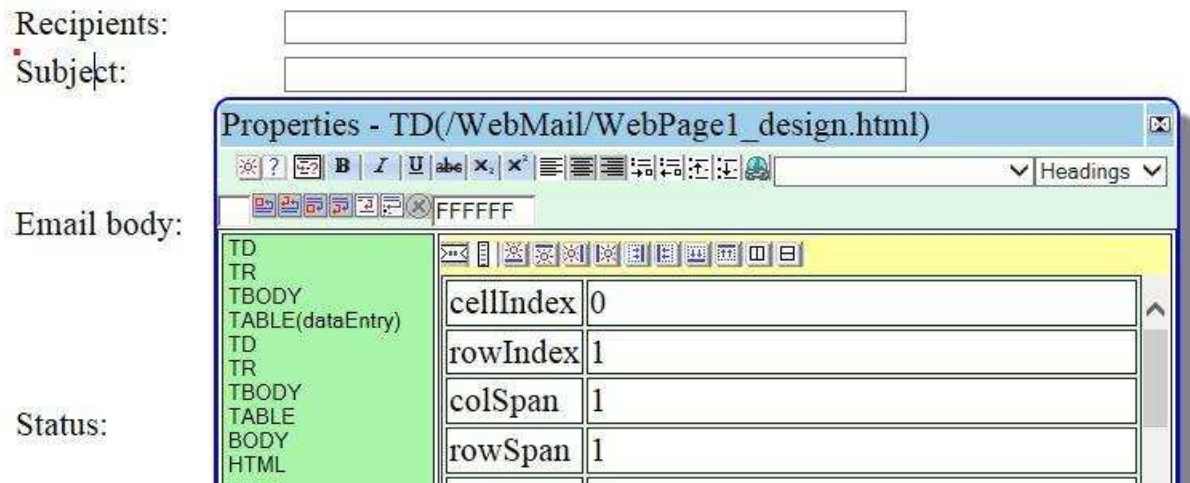
You may not like this automatically created id, “text1”. It is a good practice to give a meaningful id for each element used for programming. For example, let’s change “text1” to “inputRecipients”:



Change other automatically created “id” too.



Even if you do not use an element for programming, giving a name or id to an element can help you identify it if it is contained in other elements. For example, if we give an “id” to a table then we can easily identify it in the element list:



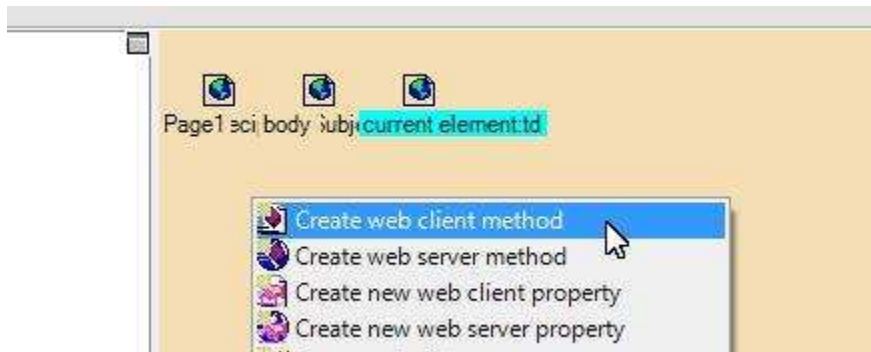
In the element list, we see two “TABLE” elements. Because we named one table “dataEntry”, we can easily distinguish the two “TABLE” elements.

User Input Validation

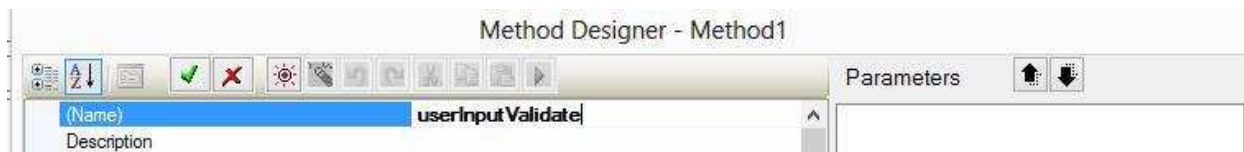
We want to do following user input validations:

1. checks that each email address in the text box for recipients is a valid email address;
2. checks that the subject text box is not empty;
3. checks that the email body is not empty;

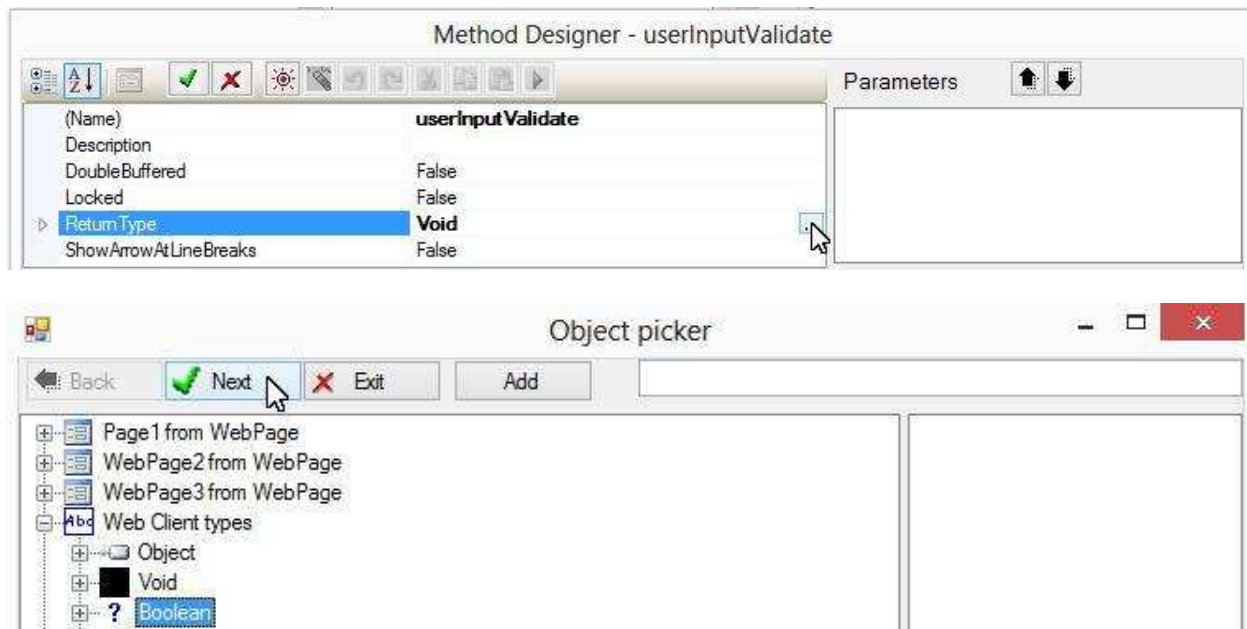
We may create a web client method to carry out these tasks:



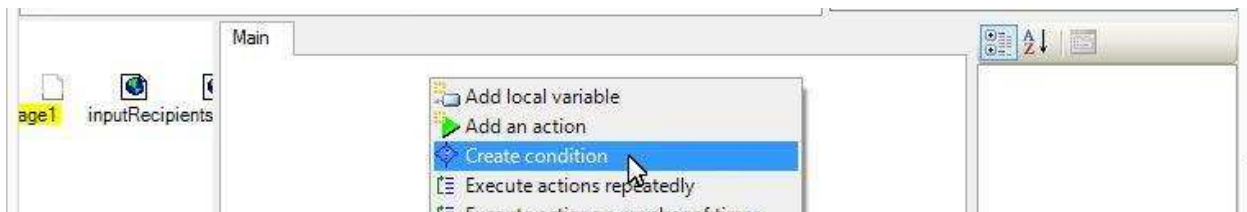
Rename it to userInputValidate:



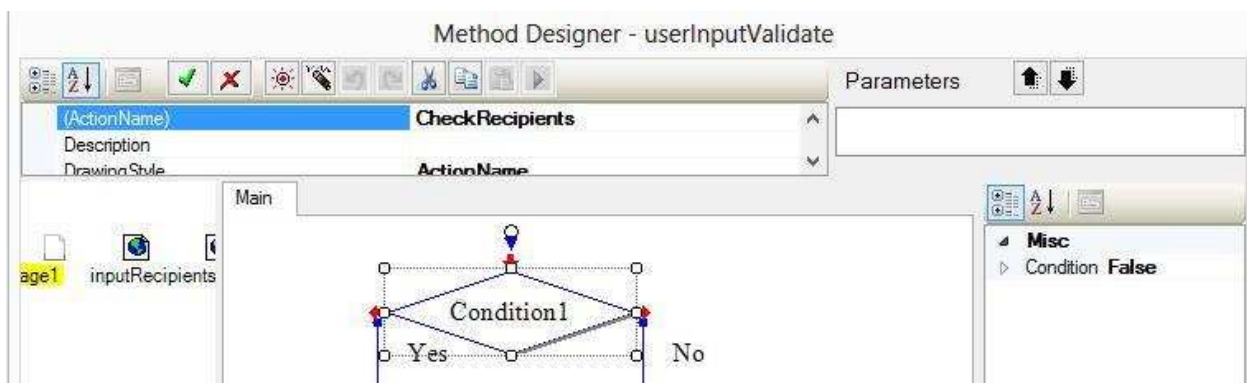
We want this method to return True if the validation passes; it returns False if the validation fails. Change the method return type to Boolean:



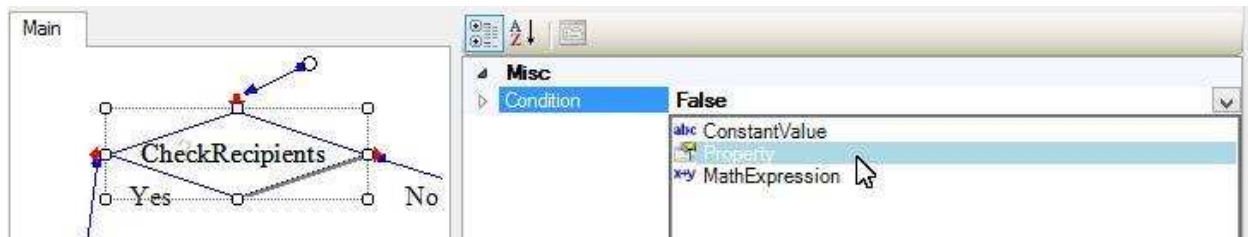
Add a Condition action for checking the recipients:



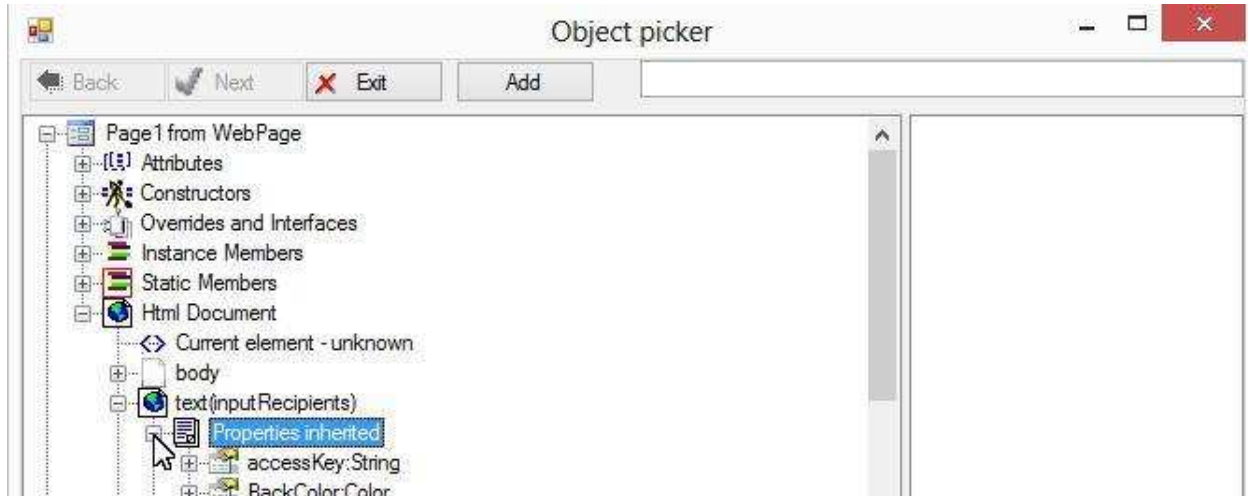
Rename it to "CheckRecipients":



Set its Condition to check the first text box:



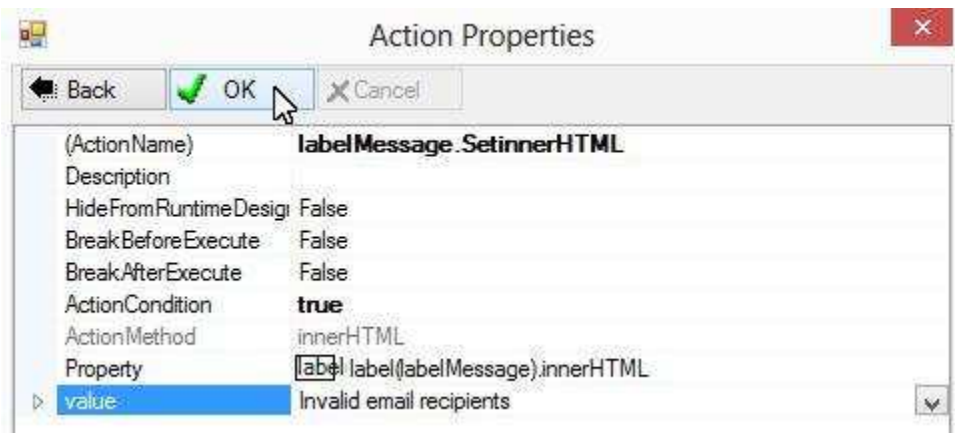
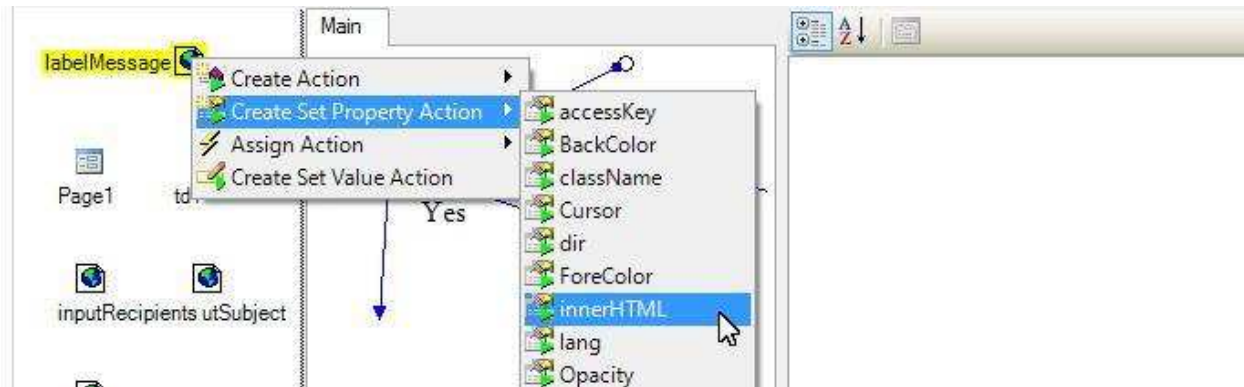
Select IsEmailAddressList property of the “value” property of the first text box:



Select IsEmailAddressList of “value” of the text box:

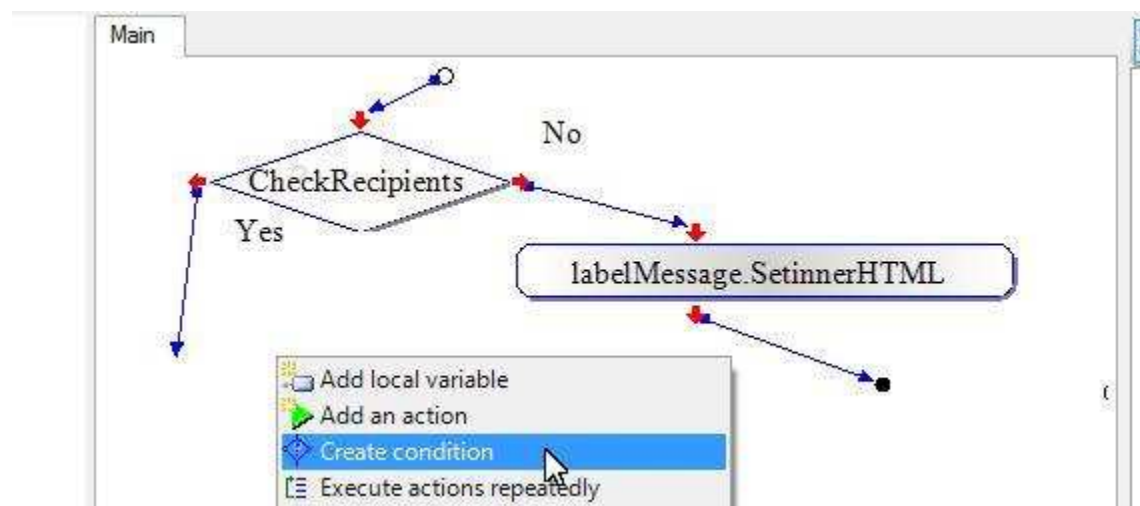


Set error message if the validation of recipients fails:

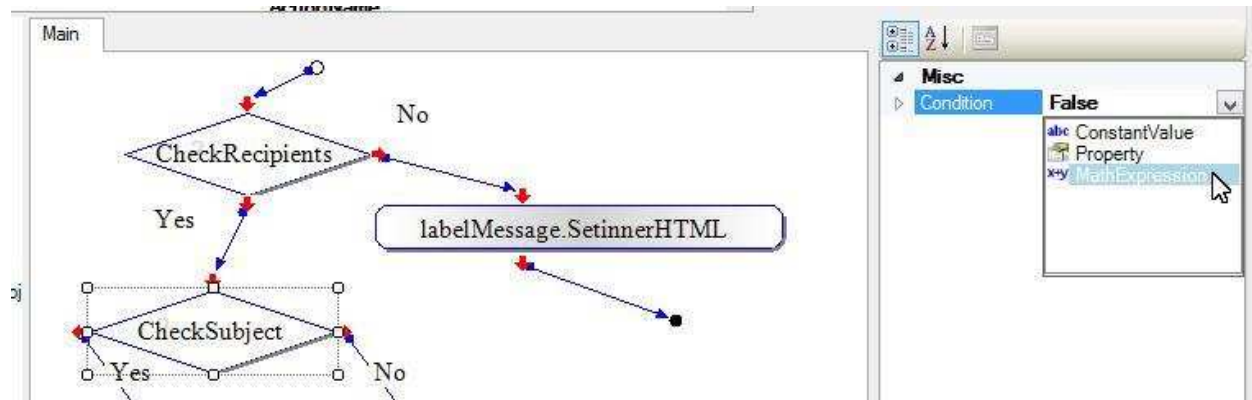


Link the action to the “No” port of the Condition action.

