Limnor Studio – User's Guide

Part – II

Action Parameters

Contents

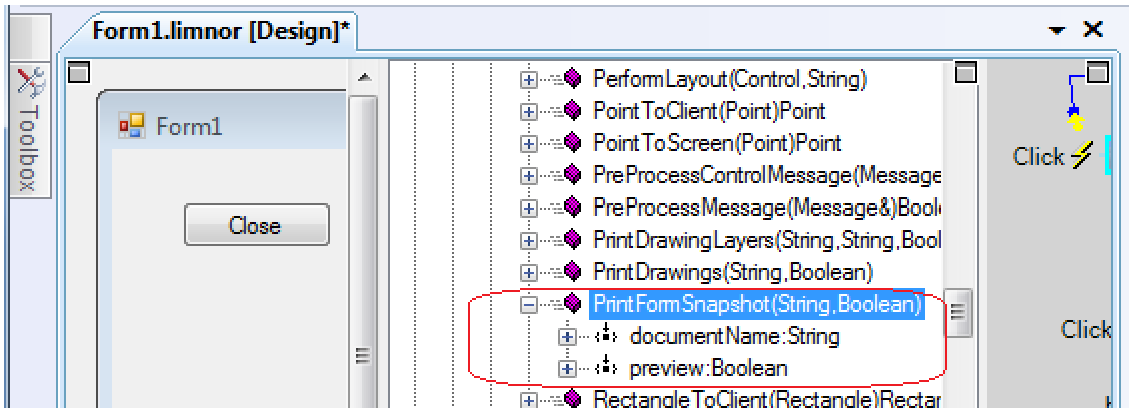
1 Use Parameters in Actions

1.1 What are action parameters

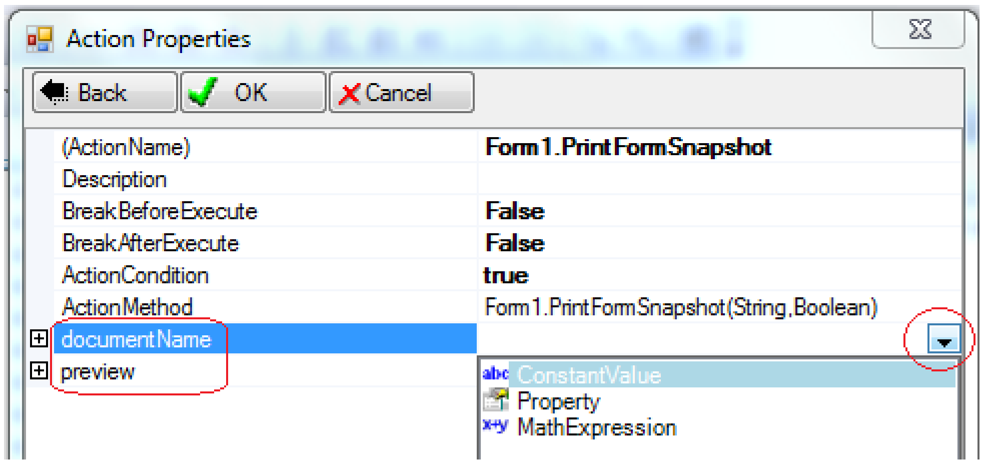
Action parameters are data needed for executing the action.

In a set-property action, it requires a “value” parameter indicating the value to be assigned to the property. In action Form1.SetText we created previously, the value parameter is a string “Hello World!”

A Method-execution action may or may not need parameters. The action Form1.Close we created previously does not require a parameter. Many other method-execution actions require parameters. The method to be executed determines the kinds of parameters required. For example, method PrintFormSnapshot has two parameters. Parameter documentName indicates the print document name; parameter preview indicates whether a preview window is displayed.


Form1.limnor [Design]*
Toolbox
Form1
Close
PerformLayout(Control,String)
PointToClient(Point)Point
PointToScreen(Point)Point
PreProcessControlMessage(Message
PreProcessMessage(Message&)Bool
PrintDrawingLayers(String,String,Bool
PrintDrawings(String,Boolean)
PrintFormSnapshot(String,Boolean)
documentName:String
preview:Boolean
RectangleToClient(Rectangle)Rectar
Click
Click

When using method PrintFormSnapshot to create actions, we may give different values for documentName and preview, for different actions:



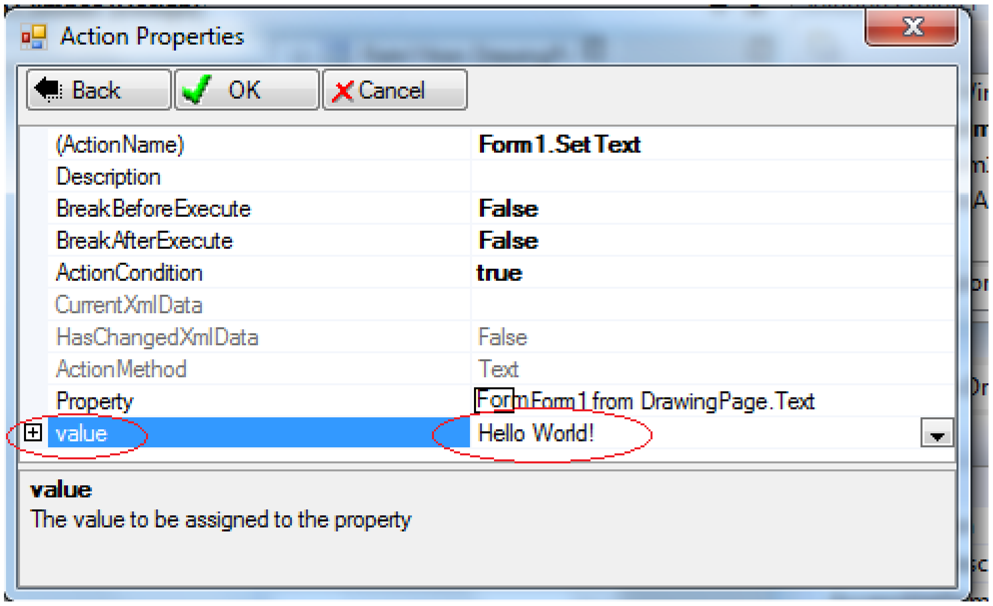
(ActionName)	Form1.PrintFormSnapshot
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	Form1.PrintFormSnapshot(String,Boolean)
documentName	
preview	

When creating an action, we must specify values for action parameters if the action requires parameters. For each action parameter, we may specify its value in 3 ways:

- ConstantValue – the value is determined at design time and compiled into the program. It will not change at runtime. For example, "Hello World!" is a constant value we used for the value of action Form1.SetText
- Property – the value is from a property of an object. It can be a reference to an object , method parameter, or a variable. The real value is determined at the time the action is executed at runtime.
- MathExpression – the value is formed by a math expression. The expression can be as simple as a constant value or a property and as complex as involving many operations and calculations.