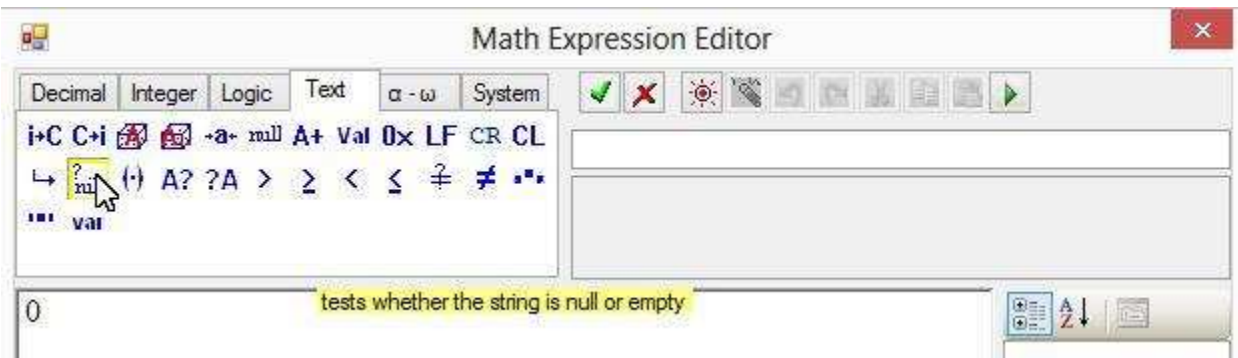
Add another Condition action to check email subject:



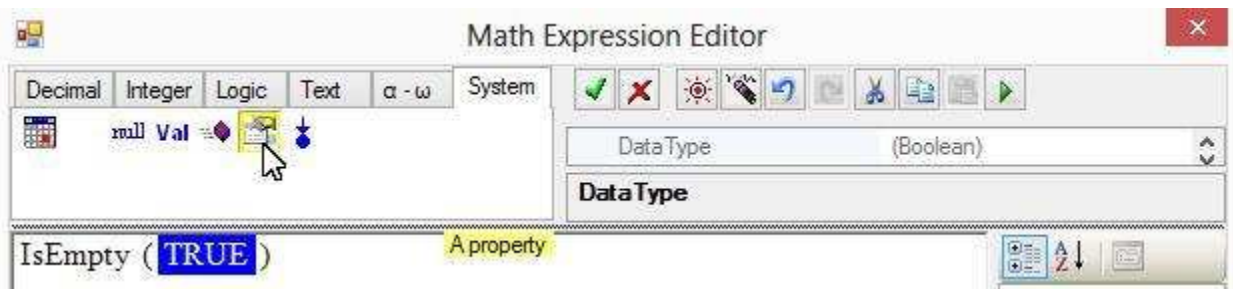
Link the new Condition to the “Yes” port of the first condition; rename the new Condition to “CheckSubject”. Set the condition to check email subject:

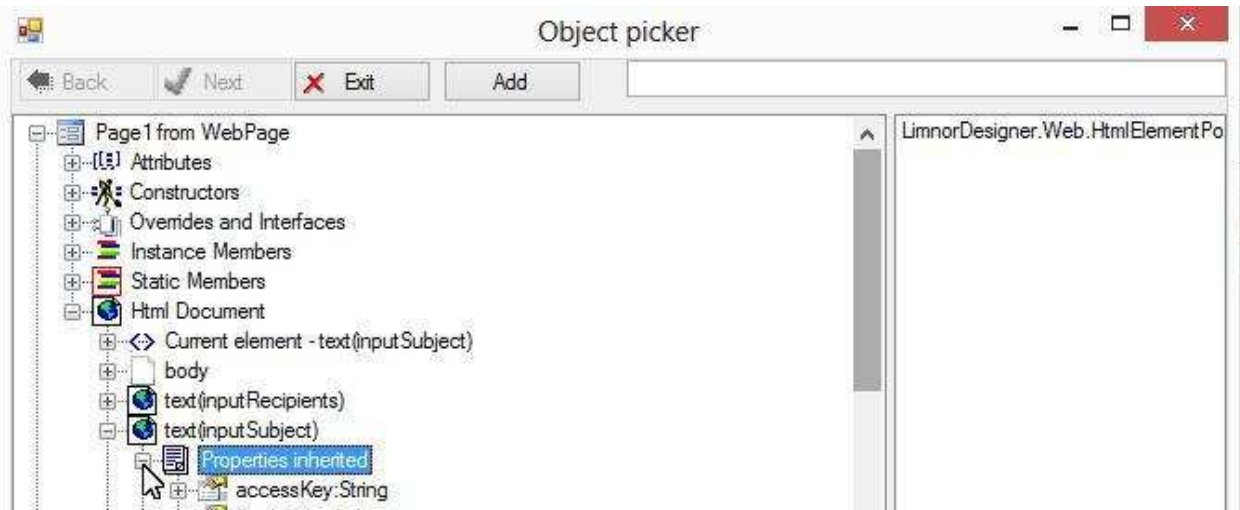


Use operator `?null` to check that email subject is empty:

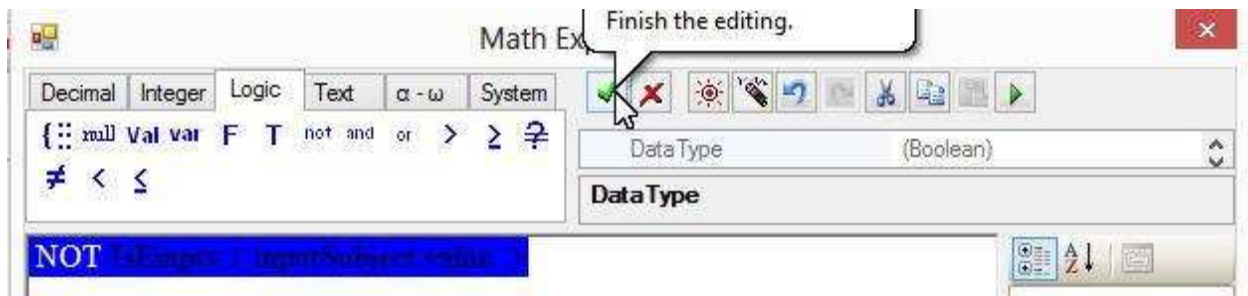
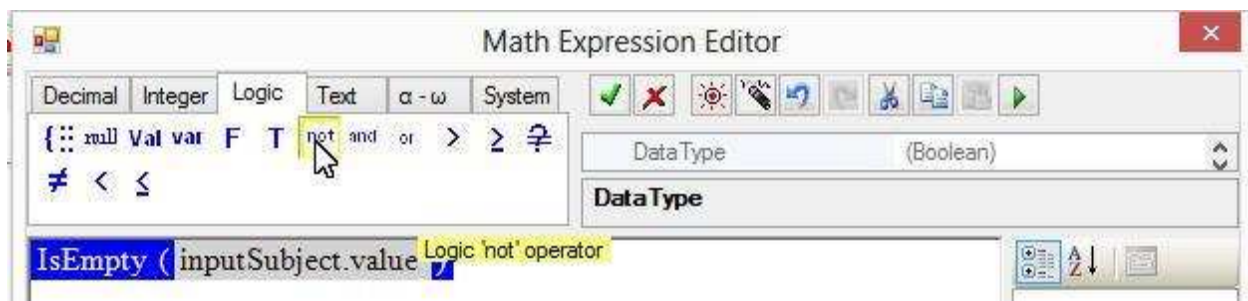


Choose the “value” of the text box for email subject:

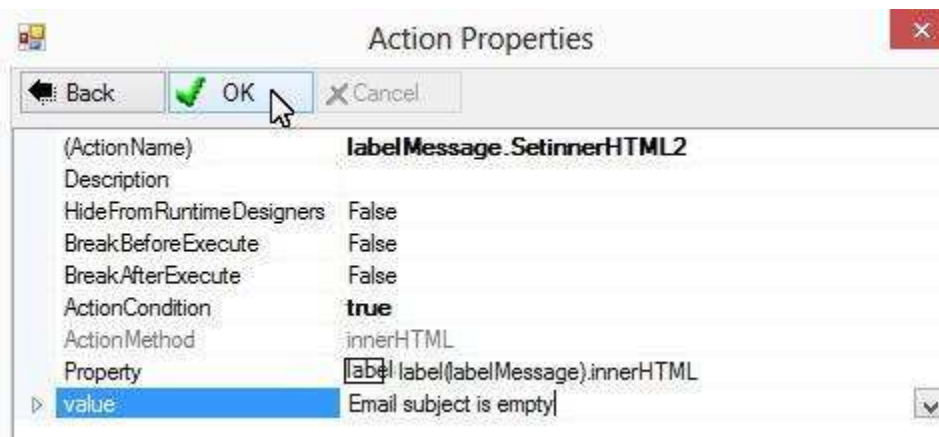
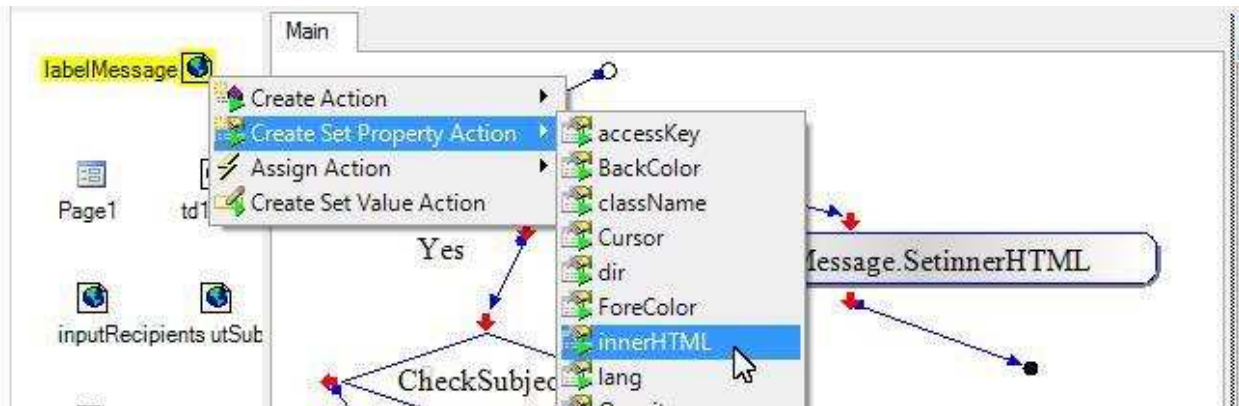




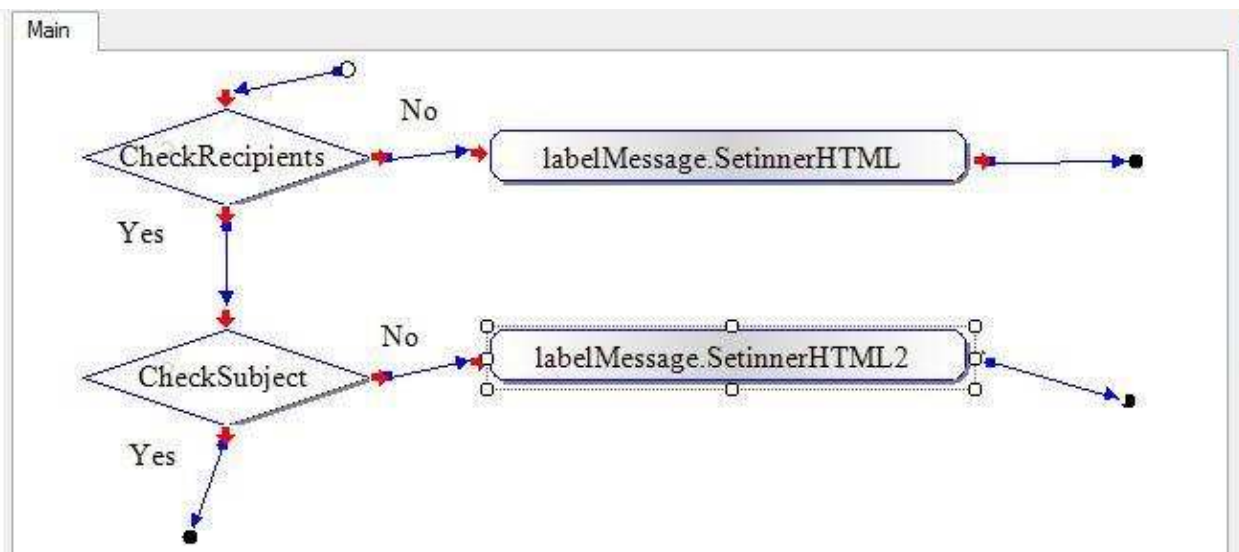
Apply “not” operator to it:



Show error message if the validation of email subject fails:

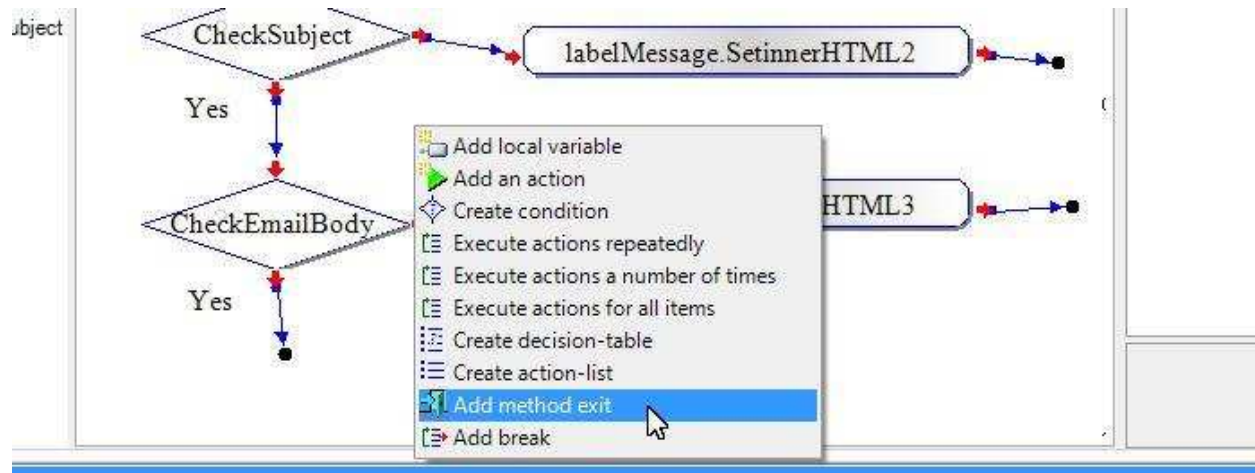


Link the action to the “No” port of the Condition:

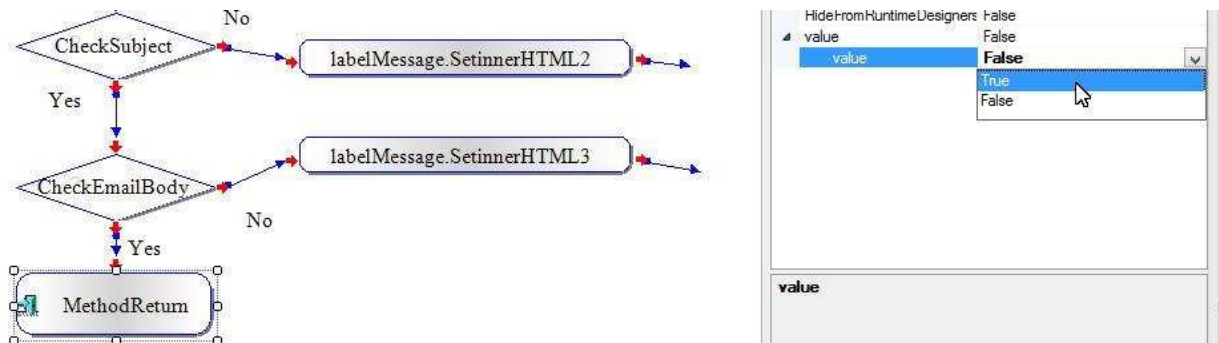


Add another Condition to check that the email body is not empty.

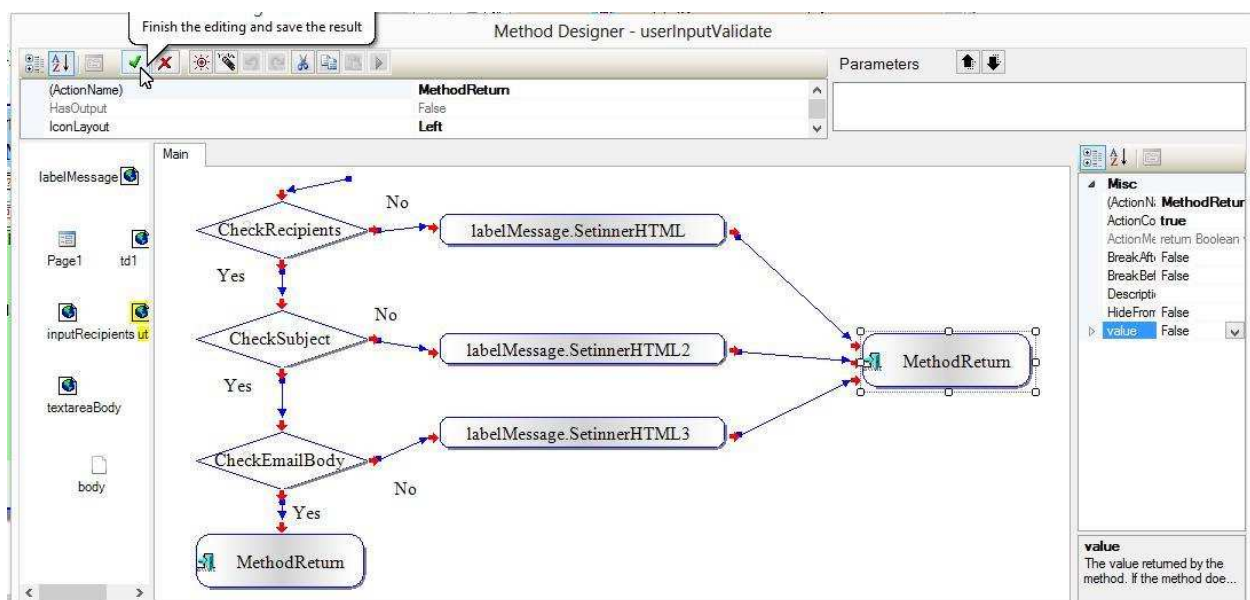
Add a method-return-action to return True if all validations pass:



Set "value" of the method return to True:



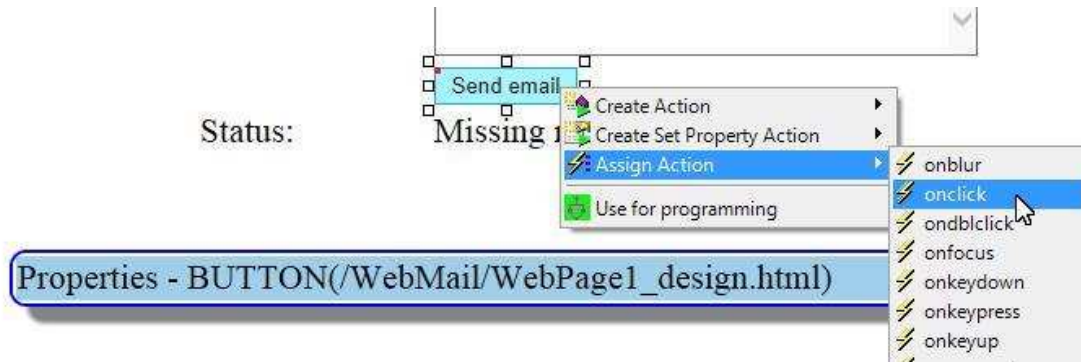
Add another method return to return False if any validations fail:



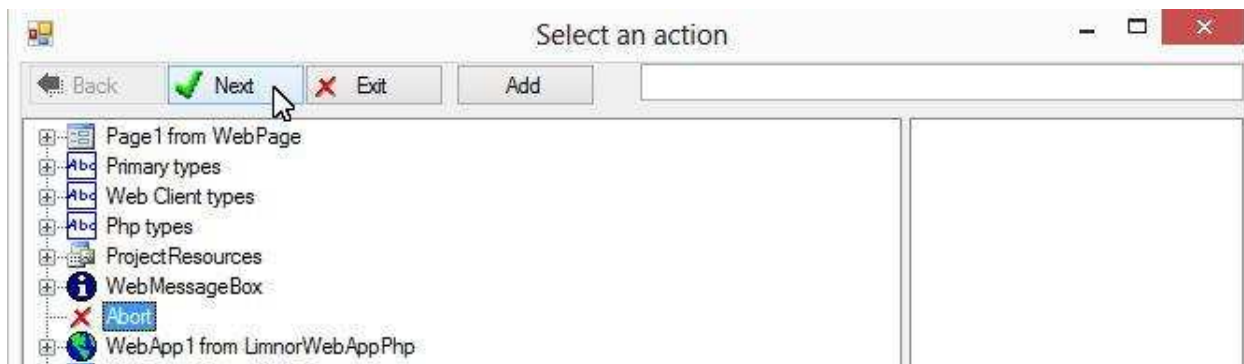
Use Abort in event handling

At the beginning of this chapter, we listed 8 tasks to be performed when the button is clicked. That is, we need to handle the click event of the button using those 8 tasks. For the first 3 tasks, we have created a method to validate user inputs. How to use this method to stop other tasks if the validations fail? This is where an Abort action should be used. Let's assign an Abort action to the onclick event of the button.

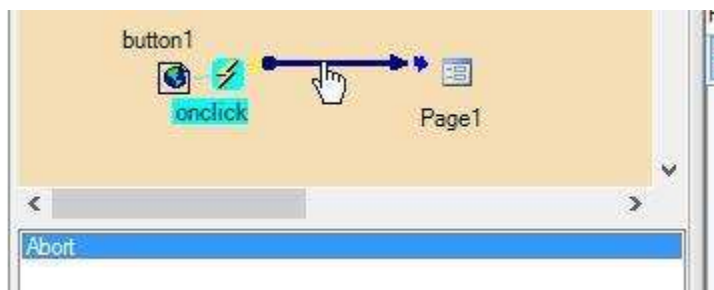
Right-click the button; choose "Assign action"; choose "onclick" event:



Select Abort action and click Next:

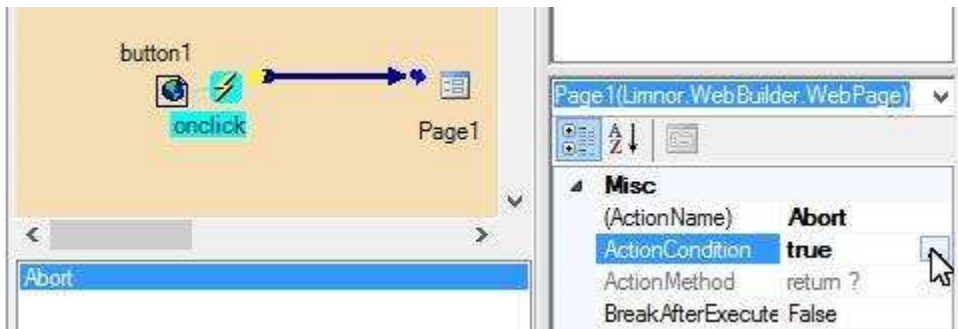


An Abort action is assigned to the event:

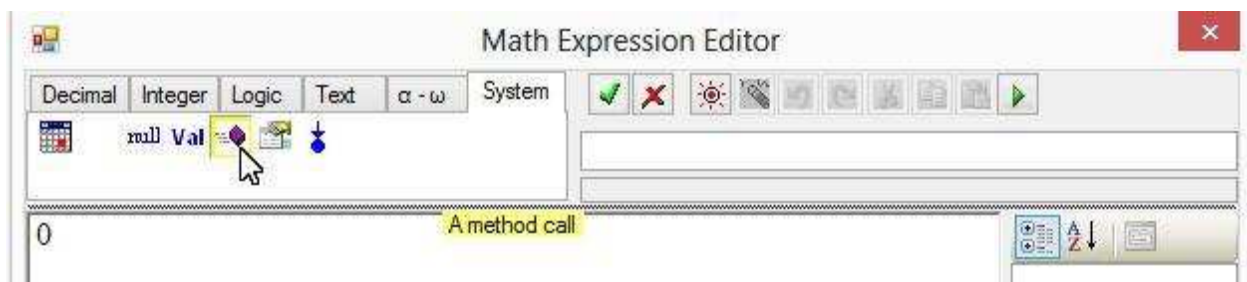


Note that we did not explicitly include the button for programming. When we did the above action assignment, the button is automatically included for programming; a unique id is given to the button; in our sample, it is "button1".

We need to set the ActionCondition of the Abort action to use the method userInputValidate:



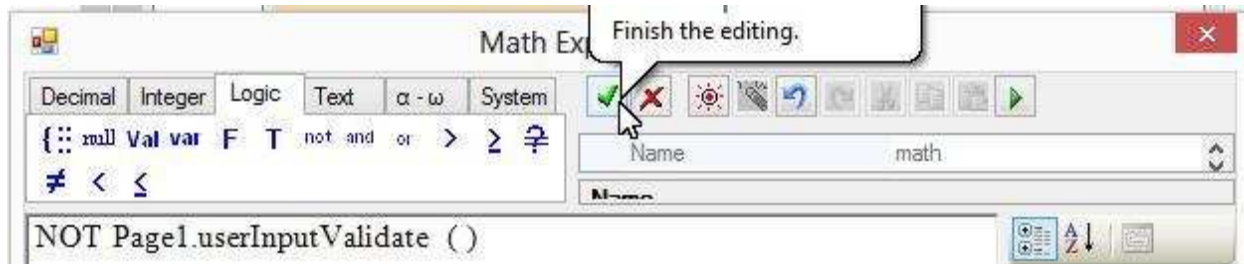
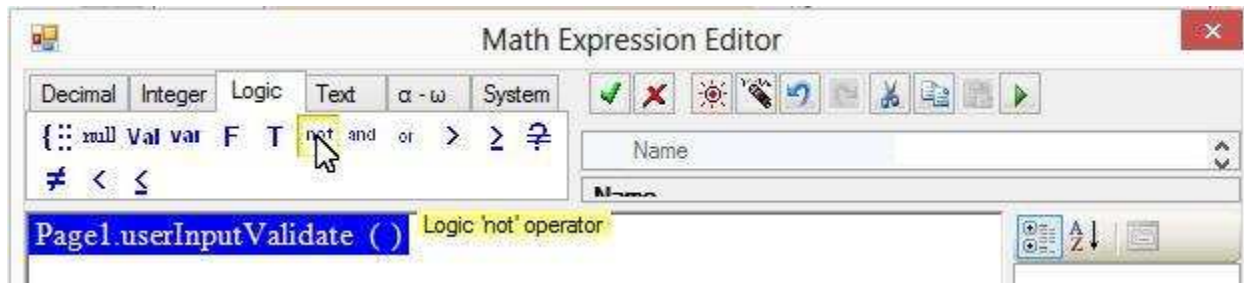
Click the method icon:



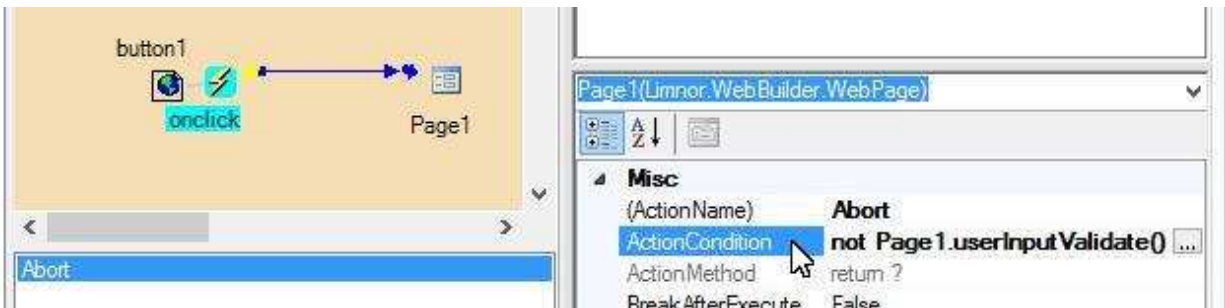
Select userInputValidate method:



Apply "not" operator:



We thus created the first part of event handling:



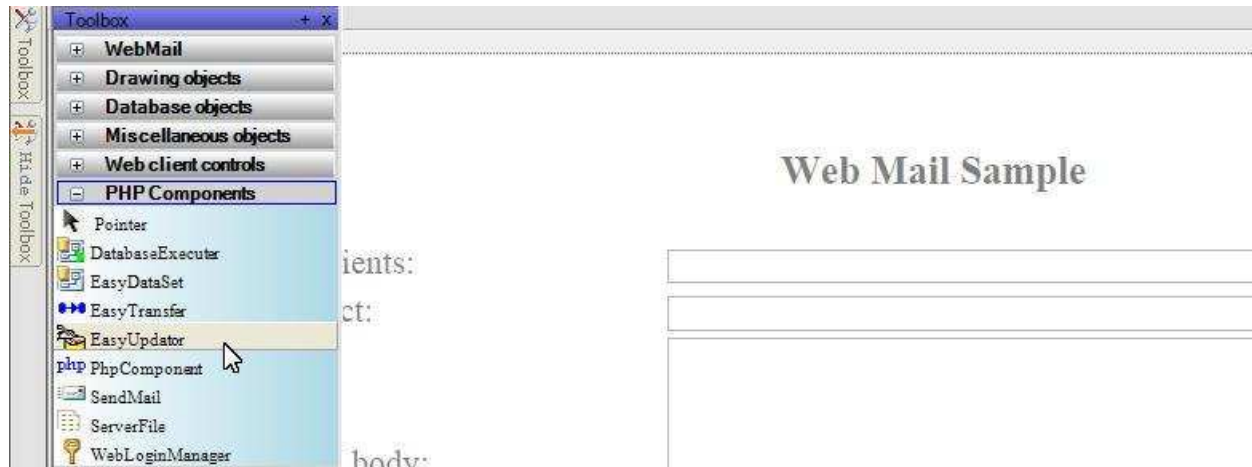
Insert Database Record

The 4th task is to create a new database record to record the web local time as the starting time, and record email parameters.

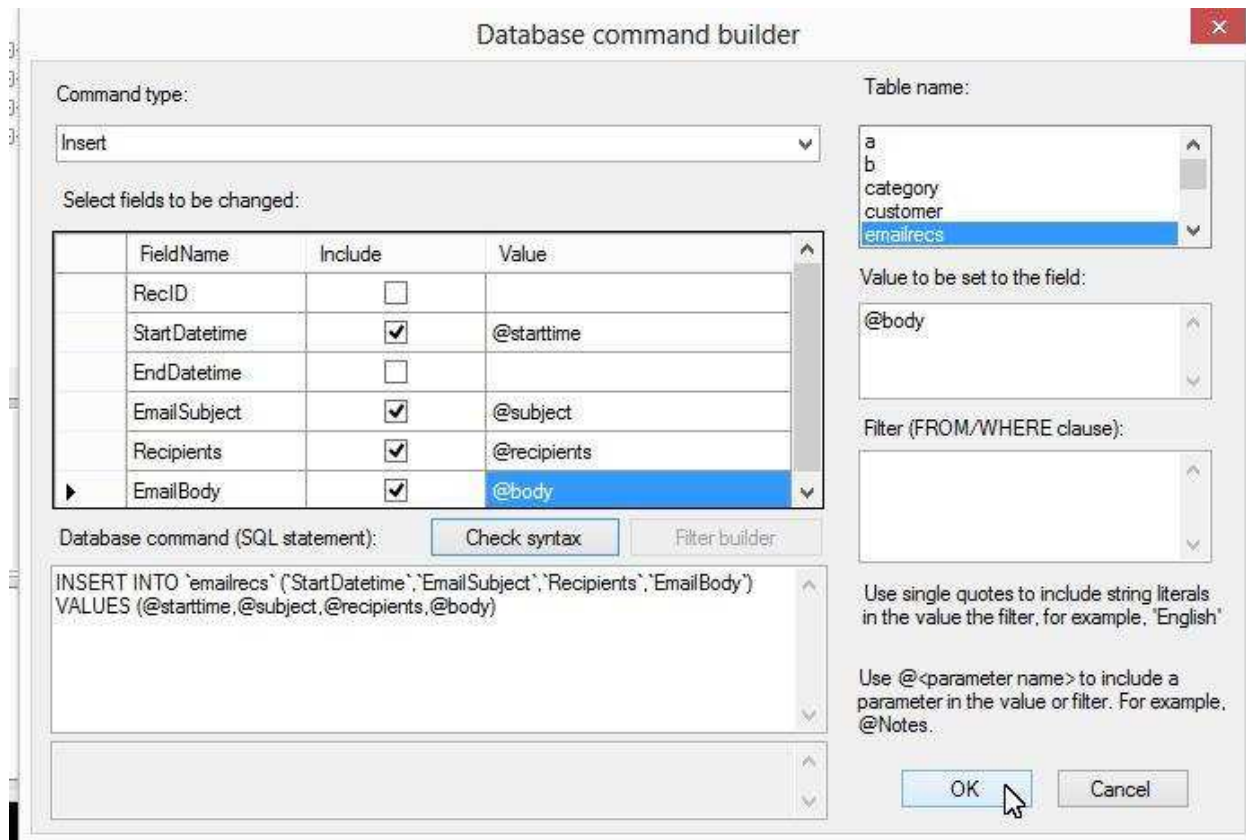
We may use an EasyUpdater to insert records into a database. To add an EasyUpdater to the web page, we need to switch to the Web Form Editor:



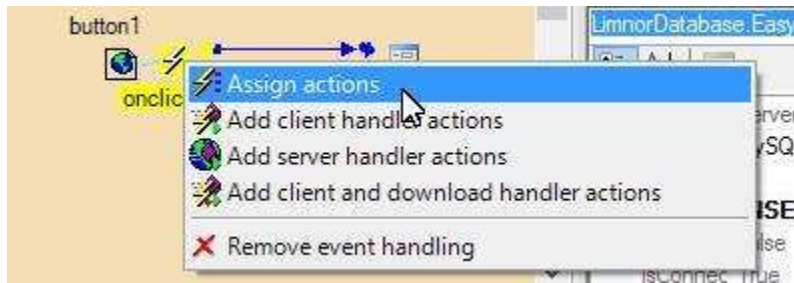
Drop EasyUpdater to the web page:



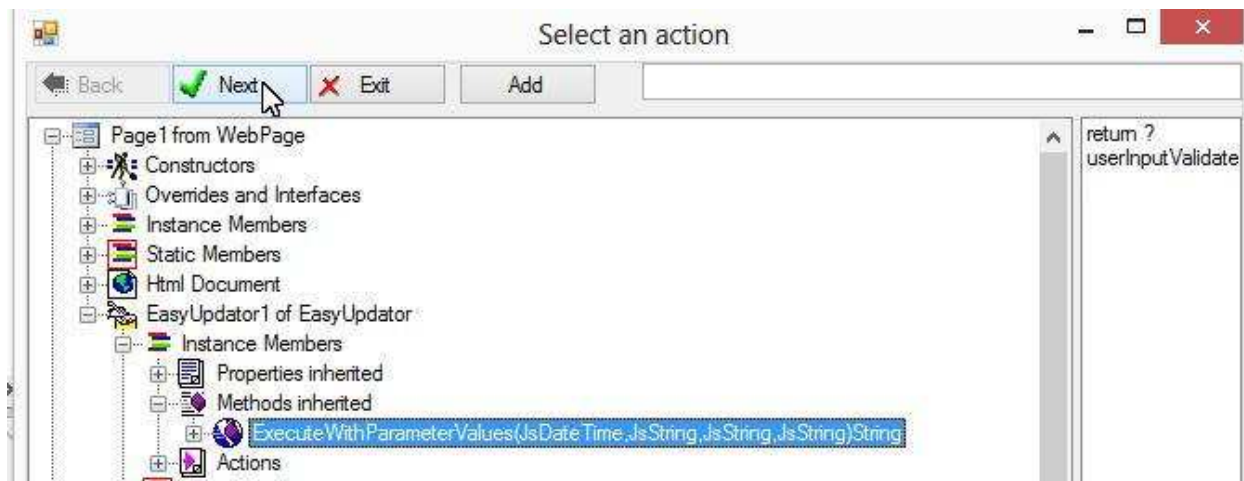
An EasyUpdater appears. Set its DatabaseConnection to connect to a database. Set its ExecutionCommand to prepare for inserting record:



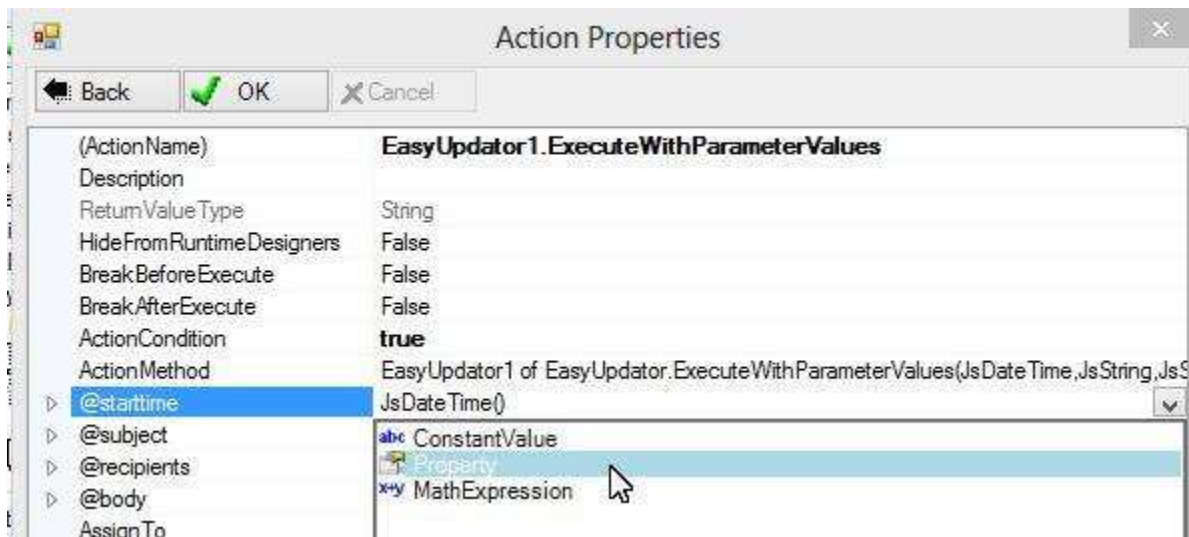
We may use this EasyUpdater in handling onclick of the button:

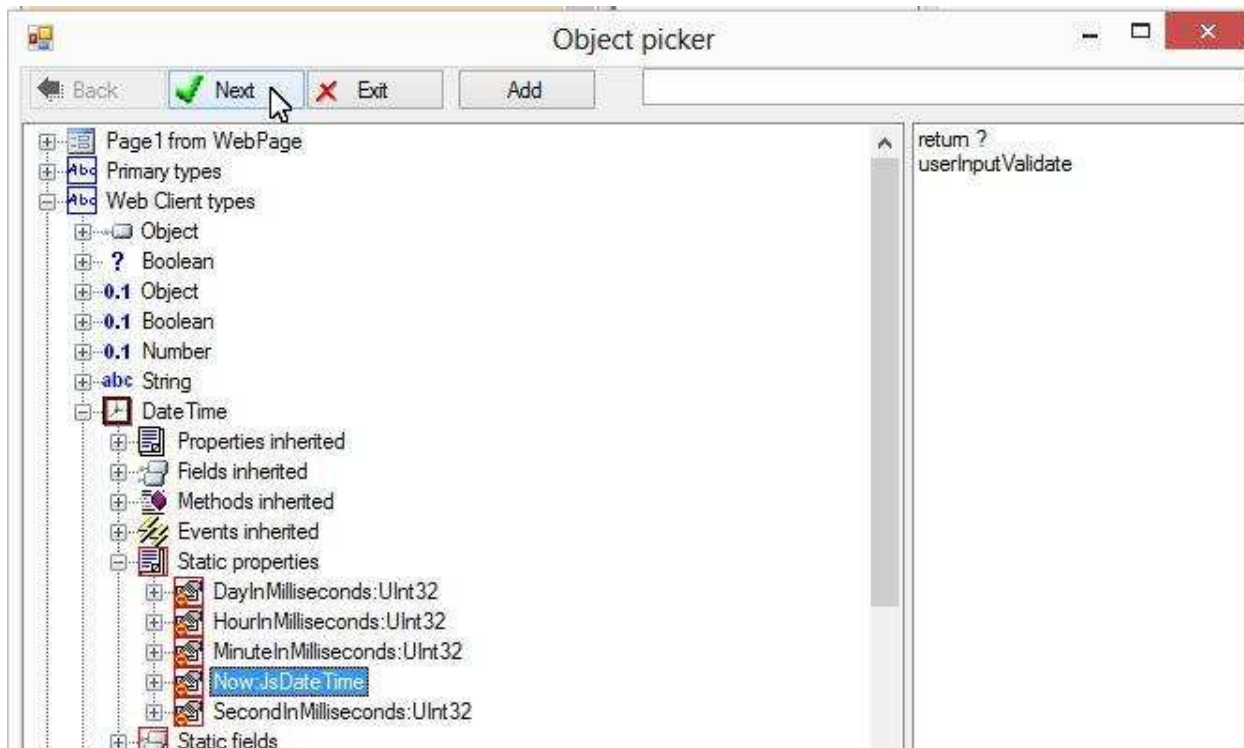


Select ExecuteWithParameterValues of EasyUpdater1:

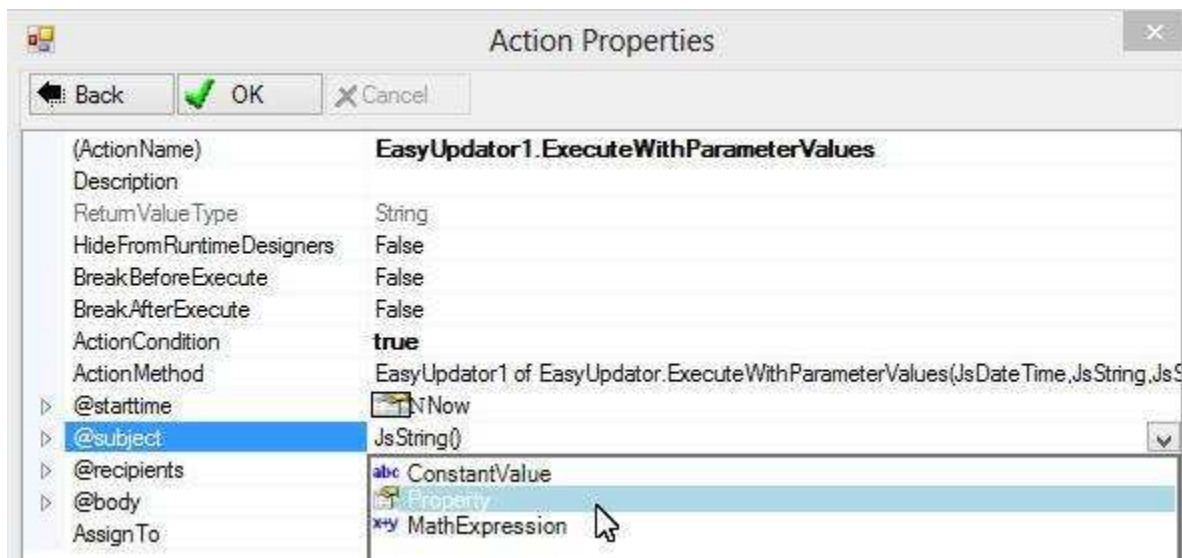


Pass "Now" to @starttime:

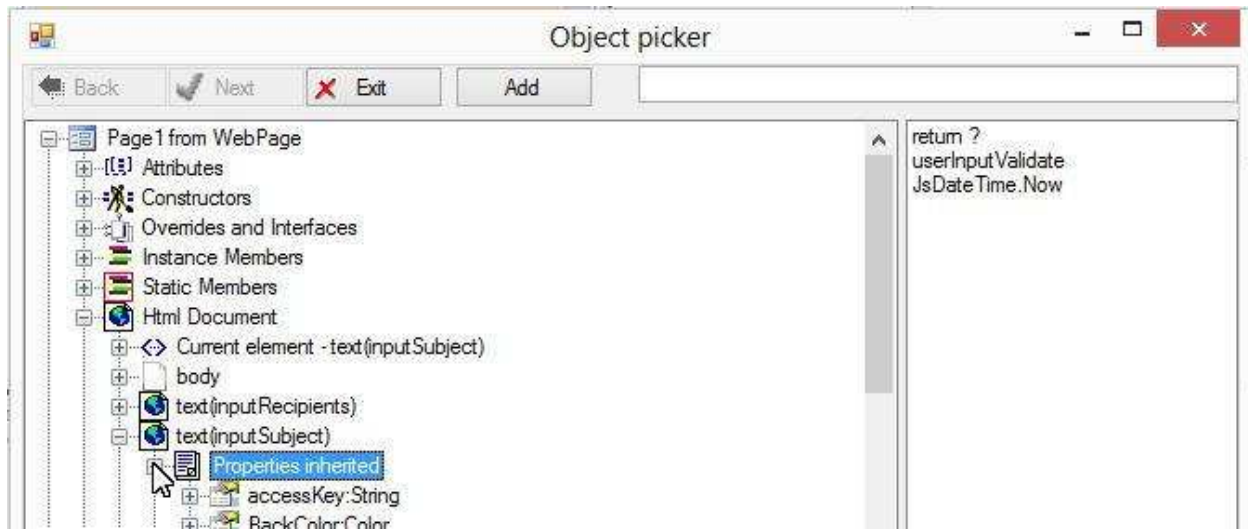




Pass “value” of the text box for email subject to @subject:

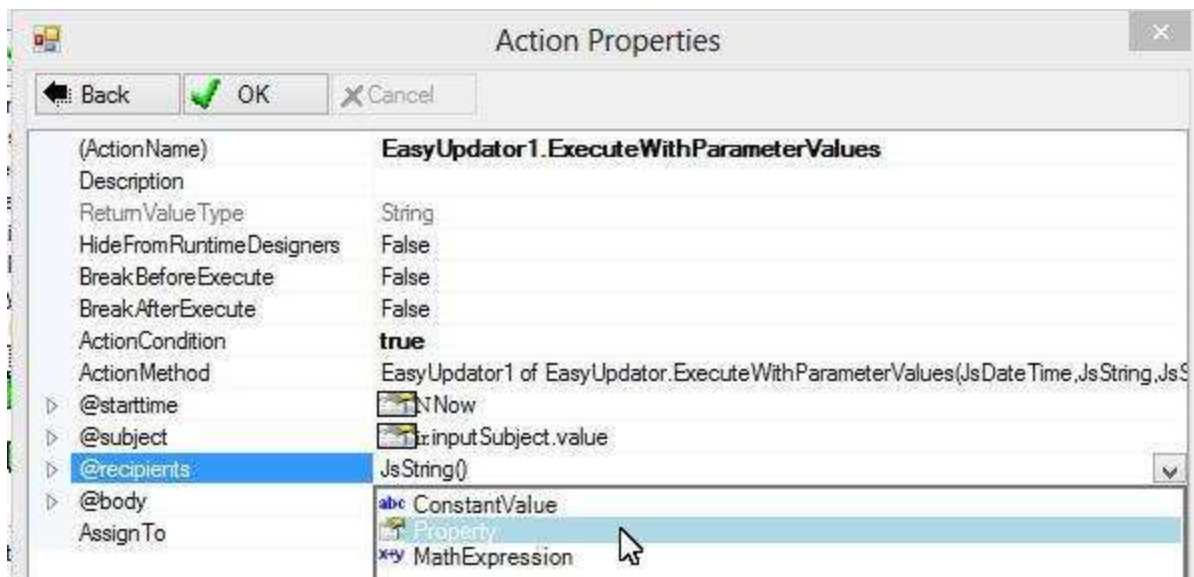


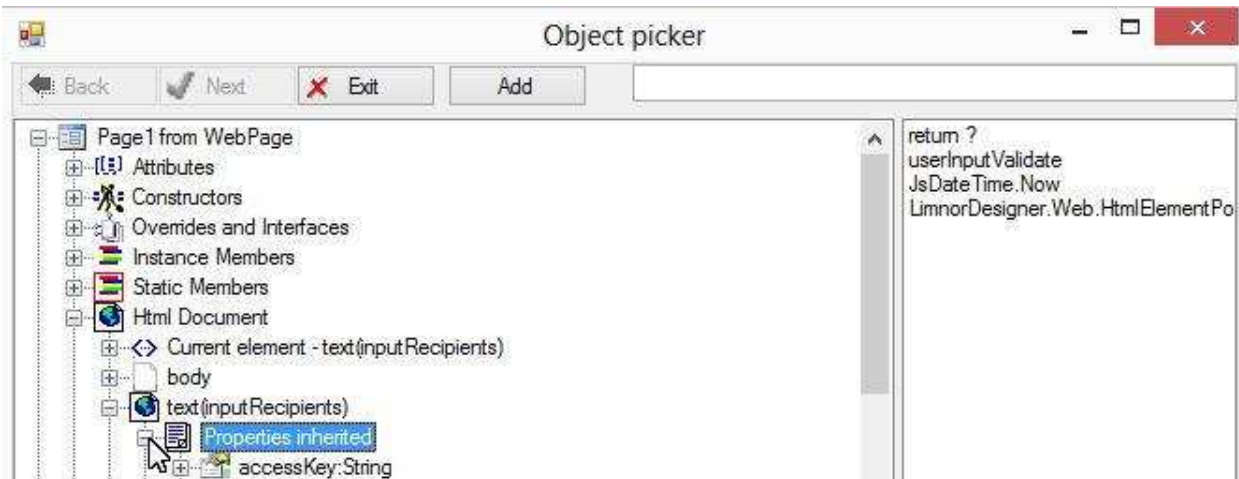
“Object picker” dialogue box appears. Because we have manually included text box elements for programming, we can see that those elements appear under “Html Document” so that we can access them. Thus we can select the “value” property of the text box for subject.



We thus use the subject text box.

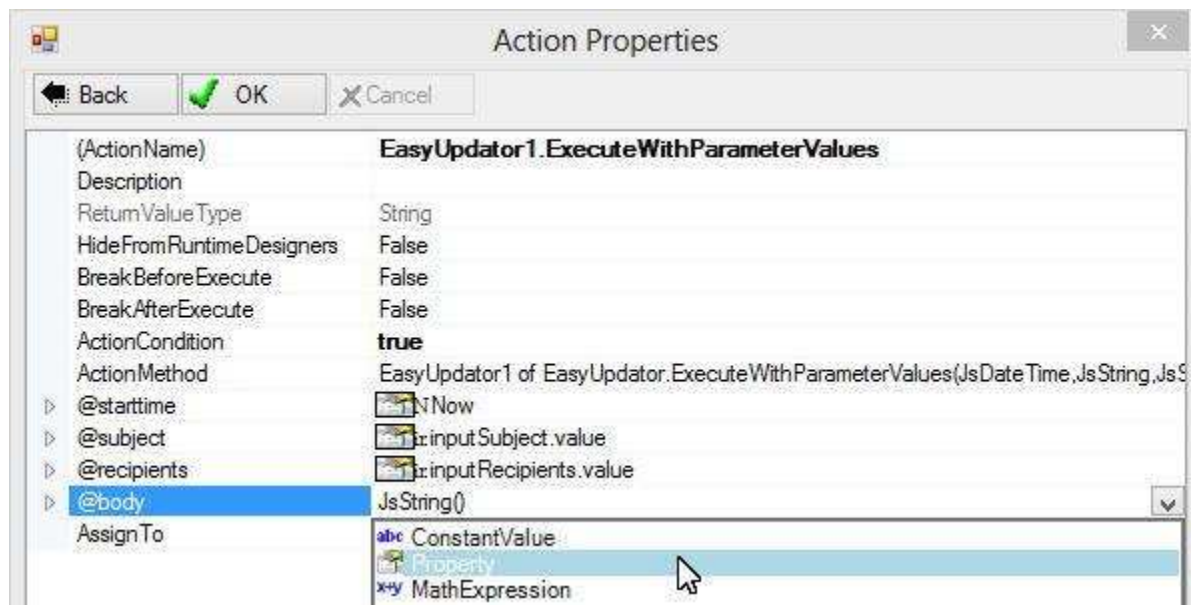
Pass "value" of the text box for email recipients to @recipients:

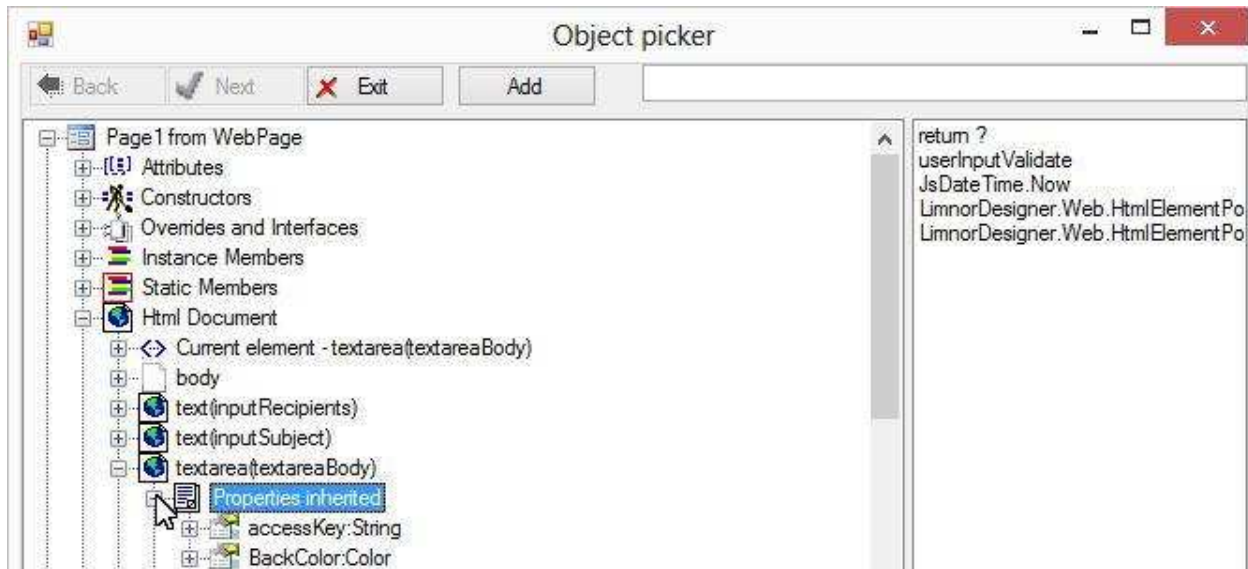




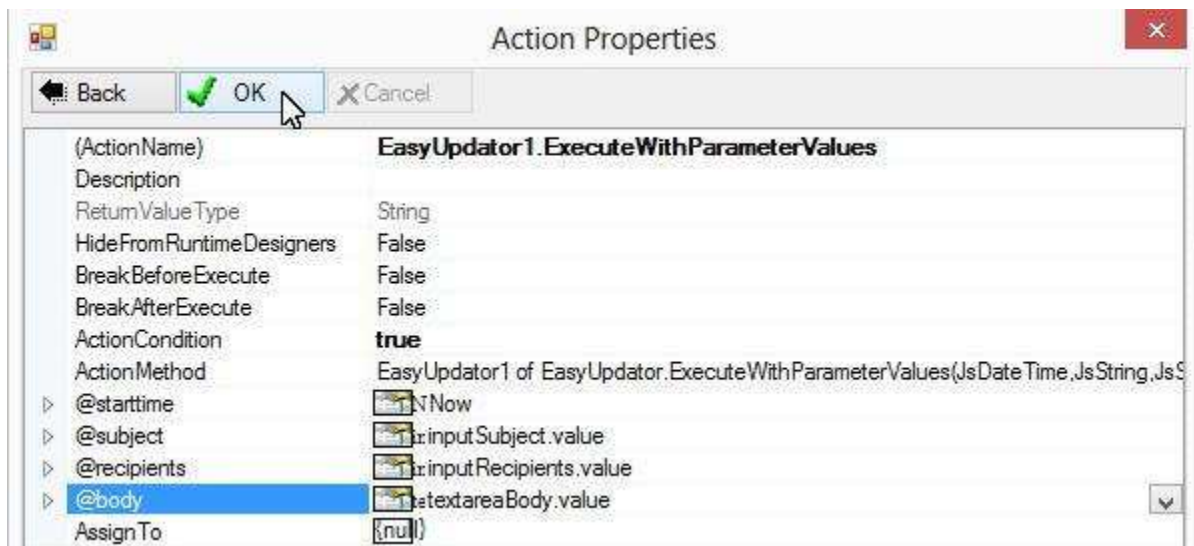
We thus used the text box for recipients.

Pass "value" of the text area for email body to @body:

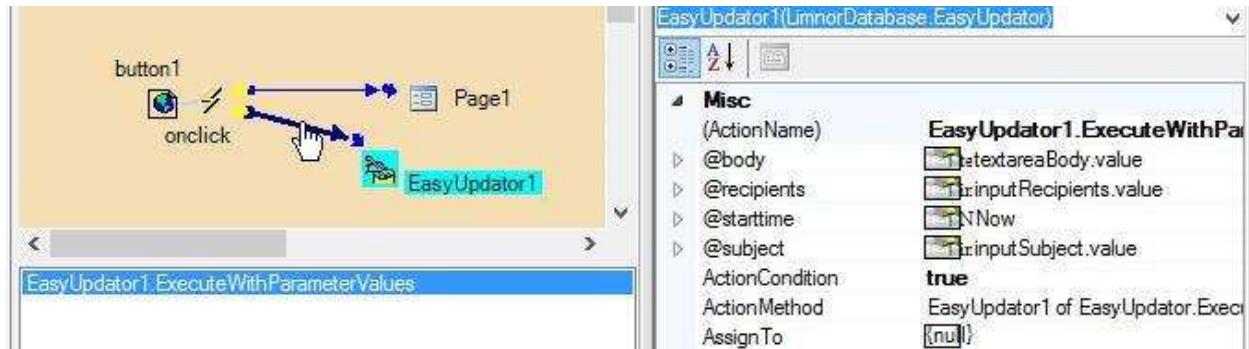




Click OK to finish creating the action:



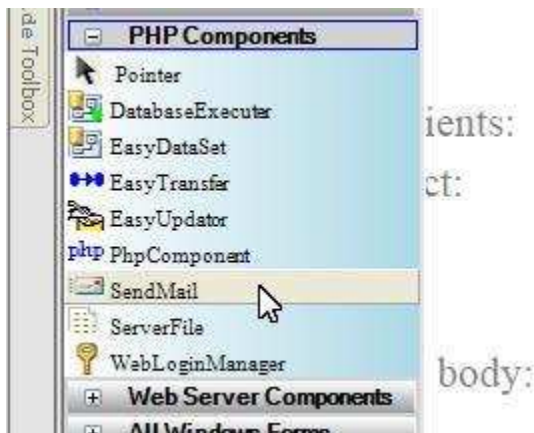
The action is created and assigned to onclick event of the button:



Note that we selected the “value” property of the text boxes for action parameters. Those text boxes appear in the Object Explorer for us to select because we have manually included those text boxes for programming.

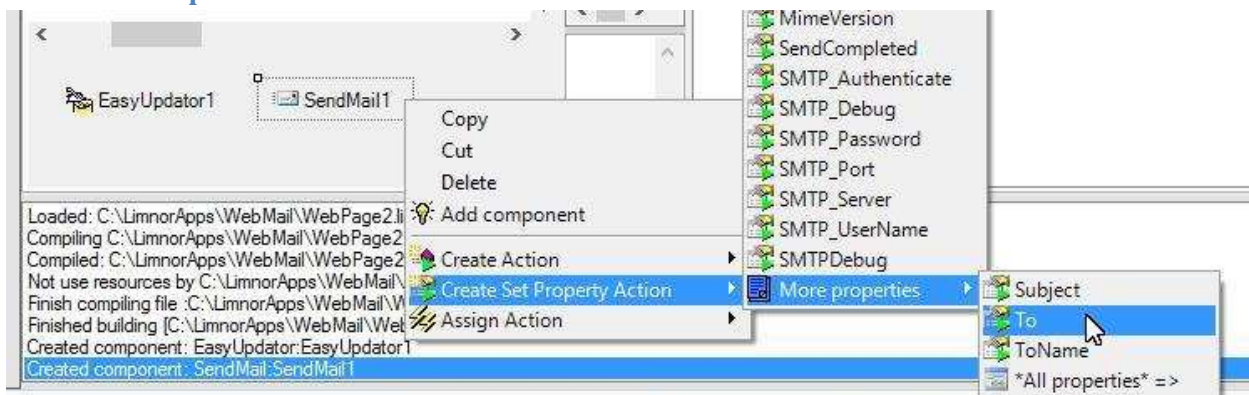
Send email

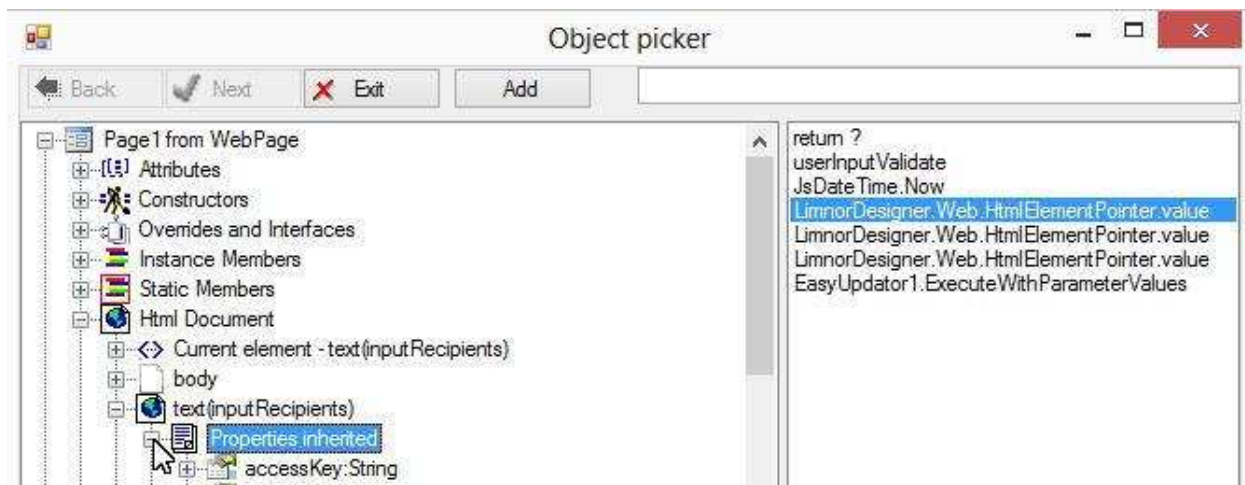
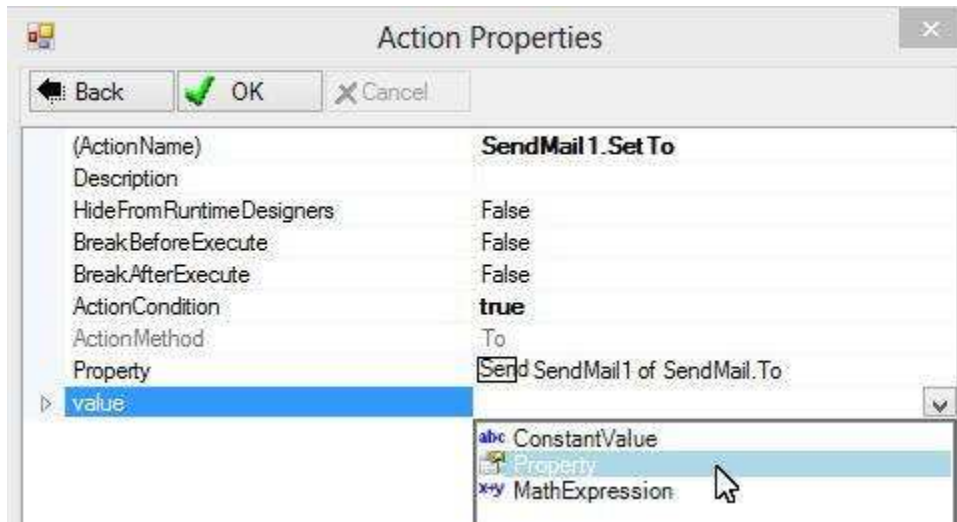
The 5th task is to send an email to the recipients. We may add an email sender component to the web page to do that.