1.2 Use Constants for Action Parameters

This is the simplest form of action parameters. Simply enter the value in the Action Properties dialogue box:

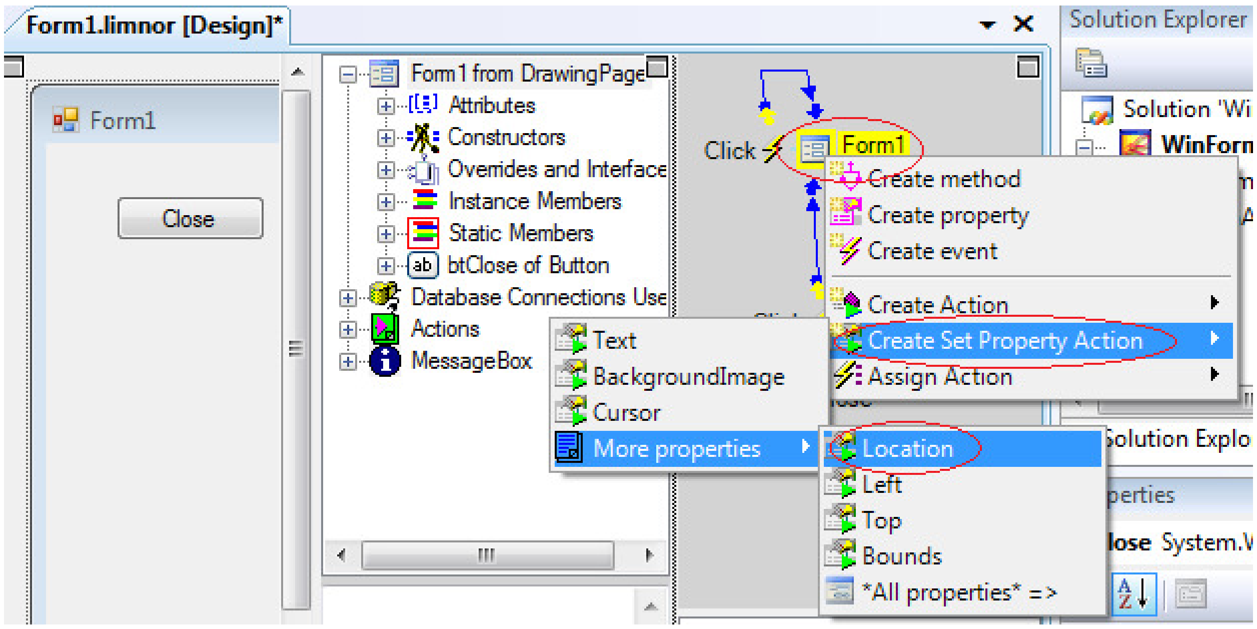


(ActionName)	Form1.SetText
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
CurrentXmlData	
HasChangedXmlData	False
ActionMethod	Text
Property	FormForm1 from DrawingPage.Text
value	Hello World!

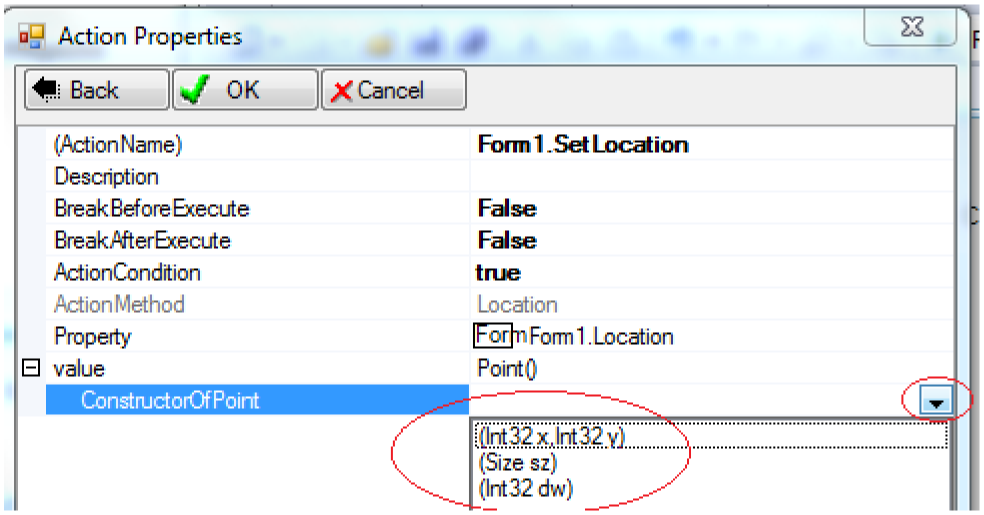
value
The value to be assigned to the property

Let's use another example to show the use of constant values. Still use the sample project we used before, WinFormApp1. Let's create an action to move the form to a location at (100,100).

Right-click icon Form1, choose "Create Set Property Action". Choose property "Location":



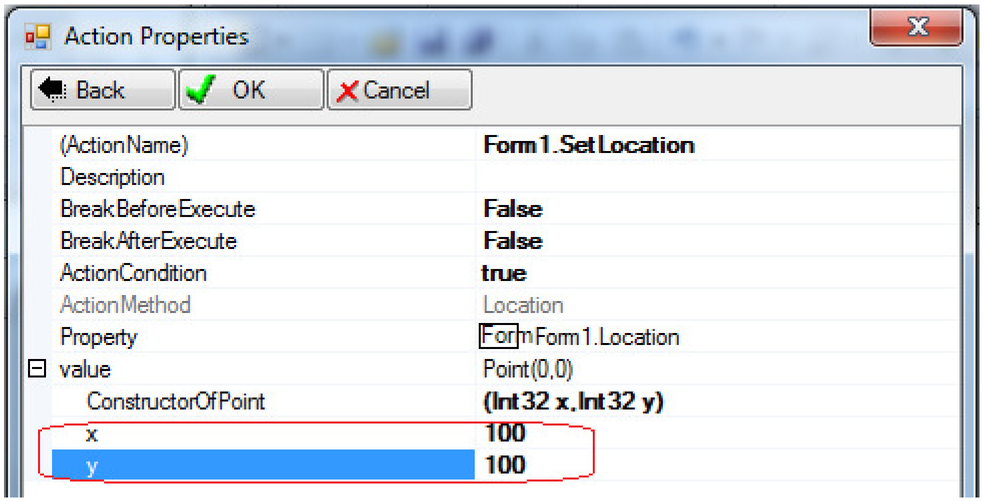
The Location property uses a Point as its value. For a Point, it allows us to use 3 different ways to enter the value. We select (Int32 x, Int32 y) to specify the value by x and y coordinates:



(ActionName)	Form1.SetLocation
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	Location
Property	Form Form1.Location
value	Point()
ConstructorOfPoint	

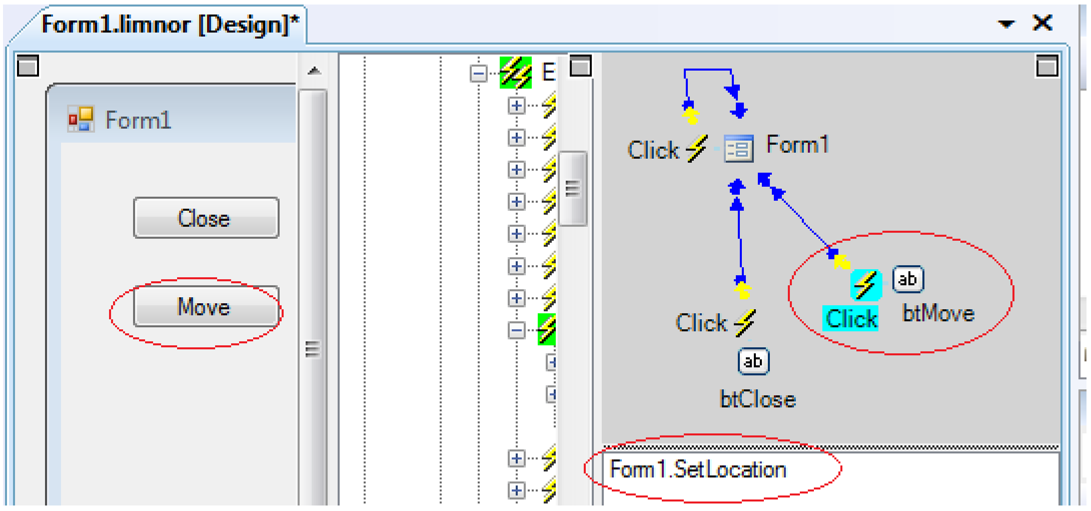
(Int32 x, Int32 y)
(Size sz)
(Int32 dw)

We may enter coordinates (100, 100):

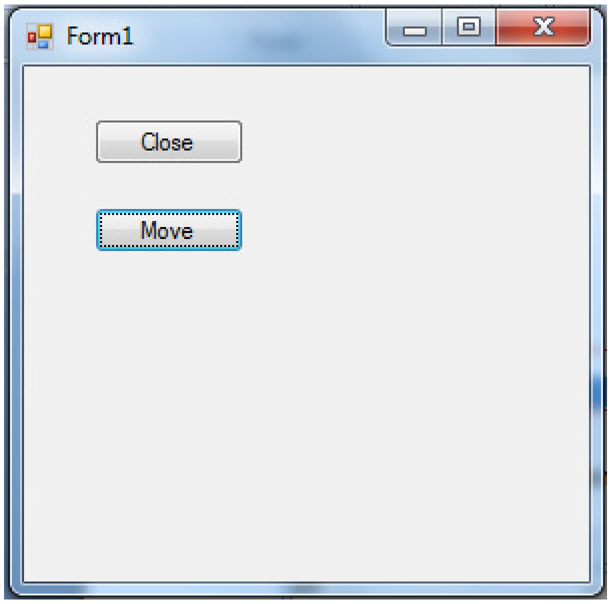


(ActionName)	Form1.SetLocation
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	Location
Property	Form1.Location
value	Point(0,0)
ConstructorOfPoint	(Int32 x,Int32 y)
x	100
y	100

Add a button to the form and assign the action to the Click event of the button:



Click to compile and run the application. The form appears:

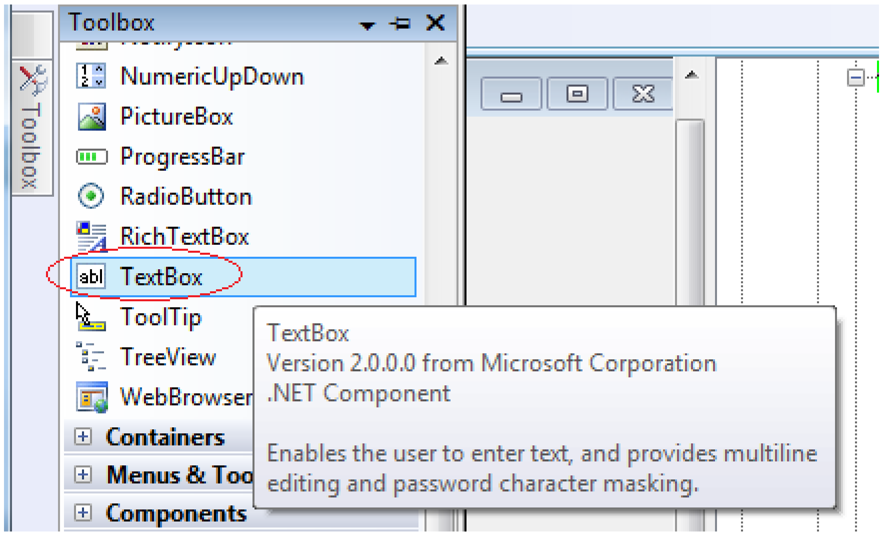


Click button "Move". The Form moves to point (100,100).

1.3 Use Properties for Action Parameters

We created an action Form1.SetText to set the caption of the form to a constant string "Hello World!" In this section we will create another SetText action to set the caption of the form to a string entered by the user.

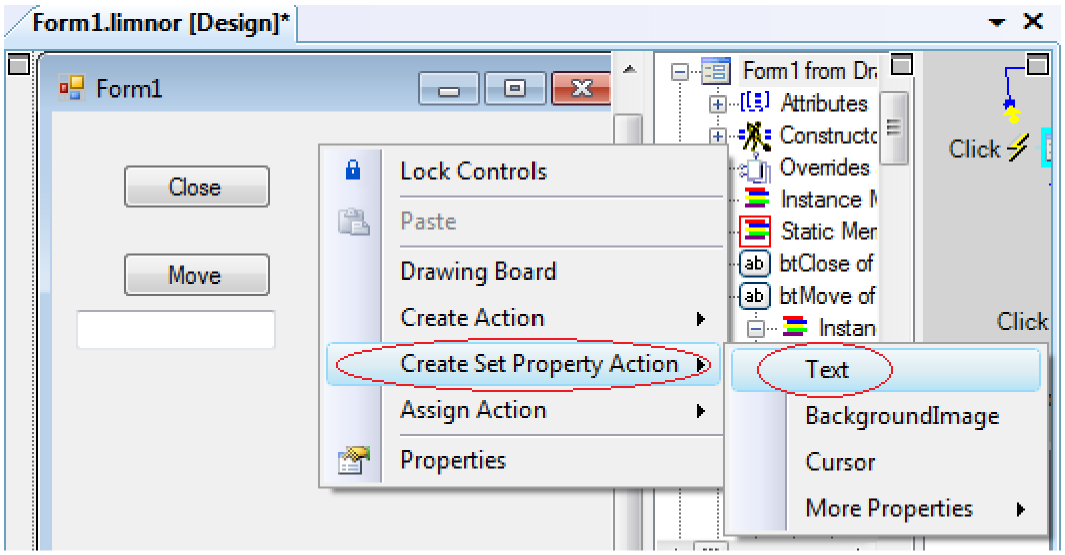
We use a TextBox to let the user enter text:



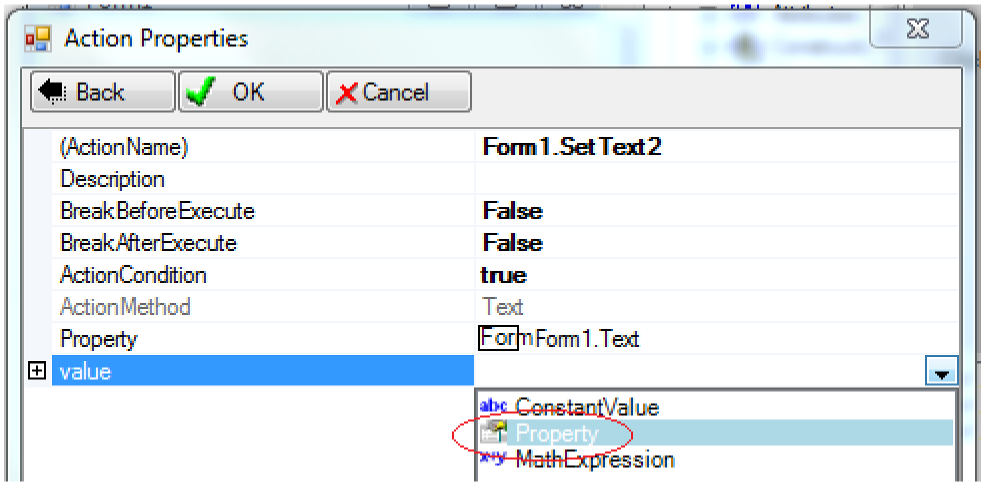
TextBox
Version 2.0.0.0 from Microsoft Corporation
.NET Component

Enables the user to enter text, and provides multiline editing and password character masking.