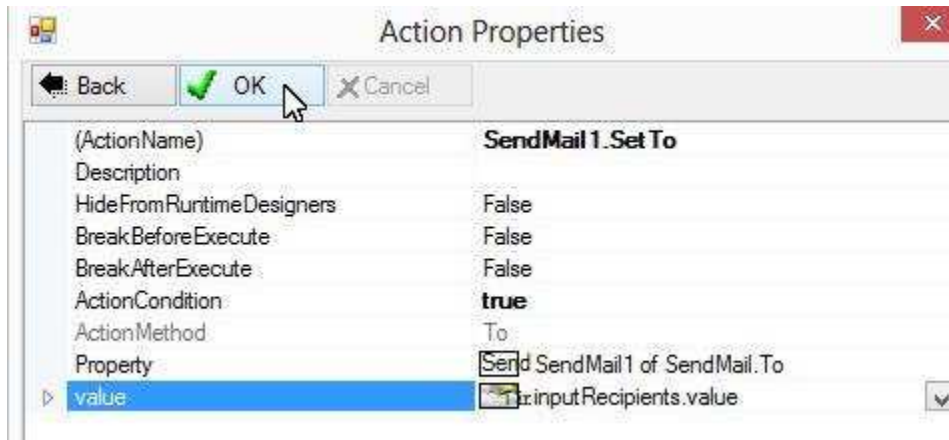


We may set properties “To”, “Subject”, and “Body” of the email sender to the “value” property of the text boxes.

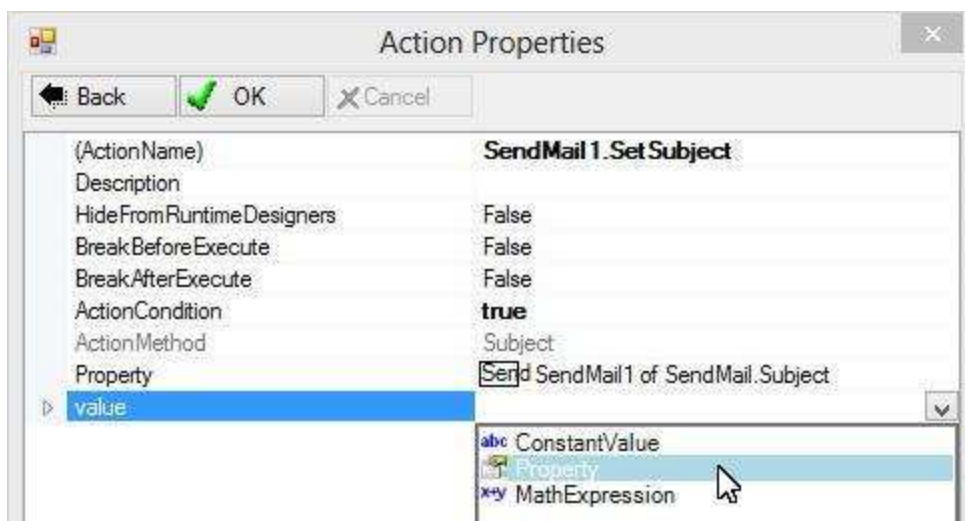
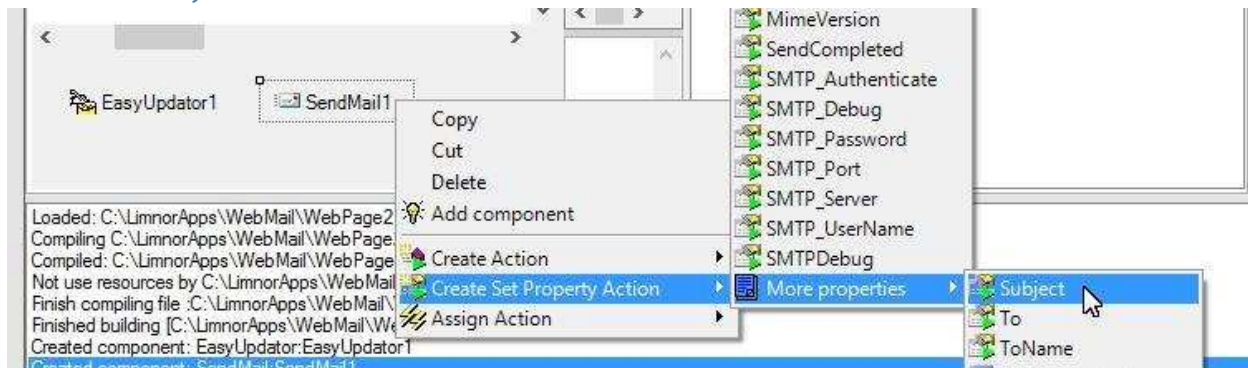
Set email recipients

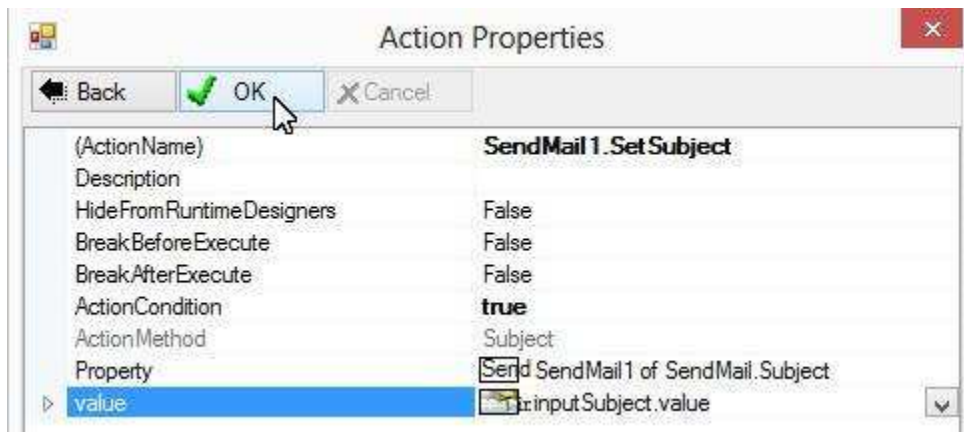
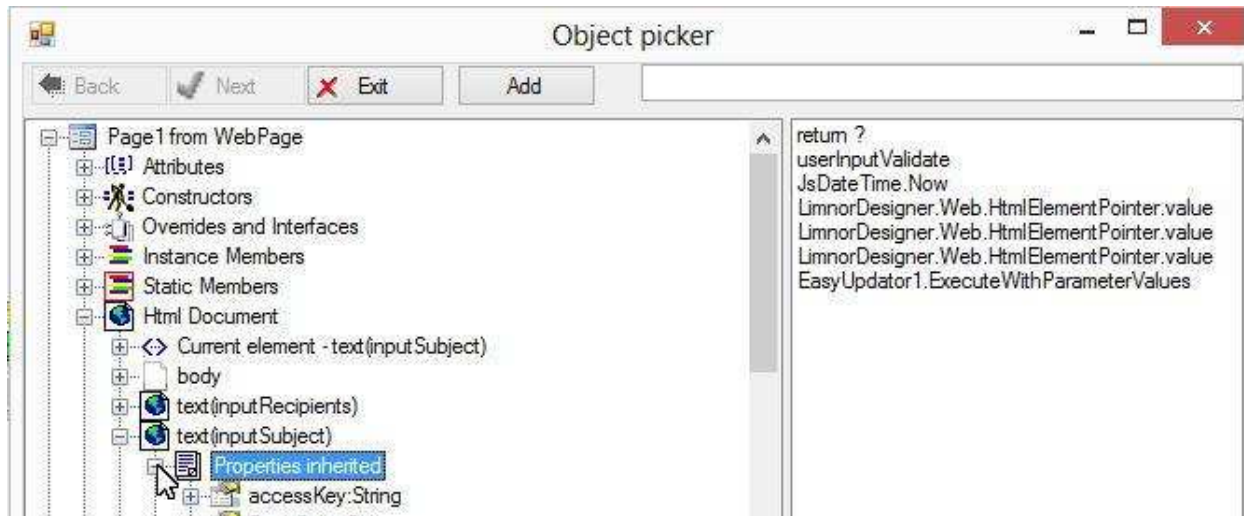




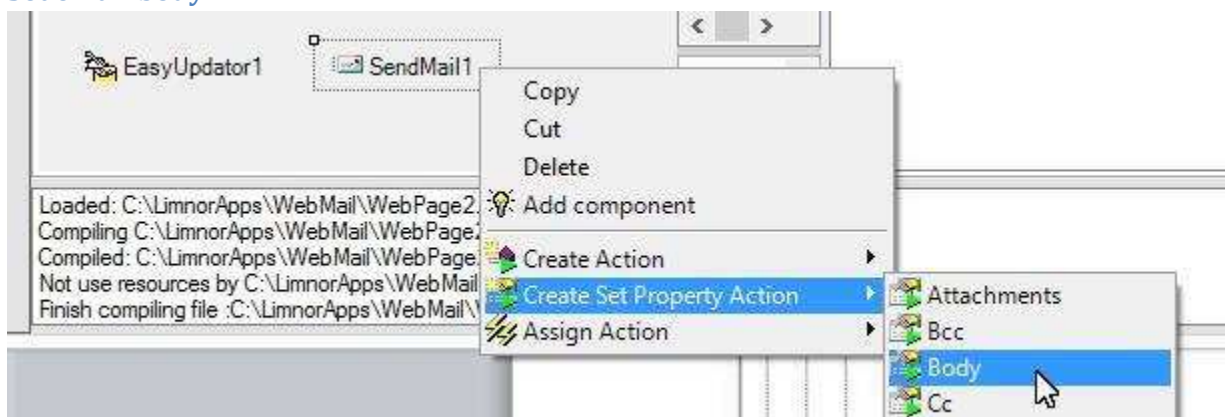


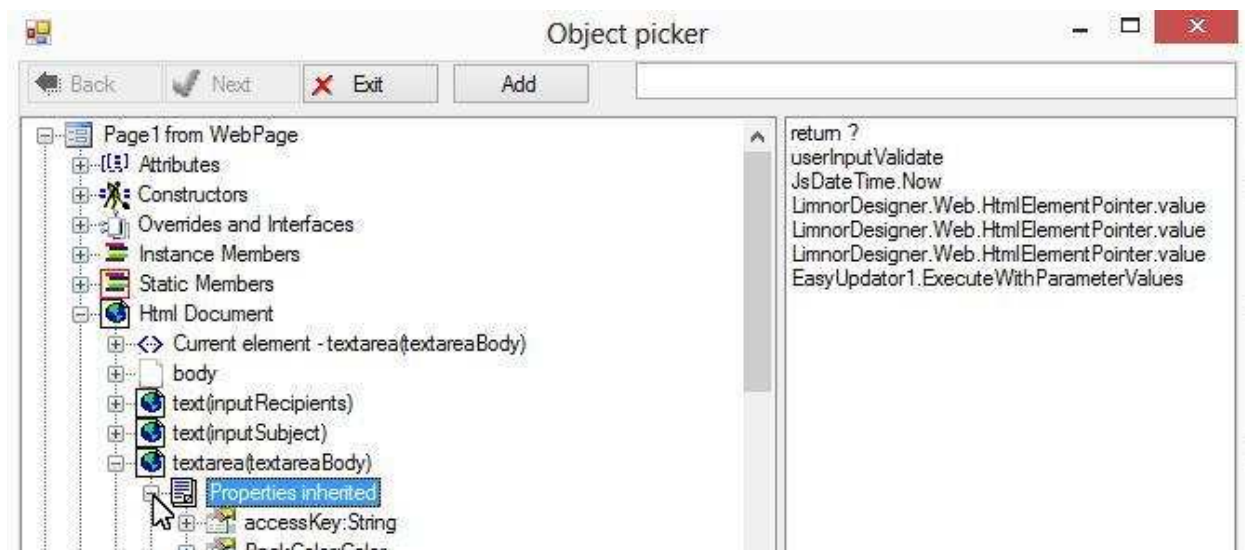
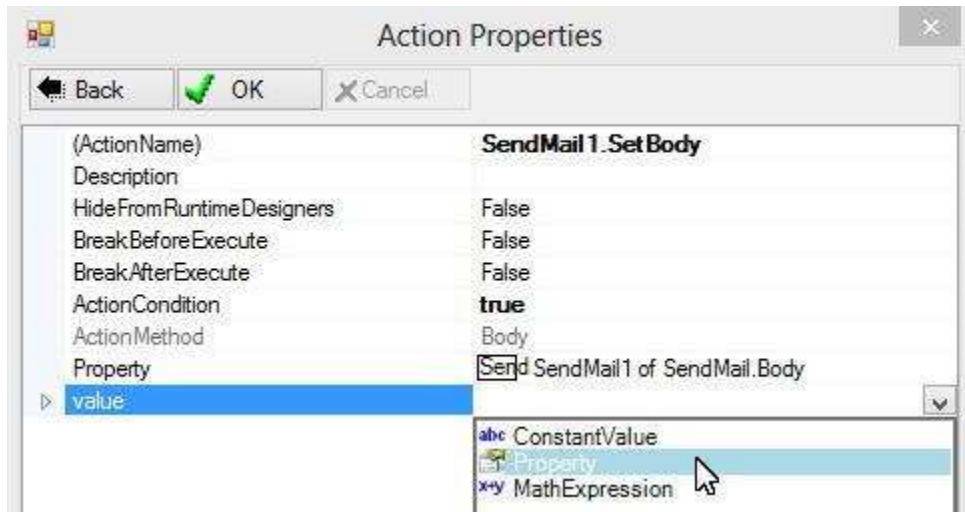
Set email subject

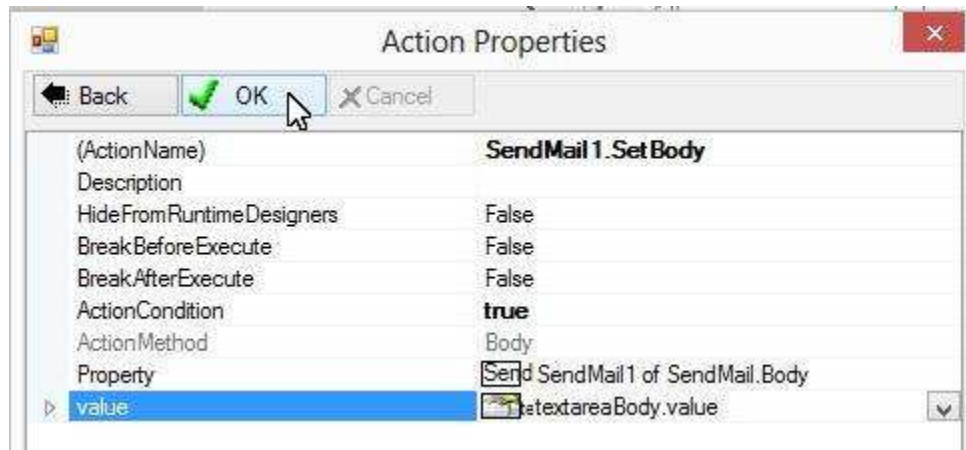




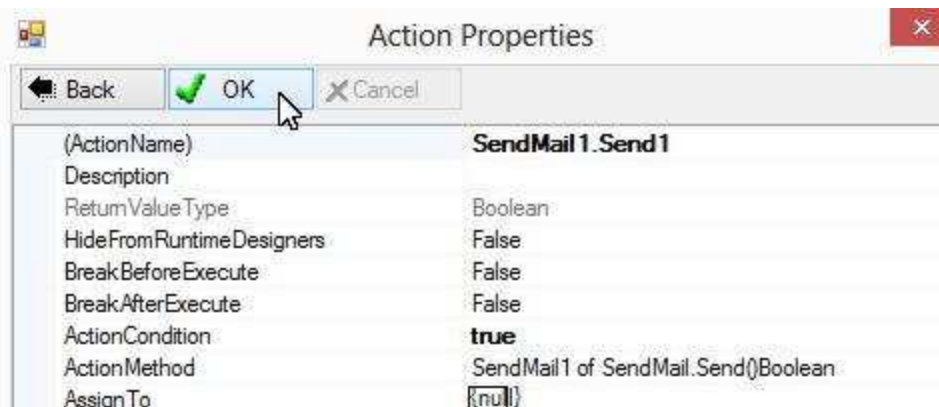
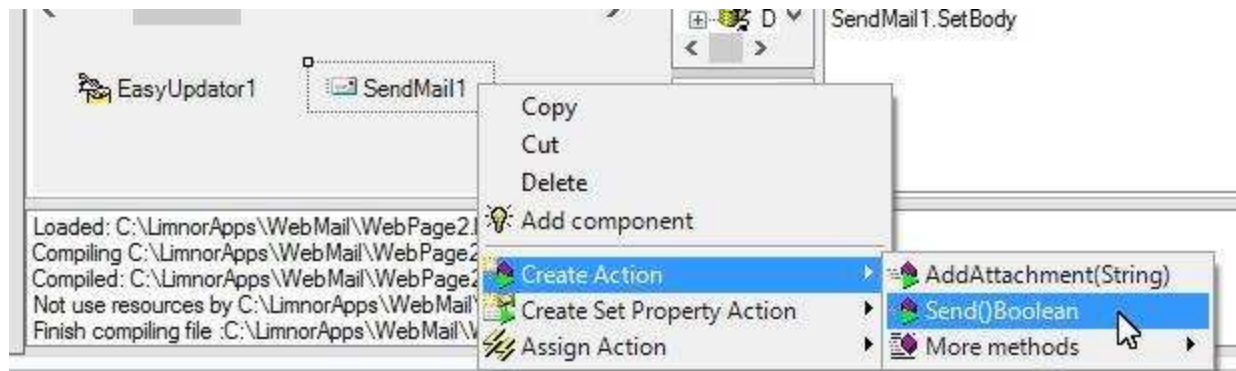
Set email body



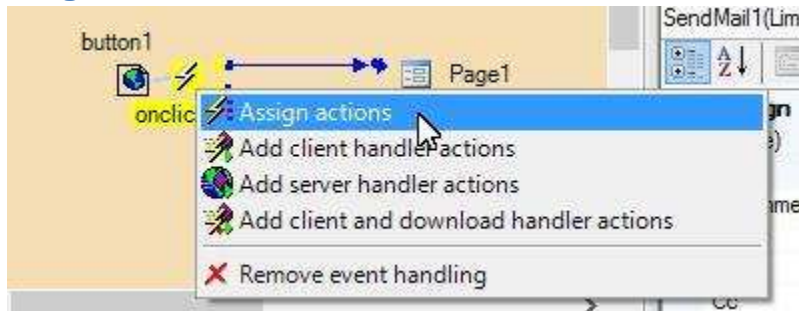




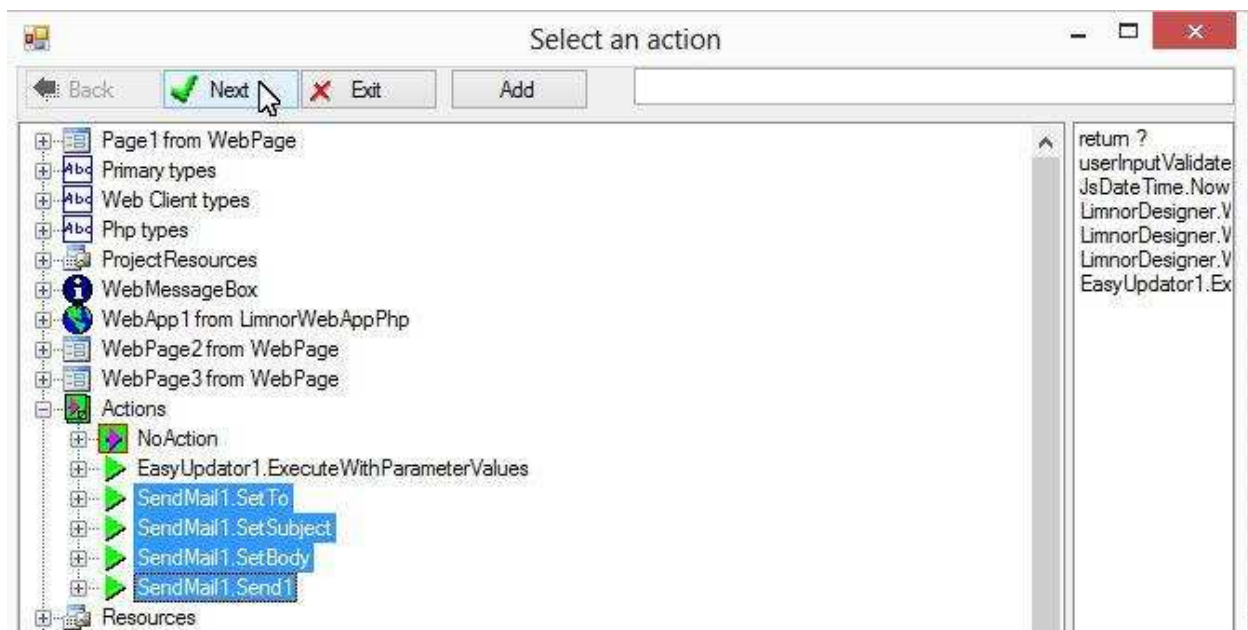
Send email



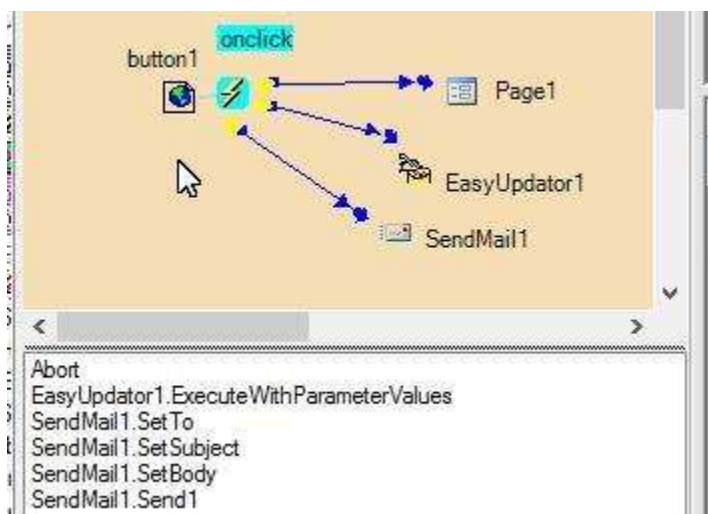
Assign actions to button



Select the email related actions we just created:



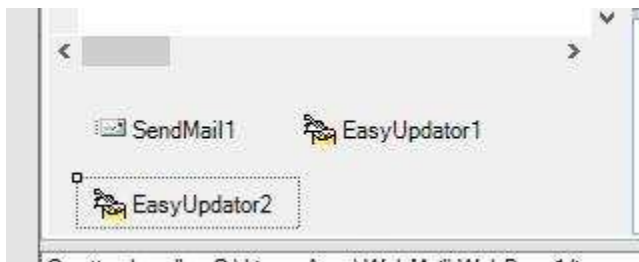
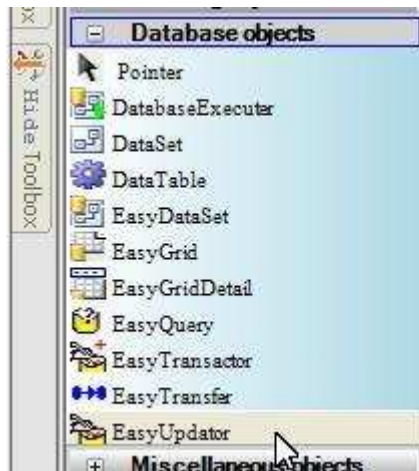
The actions are assigned to the button:



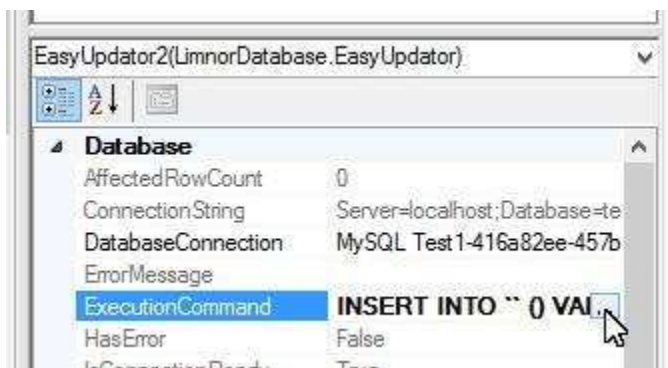
Save Error Message to Database

The email sender, SendMail1, has a property, ErrorMessage, holding the error message if the Send action fails. We want to save the contents of this property to a field of the record created by the 4th task.

We add another EasyUpdater to do this task.



Set its ExecutionCommand property to update the record:



Database command builder

Command type: Update

Table name: emailrecs

Select fields to be changed:

Field Name	Include	Value
EndDatetime	<input type="checkbox"/>	
EmailSubject	<input type="checkbox"/>	
Recipients	<input type="checkbox"/>	
EmailBody	<input type="checkbox"/>	
ErrorMessage	<input checked="" type="checkbox"/>	@message

Value to be set to the field: @message

Filter (FROM/WHERE clause):

Database command (SQL statement):
 UPDATE 'emailrecs' SET 'ErrorMessage'=@message

Check syntax Filter builder

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

Set filter to RecID=@id to locate the record to be updated:

Database command builder

Command type: Update

Table name: emailrecs

Select fields to be changed:

Field Name	Include	Value
RecID	<input type="checkbox"/>	
StartDatetime	<input type="checkbox"/>	
EndDatetime	<input type="checkbox"/>	
EmailSubject	<input type="checkbox"/>	
Recipients	<input type="checkbox"/>	
EmailBody	<input type="checkbox"/>	

Value to be set to the field:

Filter (FROM/WHERE clause):
 RecID=@id

Database command (SQL statement):
 UPDATE 'emailrecs' SET 'ErrorMessage'=@message WHERE RecID=@id

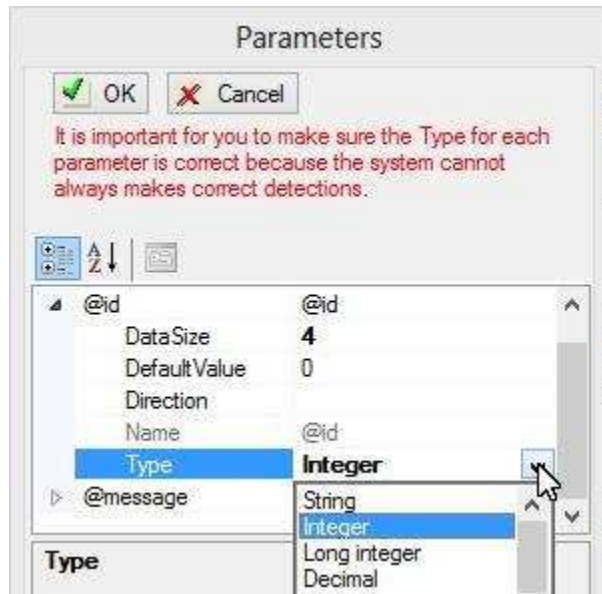
Check syntax Filter builder

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

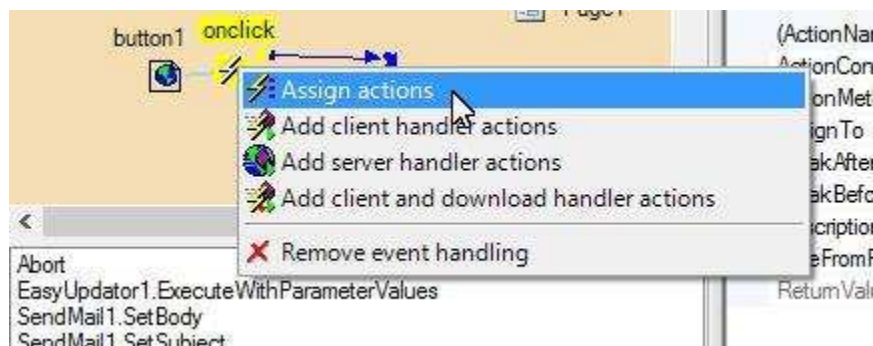
Use @<parameter name> to include a parameter in the value or filter. For example, @Notes.

OK Cancel

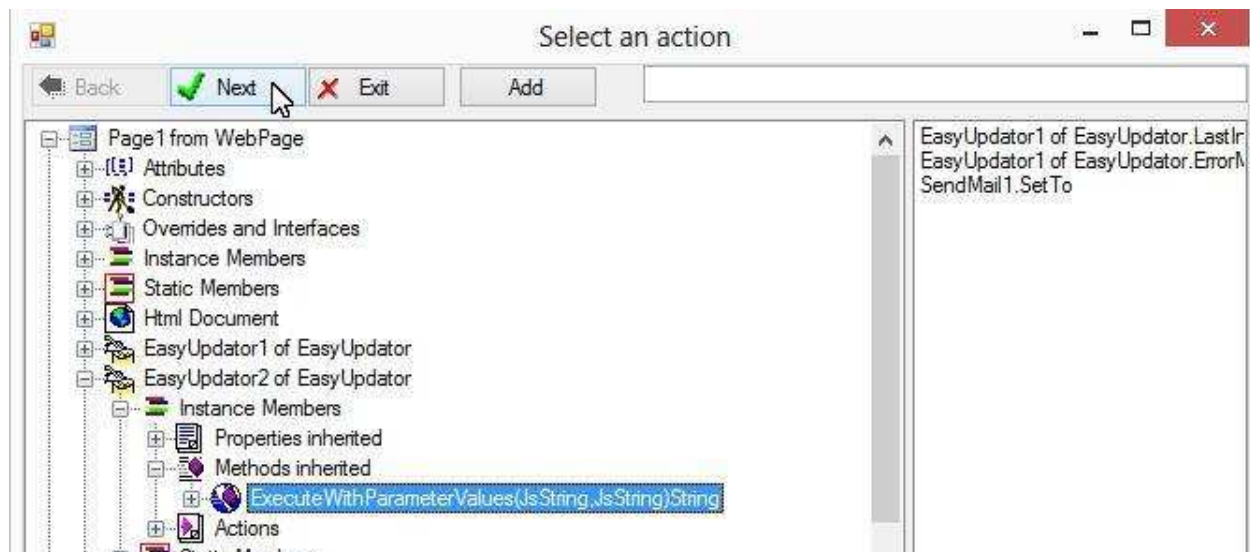
Make sure @id is an integer:



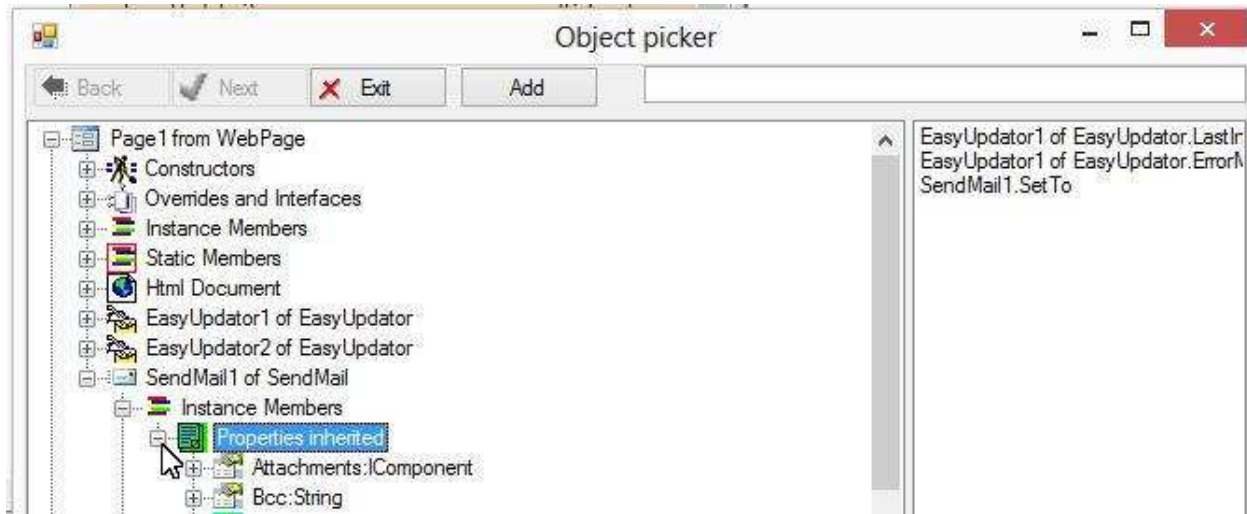
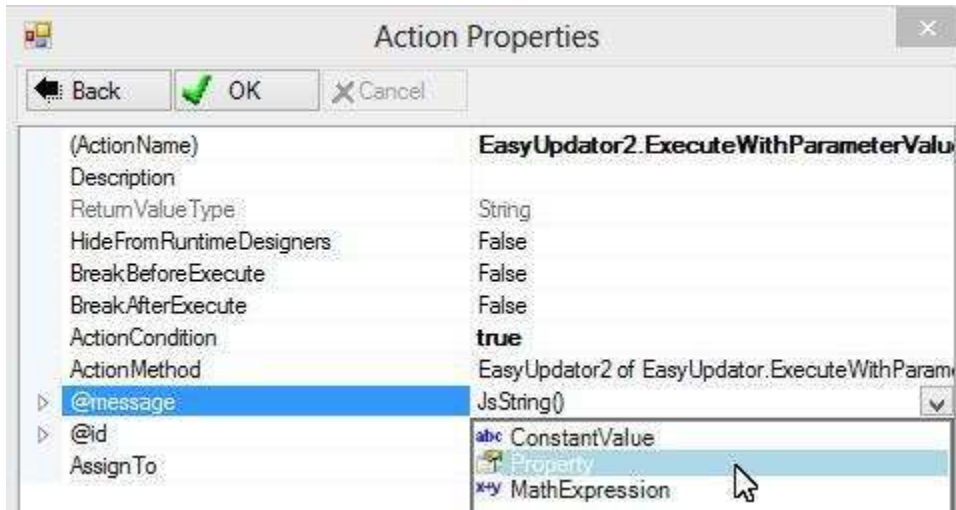
Create a new action for the button's onclick event:



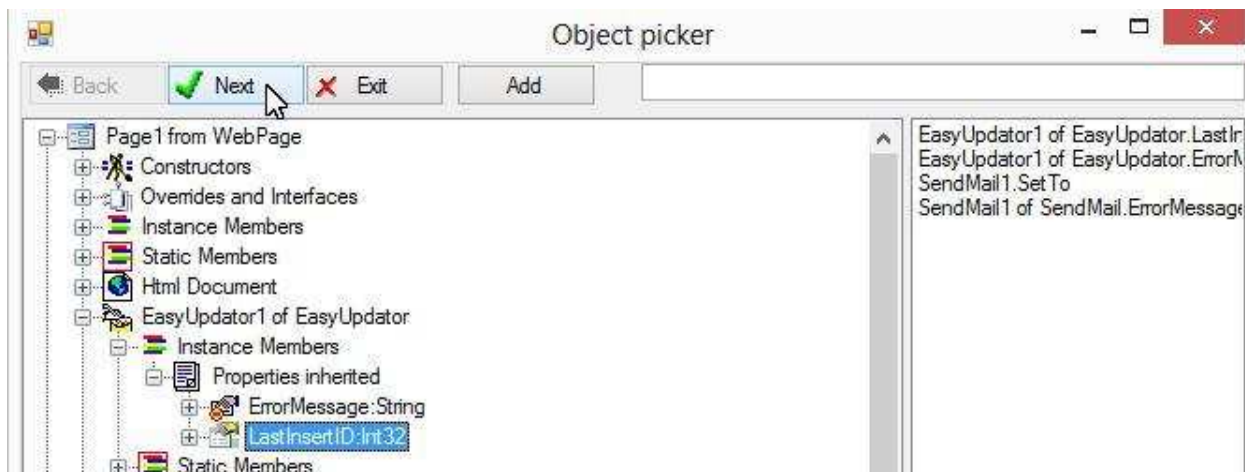
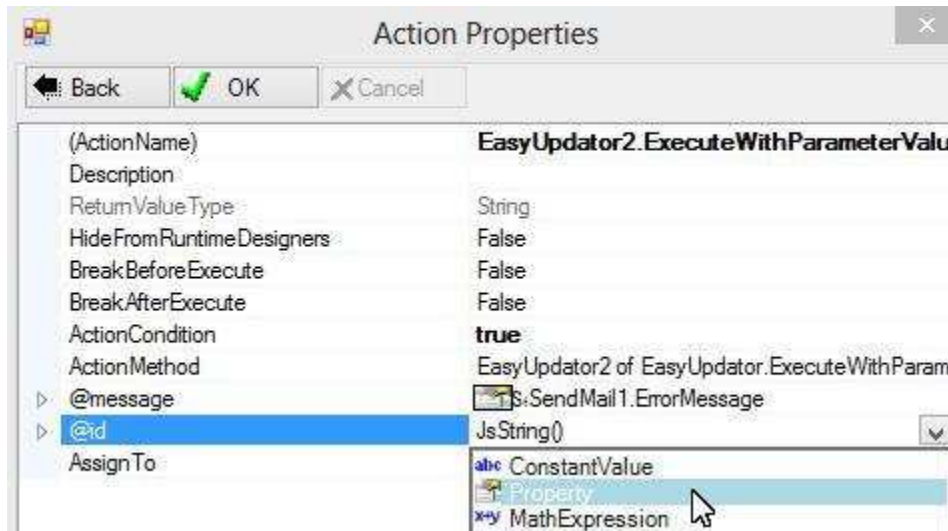
Select ExecuteWithParameterValues of EasyUpdator2:



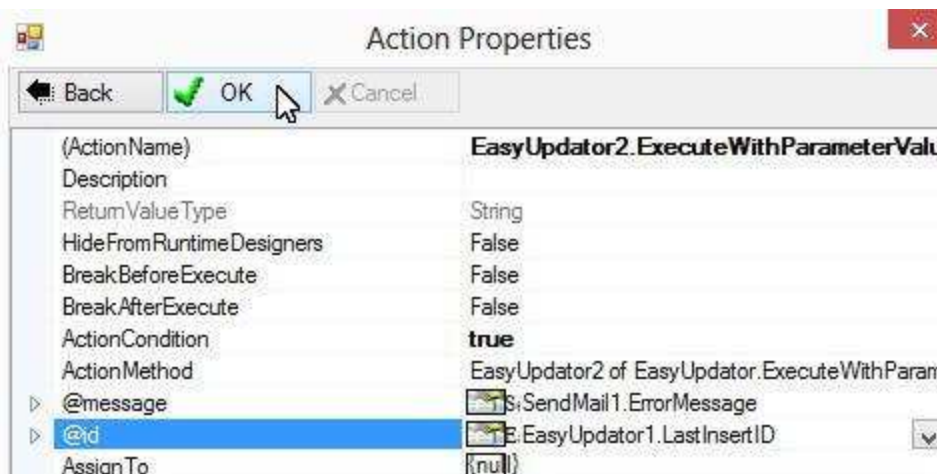
Pass ErrorMessage of SendMail1 to @message:



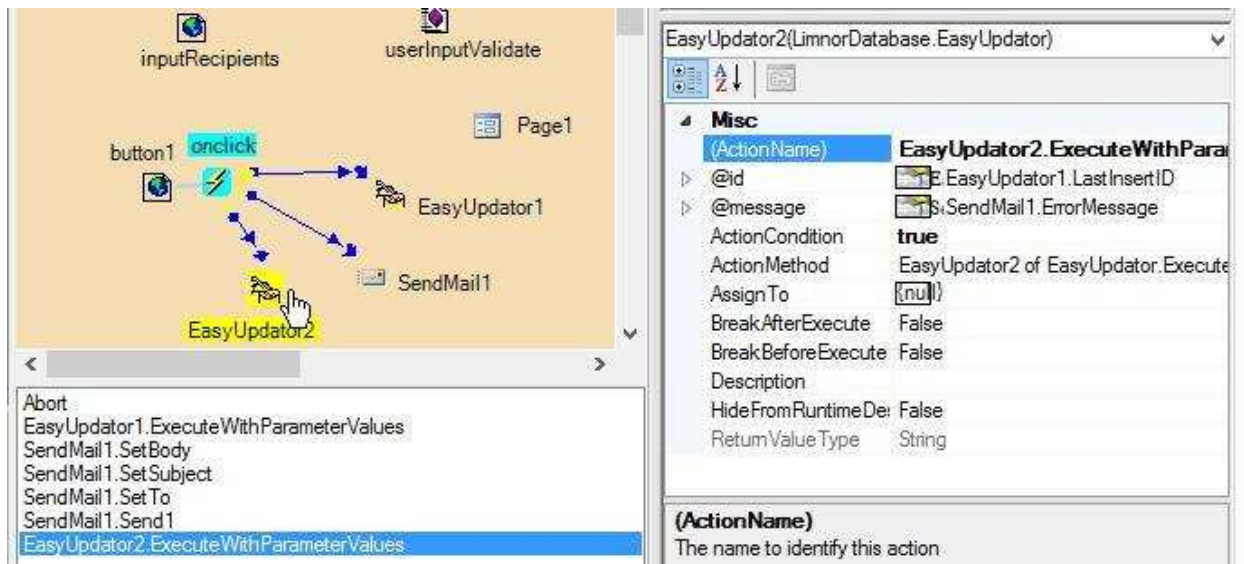
Pass LastInsertID property of EasyUpdator1 to @id. When we use EasyUpdator1 to create a new record, LastInsertID of EasyUpdator1 is the new auto number of the new record. We use this number to locate the record we want to update with EasyUpdator2:



Click OK to finish creating the action:

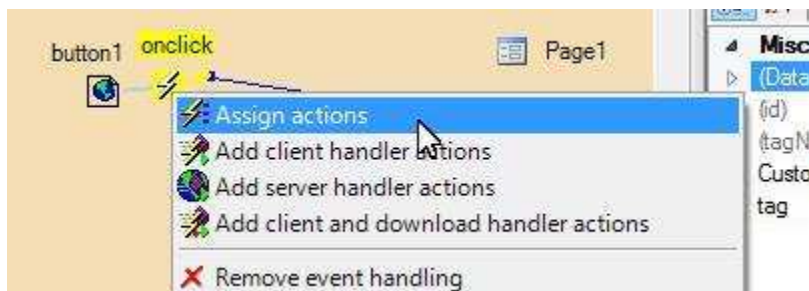


The action is created and assigned to the button:

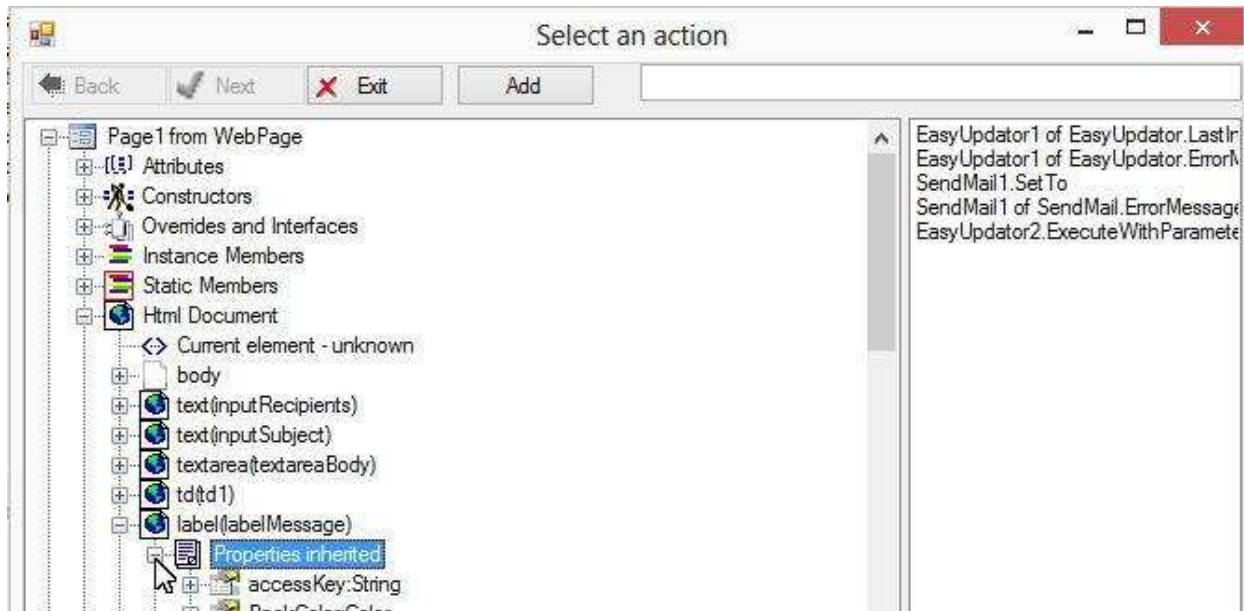


Show error message on web page

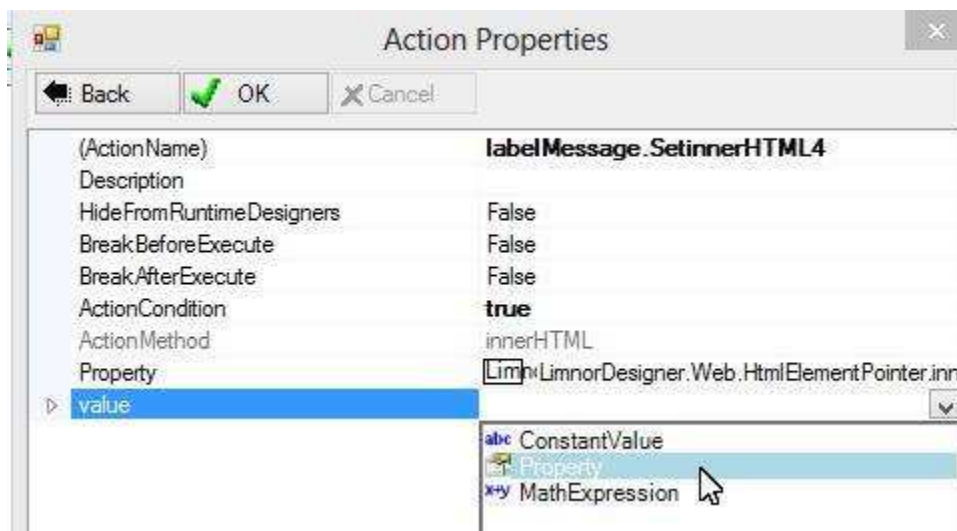
Besides saving the ErrorMessage of SendMail1 to database, we also want to show the contents of this property to the web page element for status message. That is our task 7.

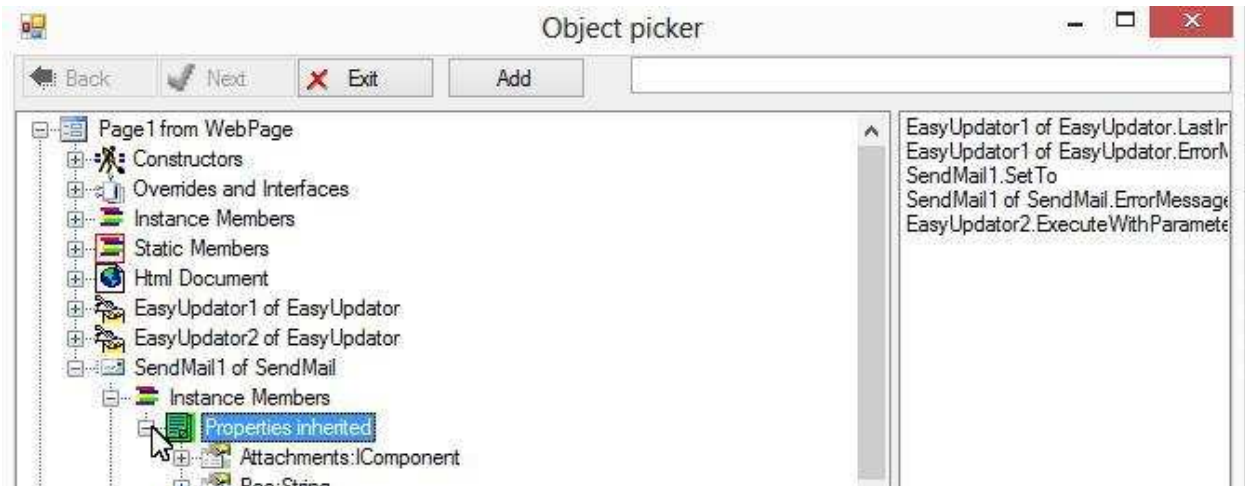


Choose innerHTML property of the label element:

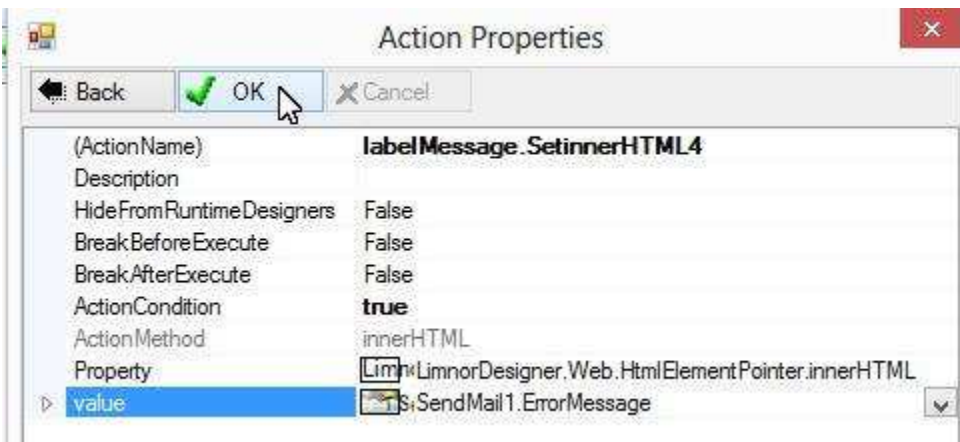


Pass the ErrorMessage of SendMail1 to “value” of the action:

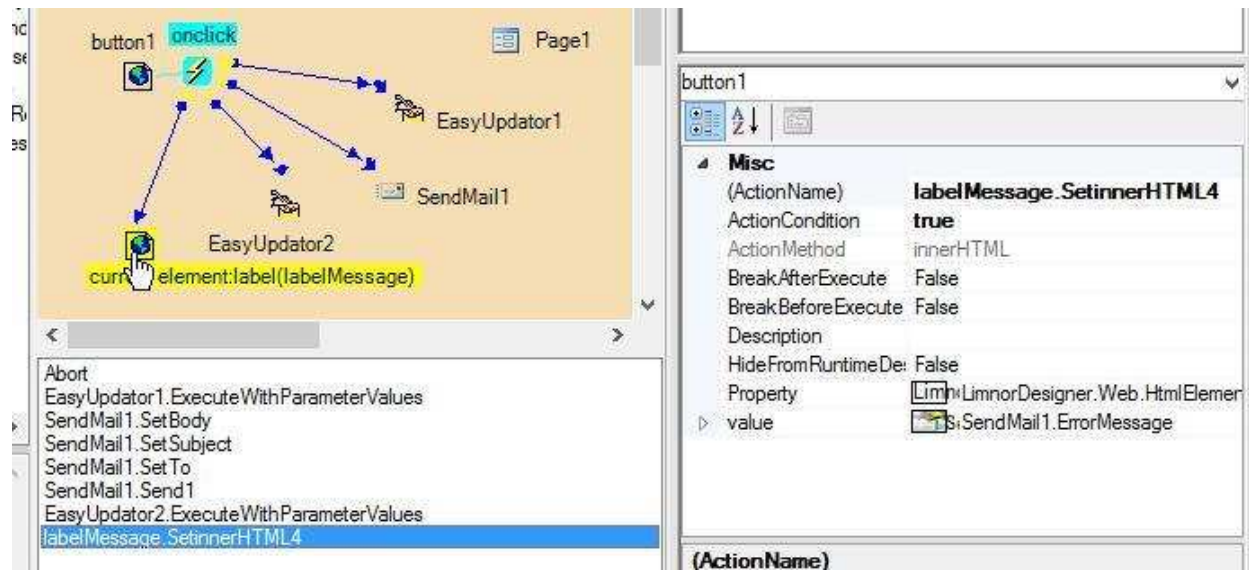




Click OK to finish creating the action:



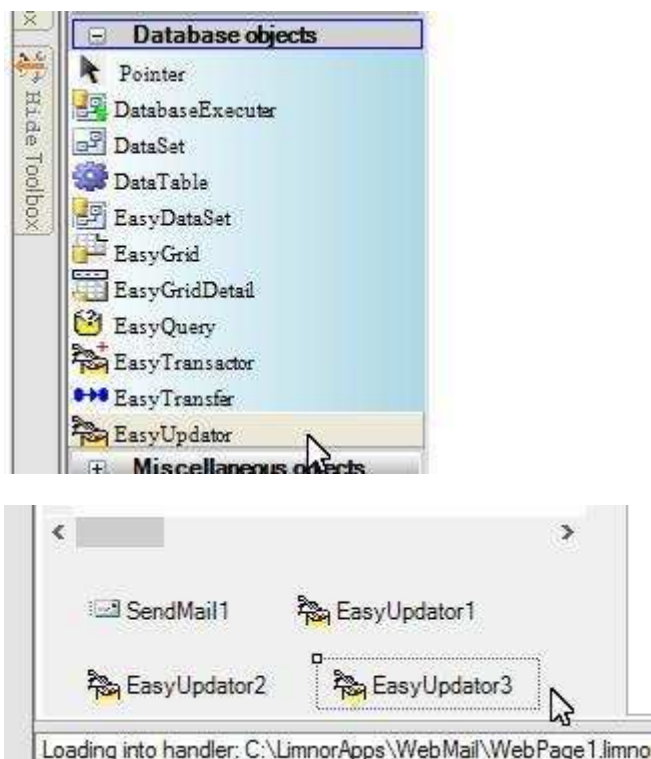
The action is created and assigned to the button:



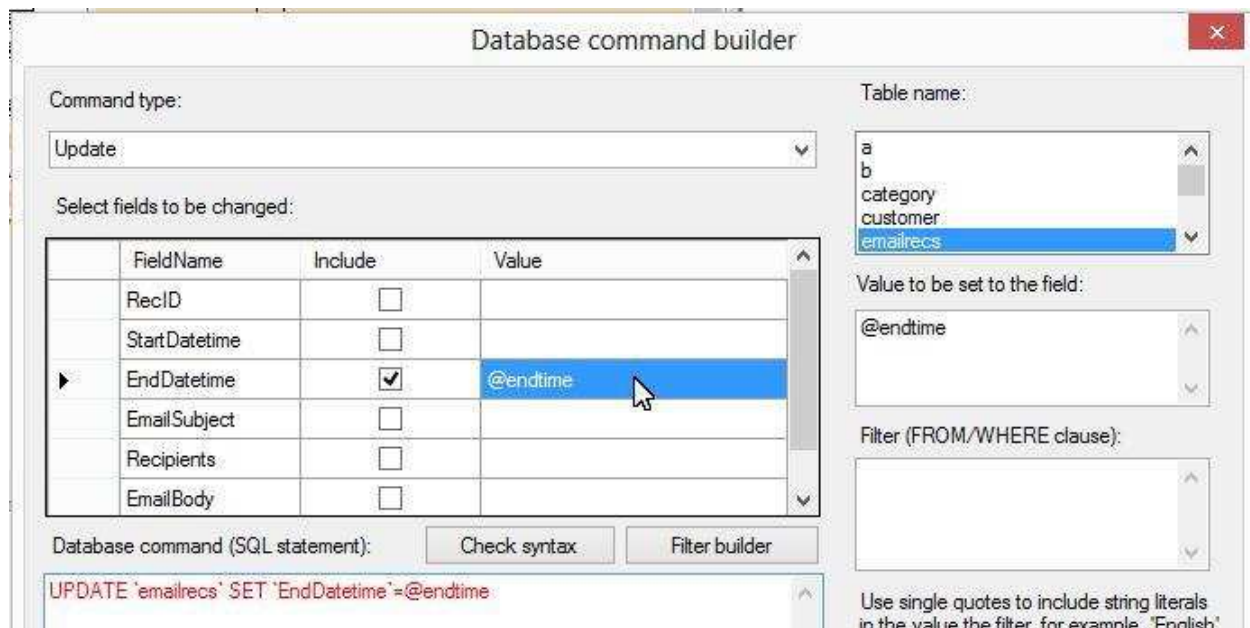
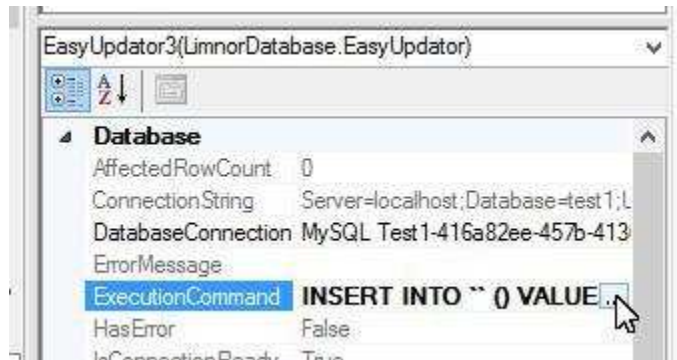
Save local ending time to database

We want to record the web page local time as the ending time to the database. This is our last task.