Right-click the form, choose "Create Set Property Action". Choose Text property:

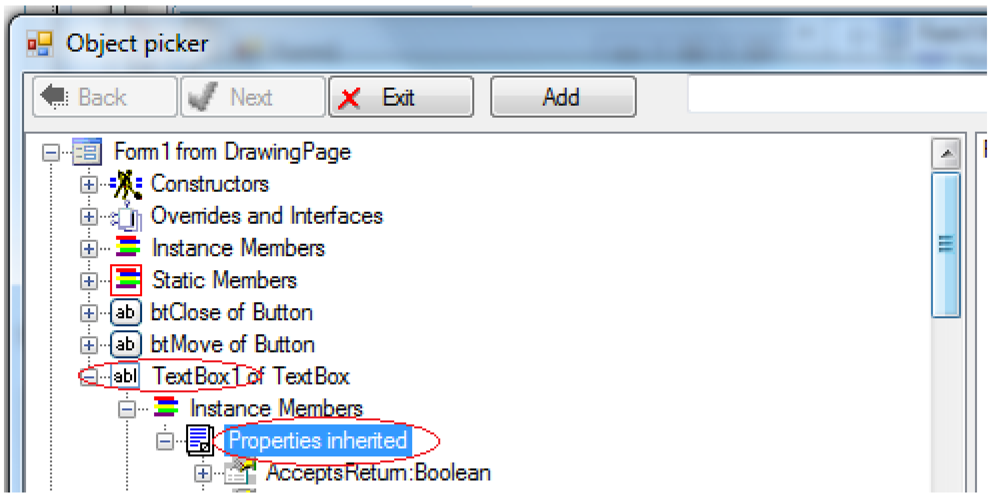


Select Property for the value:

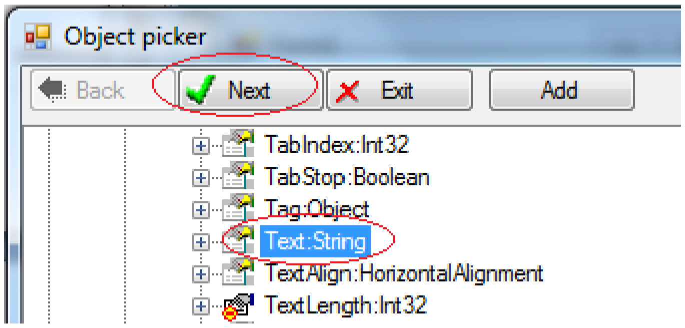


(ActionName)	Form1.SetText2
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	Text
Property	Form Form1.Text
value	

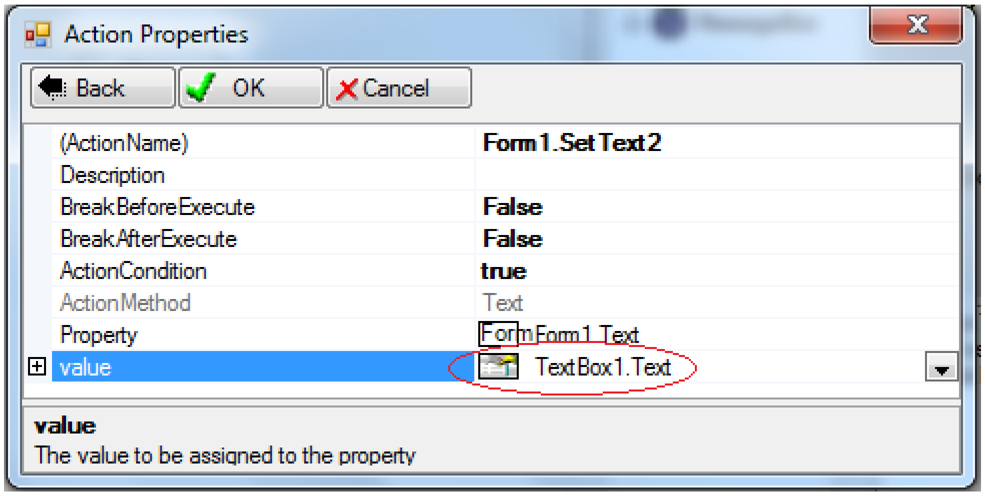
A dialogue box appears to let us choose the property. We want to use the Text property of the TextBox because it represents the text the user enters at runtime. To find the Text property of the TextBox, expand the "Properties inherited" node:



Scroll down to find the Text property and click Next:



The action Form1.SetText2 uses the Text property of TextBox1 for the value:

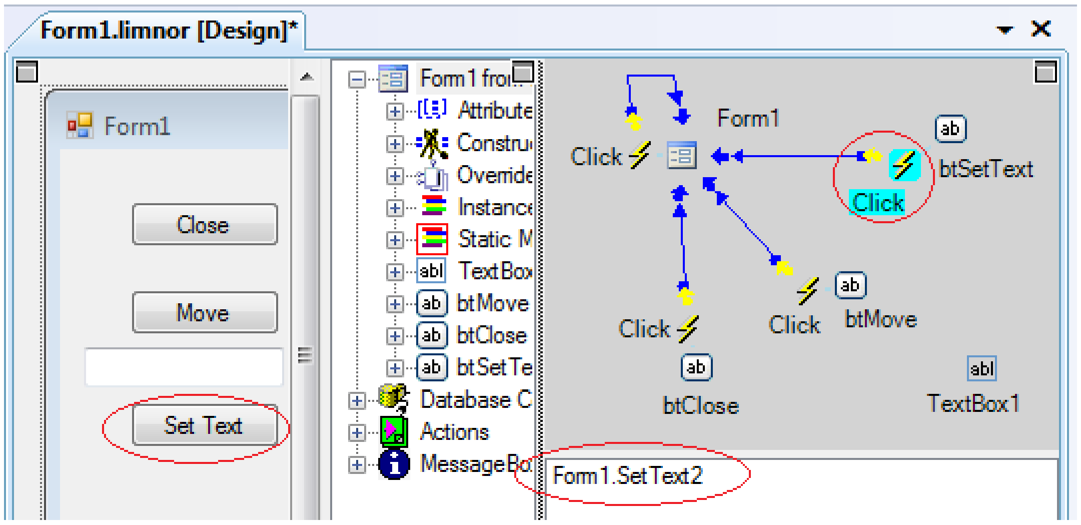


(ActionName)	Form1.SetText2
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	Text
Property	Form Form1.Text
value	TextBox1.Text

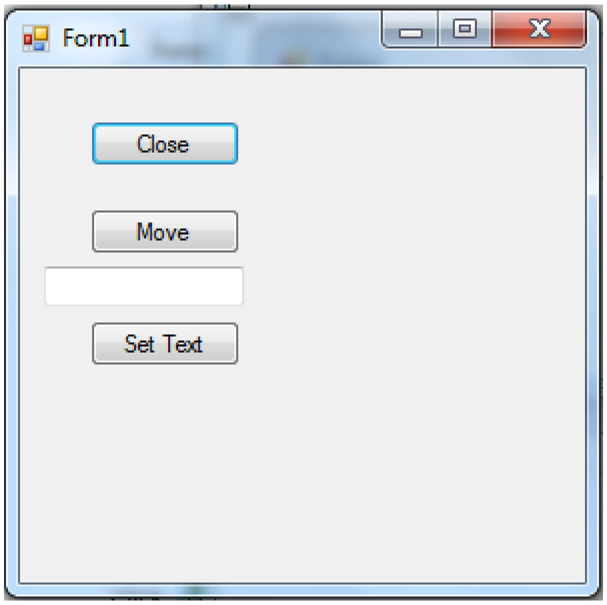
value
The value to be assigned to the property

Note that icon indicates that the value is a property.

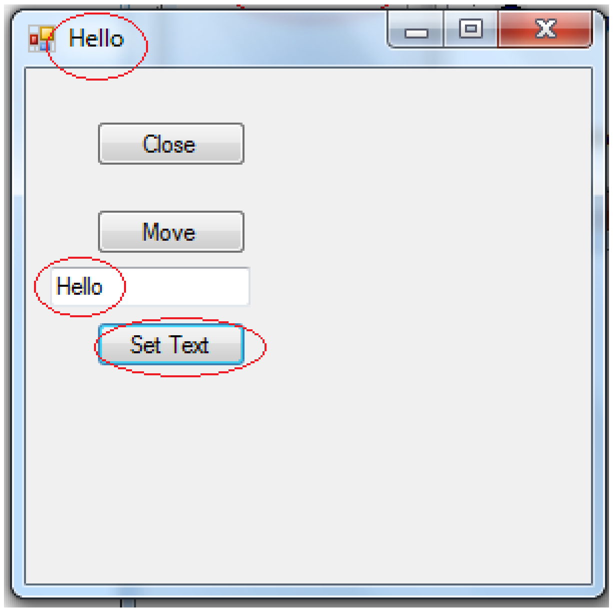
Add a button to the form and assign the new action, Form1.SetText2, to the Click event of the button:



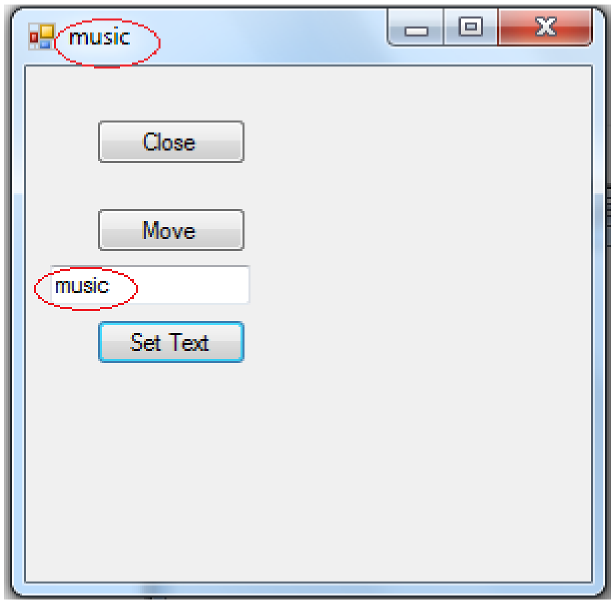
Click to compile and run the application. The form appears:



Enter some text in the TextBox, click button “Set Text”. We can see that the caption of the form becomes the same as what we entered in the text box:



Enter some text and click the button again, we can see the caption changed accordingly:



Compare to action Form1.SetText which always sets the caption to “Hello World!” This action sets the caption to different values determined only at the time of execution.

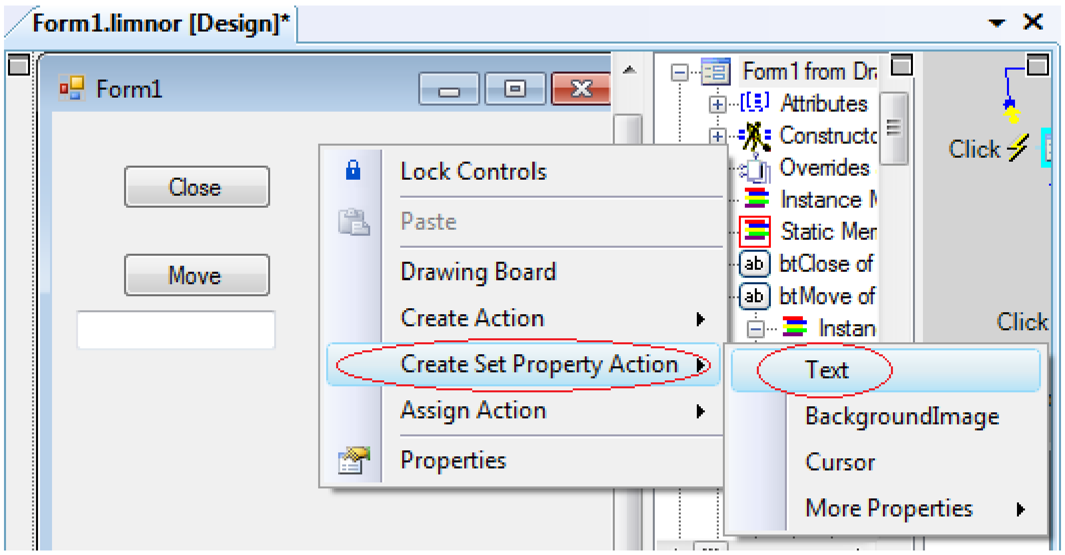
1.4 Use Expressions for Action Parameters

Constants and properties can be combined by operations to form expressions to provide values for action parameters.

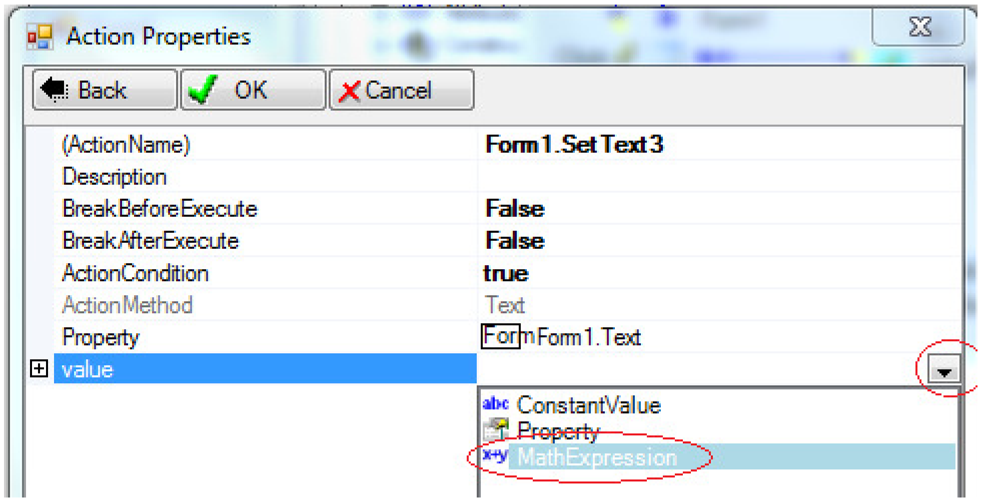
Limnor Studio provides a What You See Is What You Get math editor for constructing expressions. It supports numeric expressions, logic expressions and text expressions. The user’s guide for the math expression editor will be provided elsewhere.

Here we use a simple text expression as a sample. We concatenate a constant text "You typed " with the Text property of TextBox1.

Right-click the form, choose "Create Set Property Action". Choose Text property:



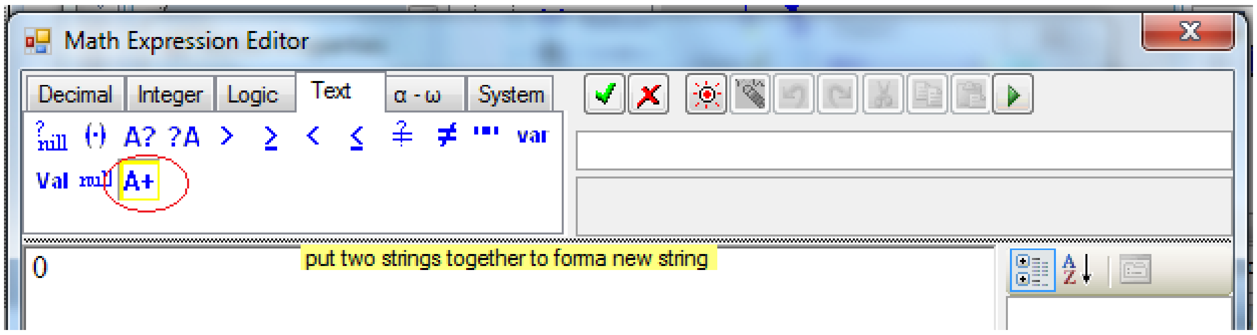
Select MathExpression for the value:



(ActionName)	Form1.SetText3
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	Text
Property	Form1.Text
value	

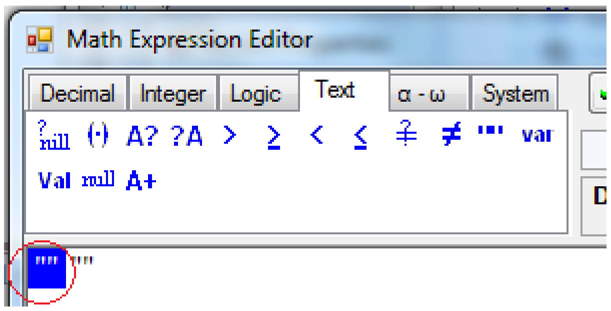
The math expression editor appears. Select Text tab to form text expression.

Click A+ to create a text concatenate expression:

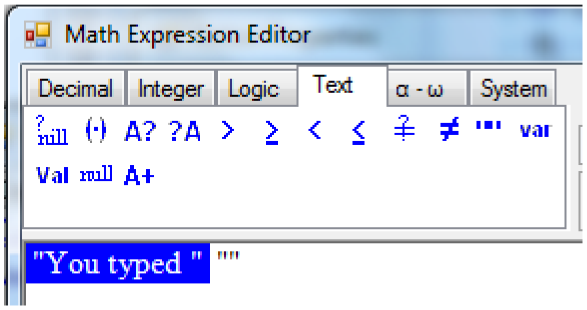


Two empty strings appear. A string constant is represented by a text enclosed by double-quotations.

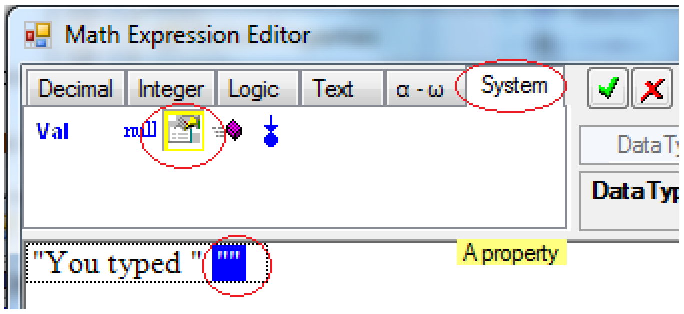
Click the first empty string to highlight it:



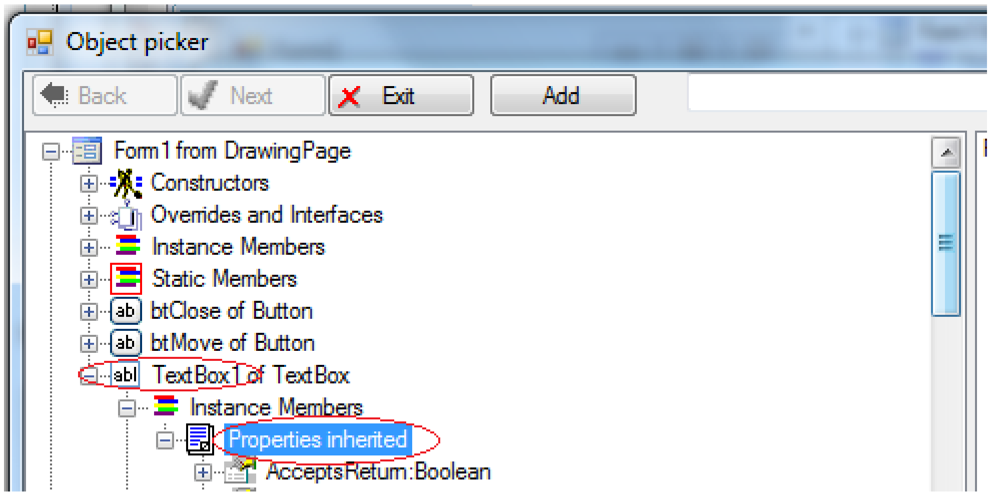
While it is highlighted, type “You typed " (do not type quotation marks):



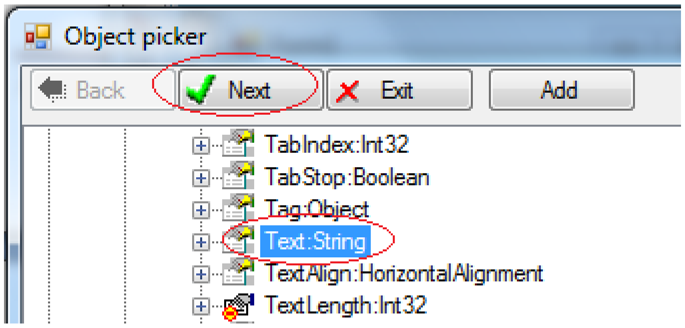
Select the second empty string. Choose System tab. Click the Property icon to use a property in the expression:



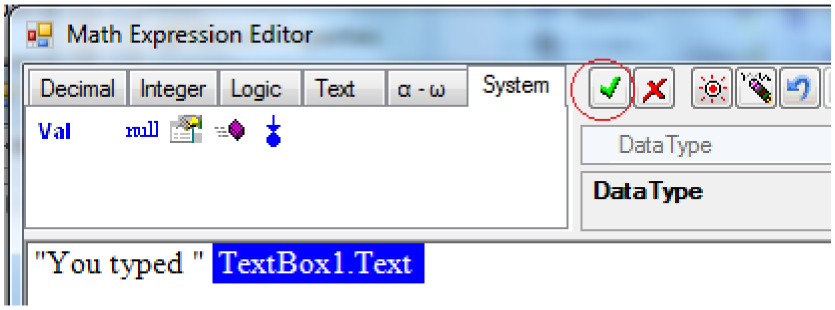
A dialogue box appears to let us choose the property. We want to use the Text property of the TextBox because it represents the text the user enters at runtime. To find the Text property of the TextBox, expand the “Properties inherited” node:



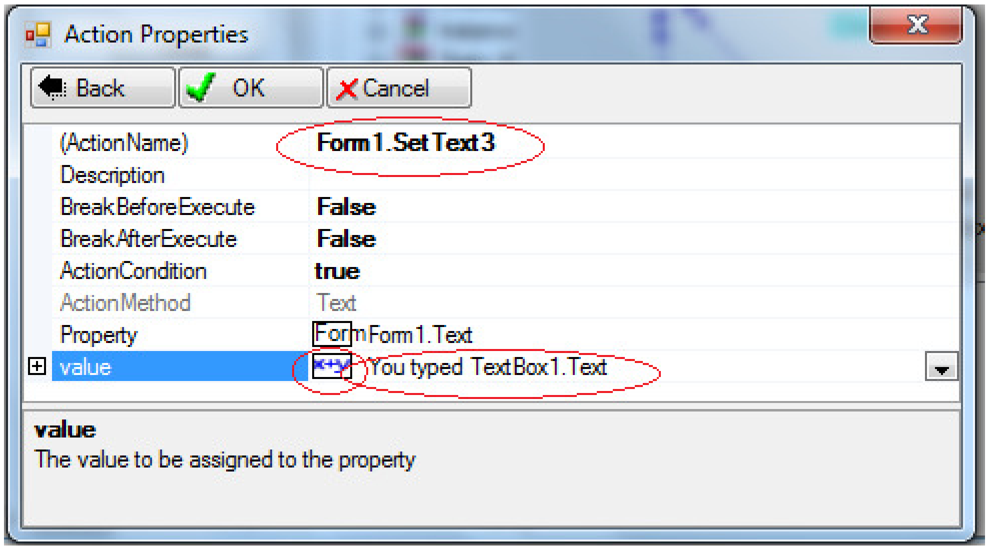
Scroll down to find the Text property and click Next:



This is our simple text expression:



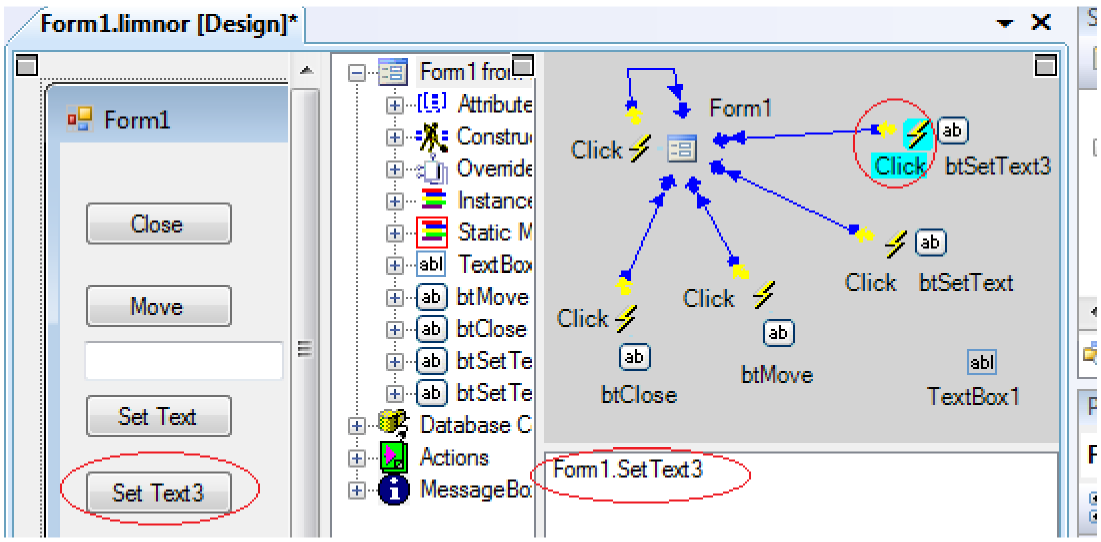
This is the new action. Note the icon which indicates that the value is an expression.