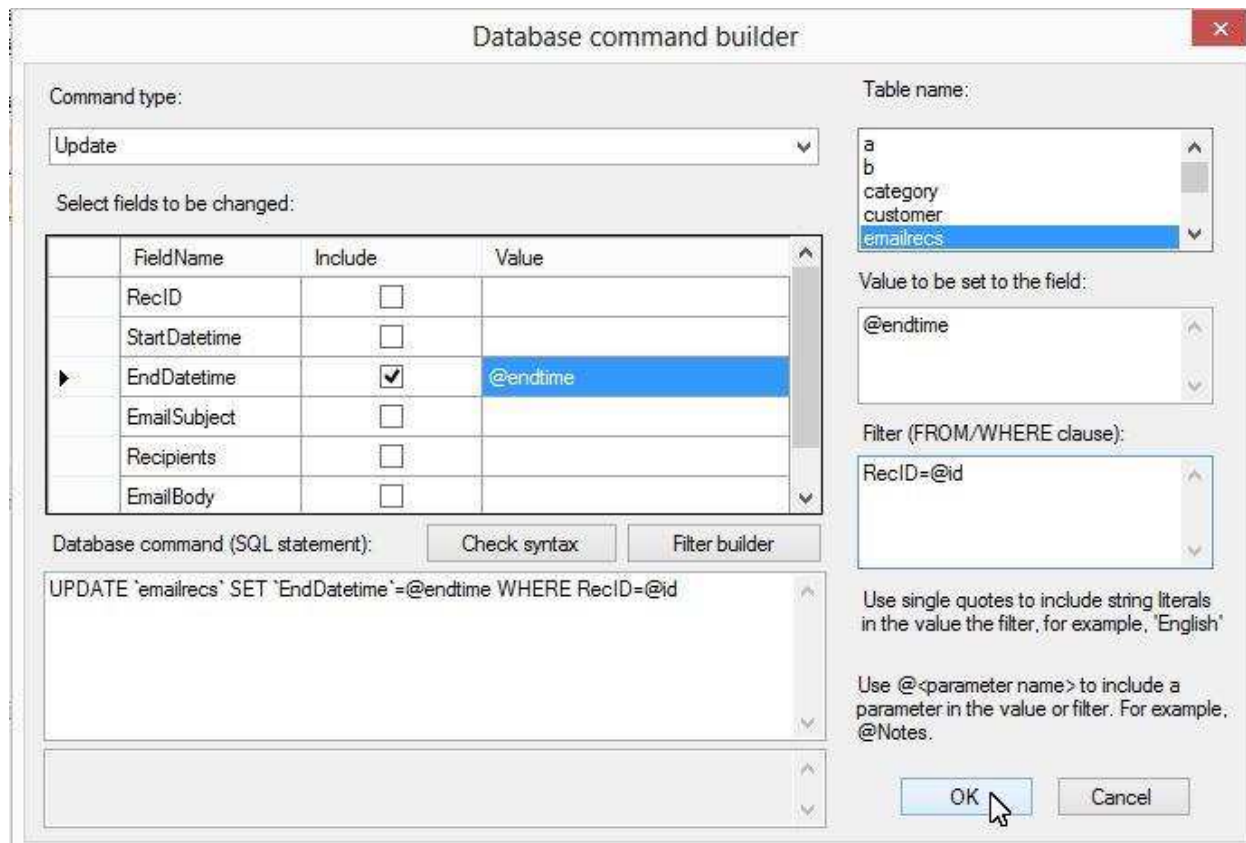
We need another EasyUpdater to do it:



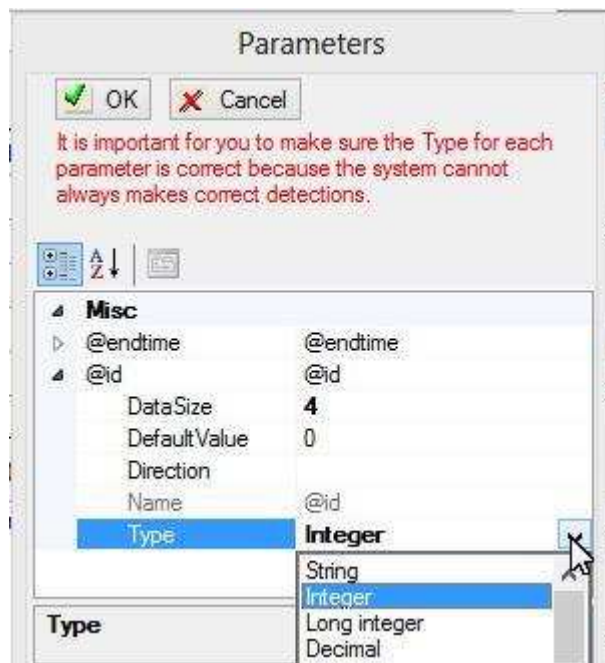
Set its ExecutionCommand for saving the ending time:



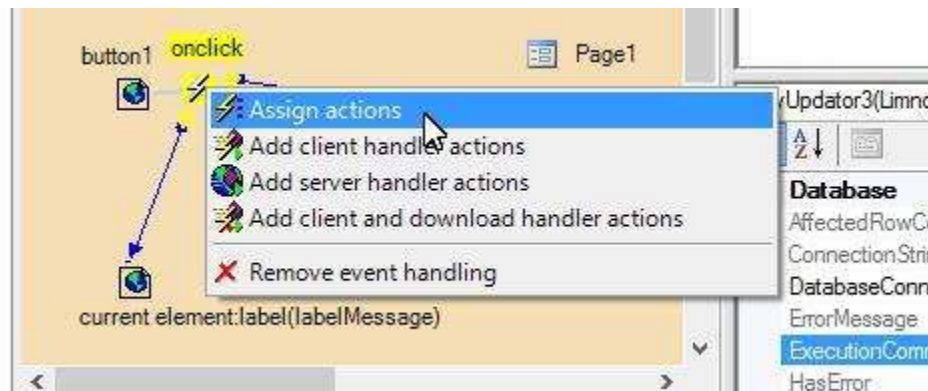
Set a filter for locating the record to be updated:



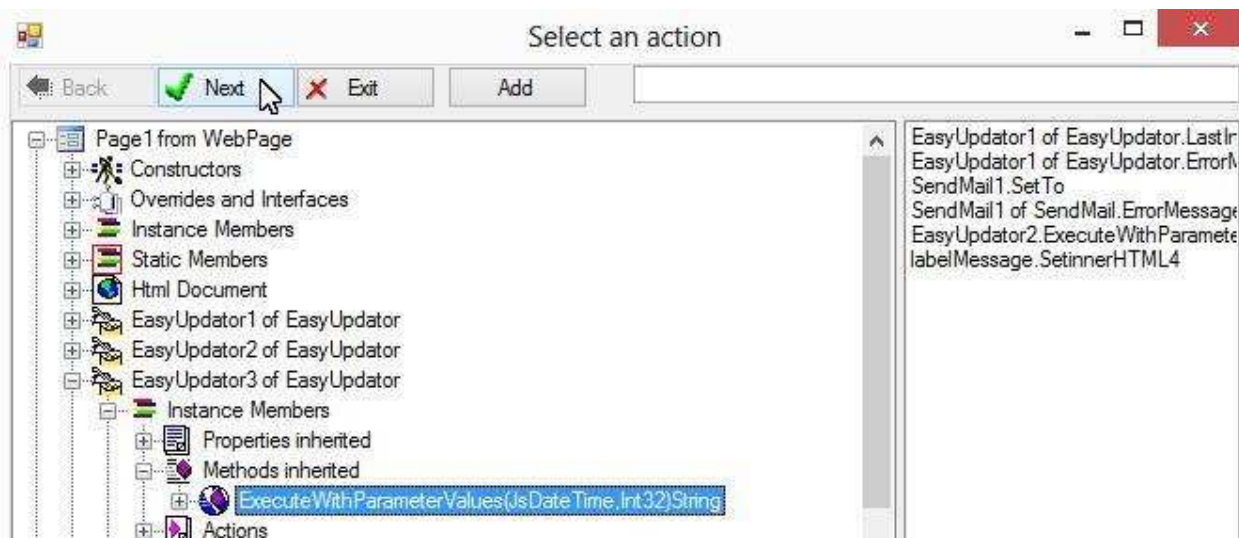
Make sure the @id is an integer:



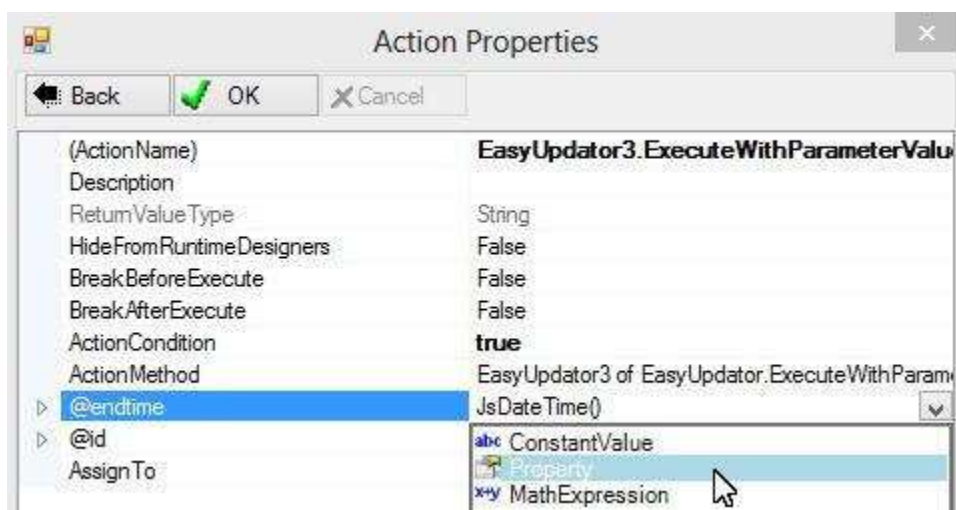
Use EasyUpdator3 for button1:

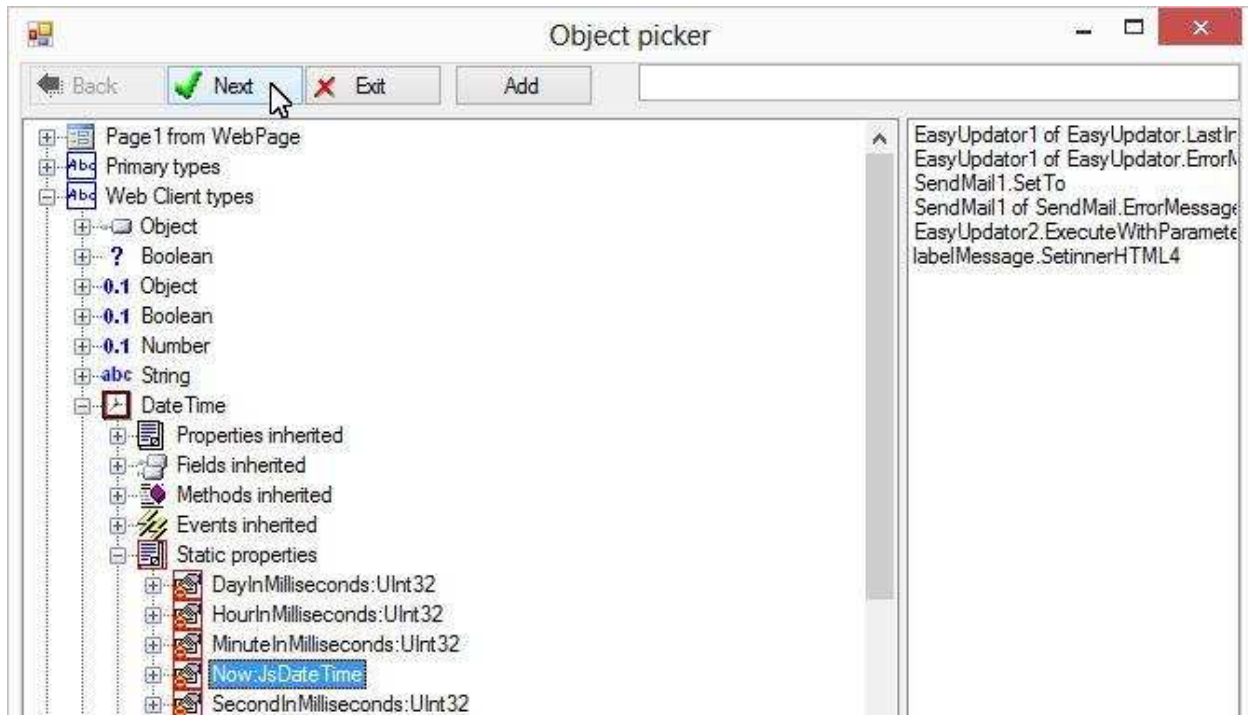


Select ExecuteWithParameterValues of EasyUpdator3:

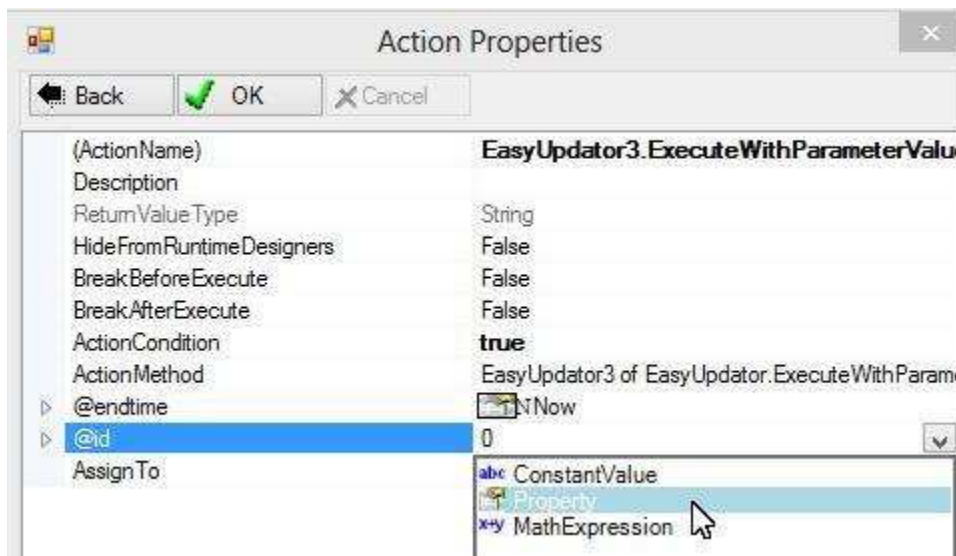


Pass Now to @endtime:



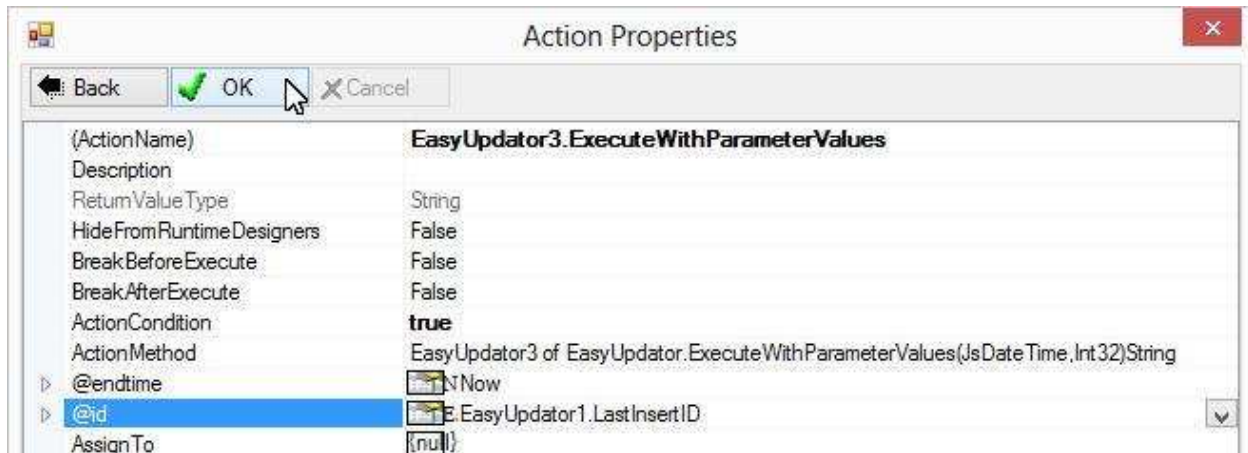


Pass LastInsertID of EasyUpdater1 to @id:

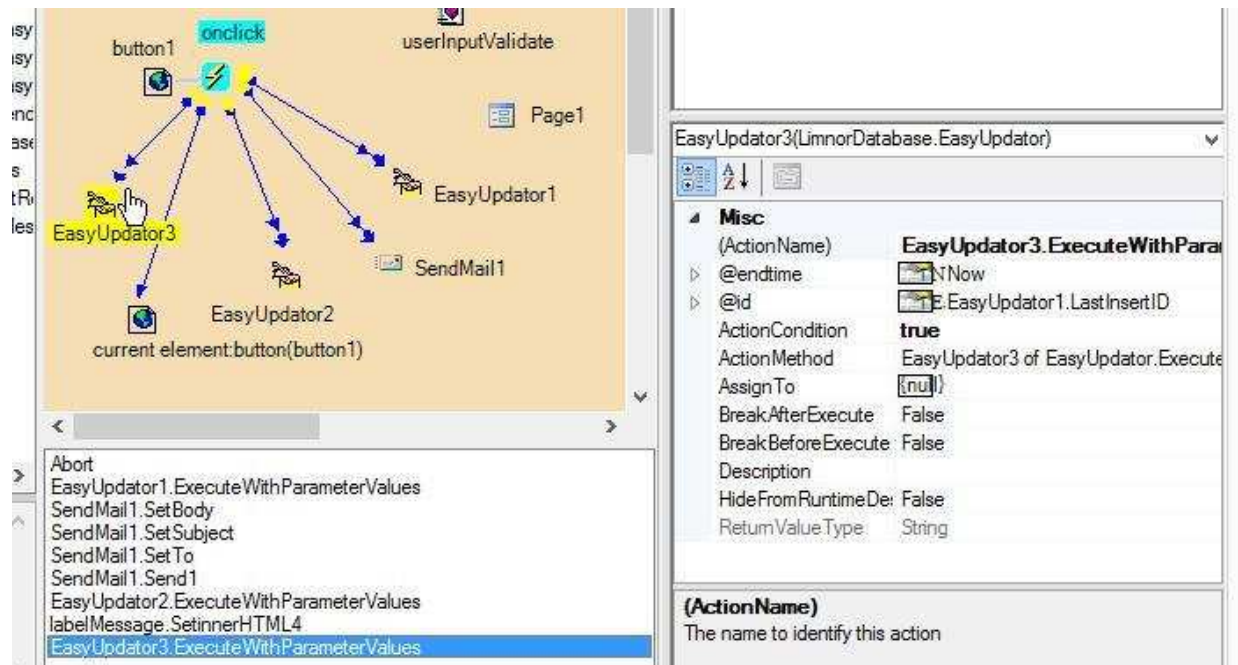




Click OK to finish creating the action:

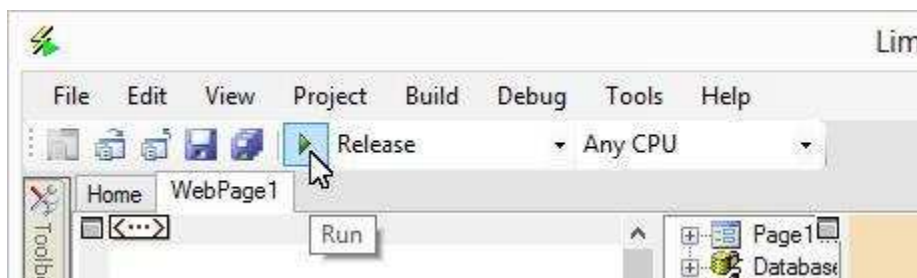


The action is created and assigned to the button:



Test

Now we are done the web page development. We may test it.



The web page appears in the default web browser:



Web Mail Sample

Recipients:

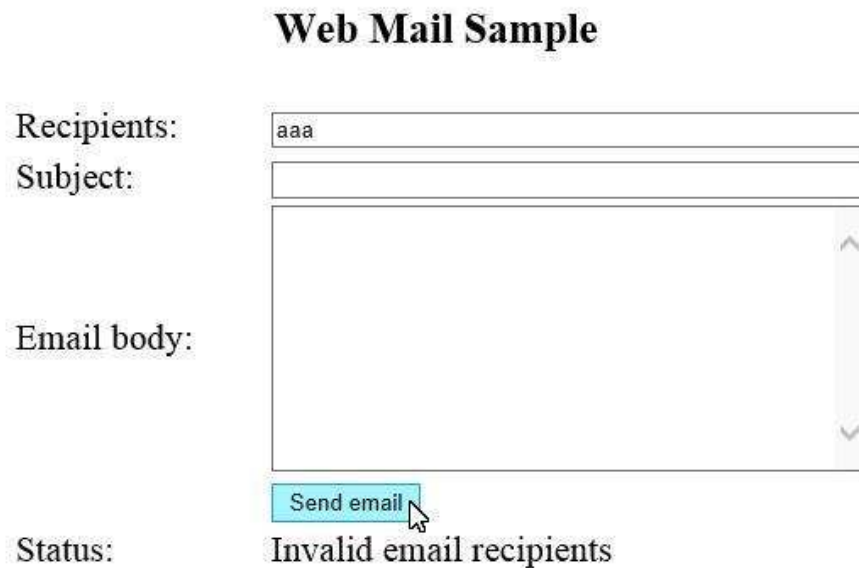
Subject:

Email body:

Status: Missing recipients

You may resize the web browser to see how the percentage settings of widths work.

Click "Send email" button, we get error message:



Web Mail Sample

Recipients:

Subject:

Email body:

Status: Invalid email recipients

Web Mail Sample

Recipients:

Subject:

Email body:

Status: Email subject is empty

Web Mail Sample

Recipients:

Subject:

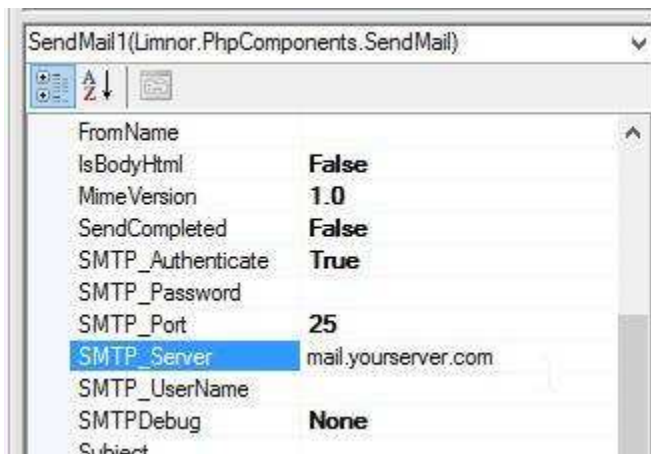
Email body:

Status: Email body is empty

Web Mail Sample

Recipients:	<input type="text" value="info@ibm.com"/>
Subject:	<input type="text" value="Test subject"/>
Email body:	<div>Test email body</div>
	<input type="button" value="Send email"/>
Status:	Unable to send the email.

The last test shows “Unable to send the email”. This is caused by the fact that we did not set SMTP related properties for SendMail1:



For this sample, that is all we want to demonstrate.

Data binding

See <http://www.limnor.com/support/WebHtmlEditorProgramming3.pdf>

File Upload

See <http://www.limnor.com/support/WebHtmlEditorProgramming4.pdf>

Feedback

Please send your feedback and suggestions to support@limnor.com. Thanks!