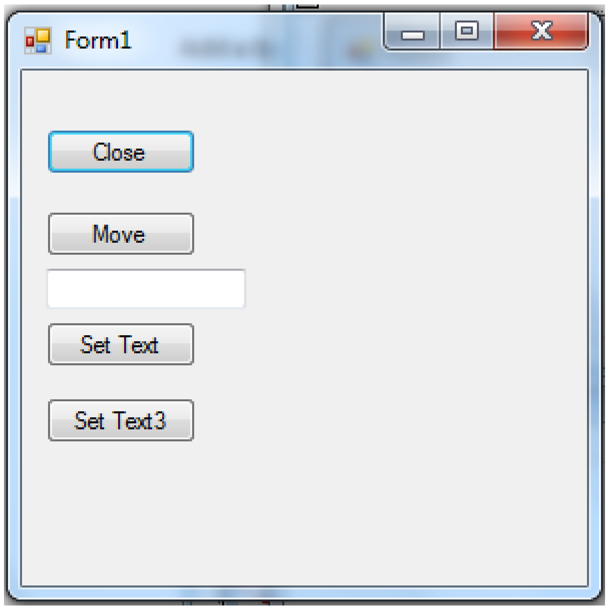
(ActionName)	Form1.SetText3
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	Text
Property	Form Form1.Text
value	You typed TextBox1.Text

value
The value to be assigned to the property

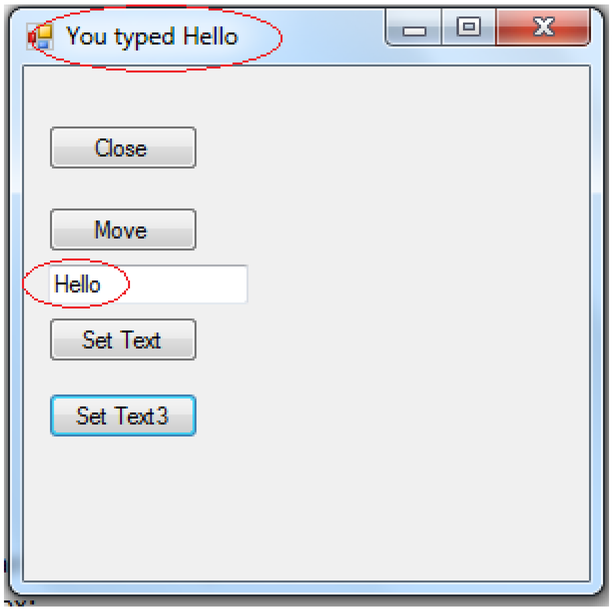
Add a button to the form and assign this new action, Form1.SetText3, to the Click event of the button.



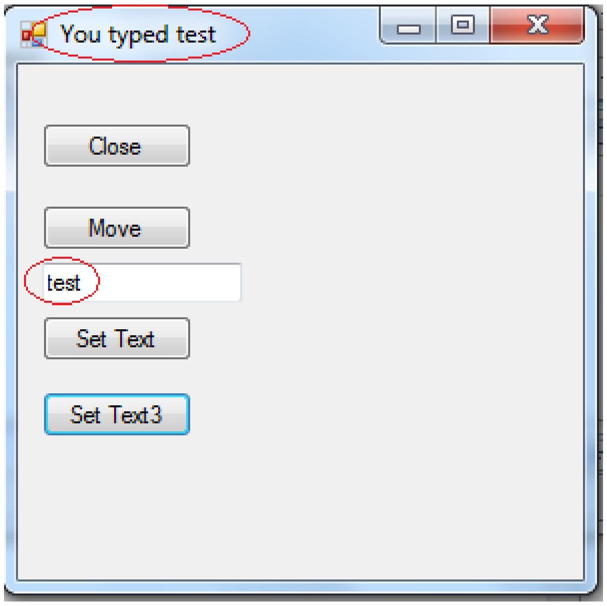
Click to compile and run the application. The form appears:



Enter some text in the TextBox, click button “Set Text3”. We can see that the caption of the form becomes a combined text using what we entered in the text box:



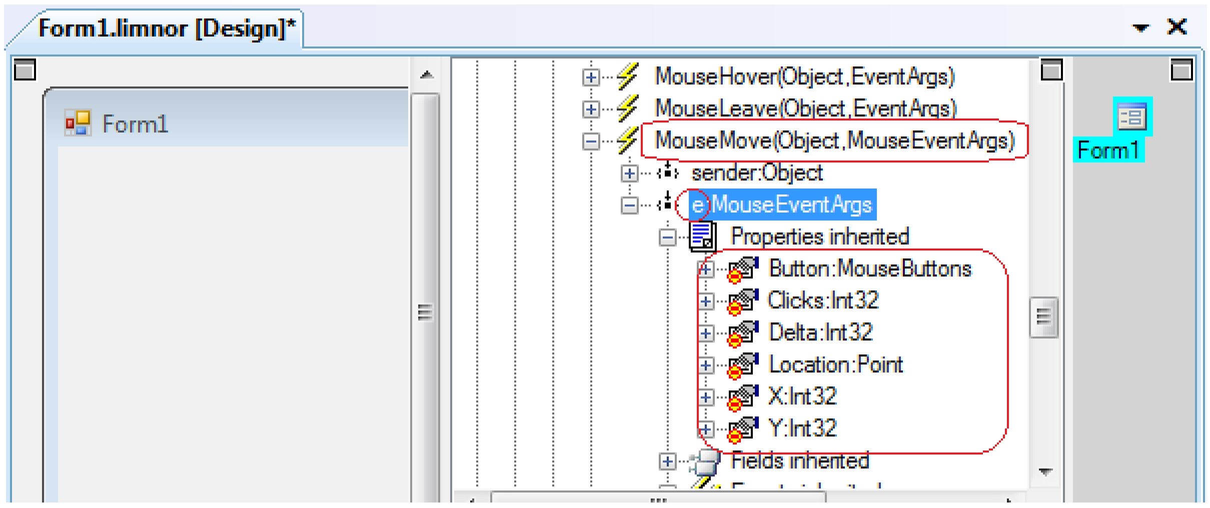
Enter some text and click button “Set Text3” again:



2 Use Event Parameters in Actions

2.1 Event Parameter

When an event occurs, some data specific to the event are available to be used by actions. Such data are called event parameters. For example, MouseMove event of a control comes with data describing the mouse properties at the time of the event. The data is a MouseEventArgs class. Its Button property indicates the mouse button pressed. Its Location property indicates the mouse pointer location. X and Y represent the coordinates of the mouse pointer location.



Suppose we want to use a Label to display mouse pointer location whenever mouse pointer location changes. We may create a “Set property” action to set the Text property of the Label to the mouse pointer location provided in the event parameter of the event MouseMove.

To use event parameters in actions, we must create an event-handler method and use actions within the event-handler method.

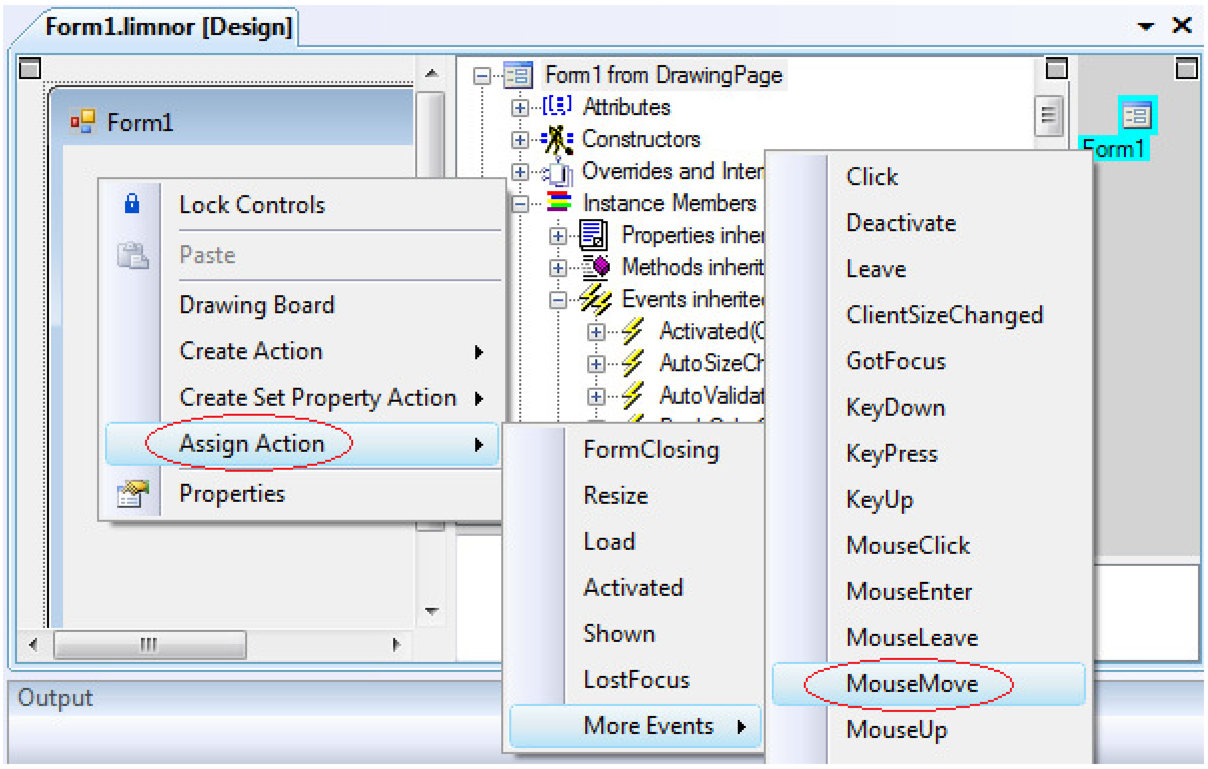
2.2 Create Event-Handler Method

An event handler method is a special method. Unlike a normal method, it cannot be used to create actions; and it does not need to create an action to execute the event-handler method. Like an action assigned to the event, the method is executed when the event occurs.

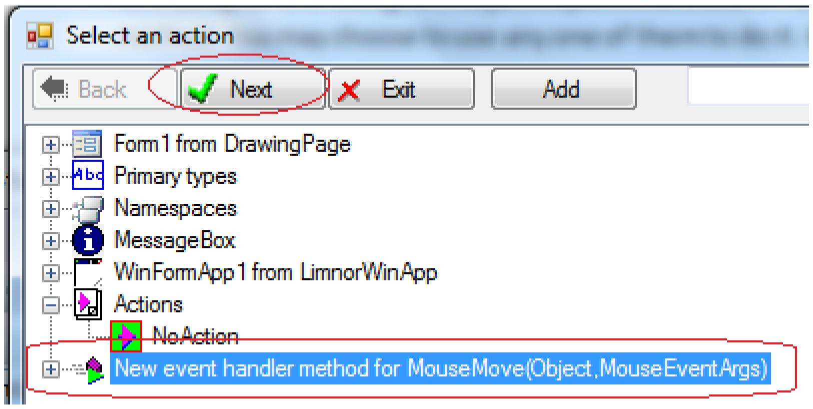
All 3 designers, UI Designer, Object Explorer, and Event Path, can be used to create event-handler method. You may choose to use any one of them to do it. We show how each designer is used to do it.

2.2.1 Create event-handler method via UI Designer

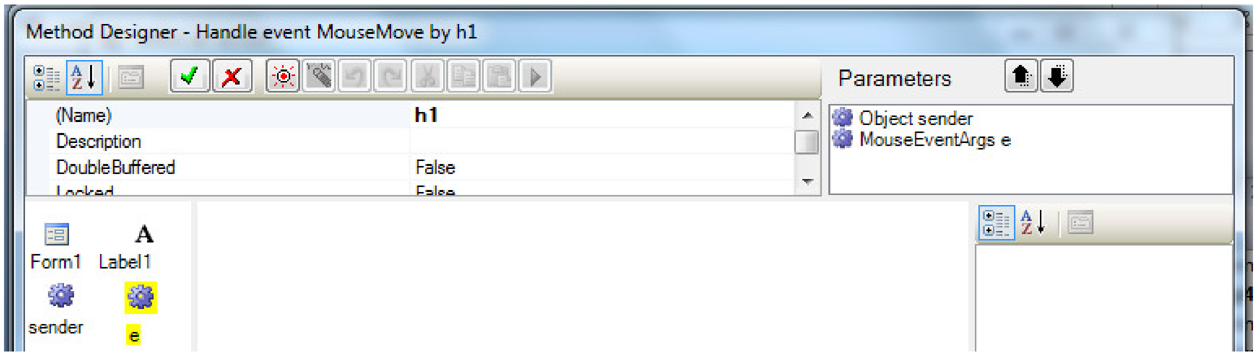
Right-click the Form, choose “Assign Action”. Choose the “MouseMove” event:



Choose “New event handler method for MouseMove(Object, MouseEventArgs)”:



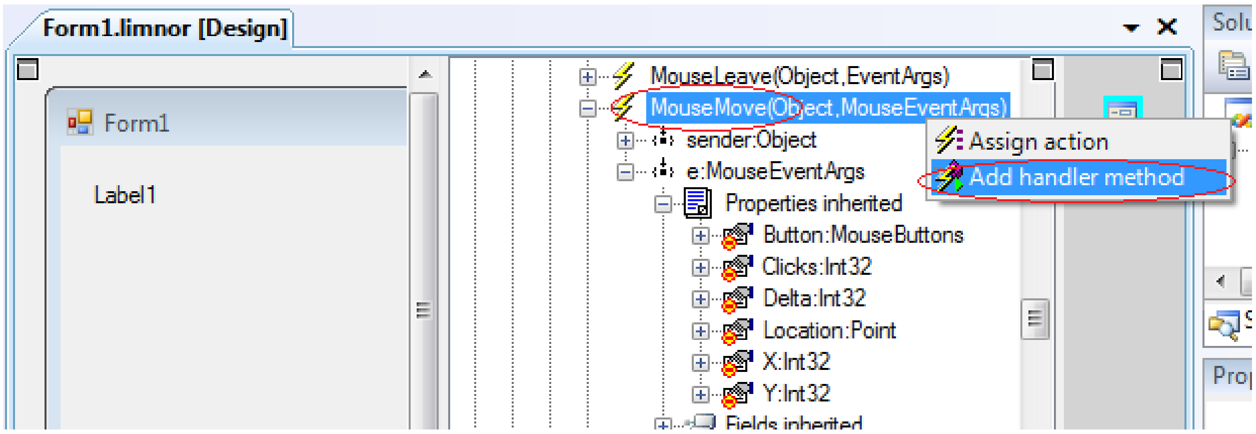
The method editor appears for developing the method:



We will show developing this method after showing getting this editor via Object Explorer and Event Path.

2.2.2 Create event-handler method via Object Explorer

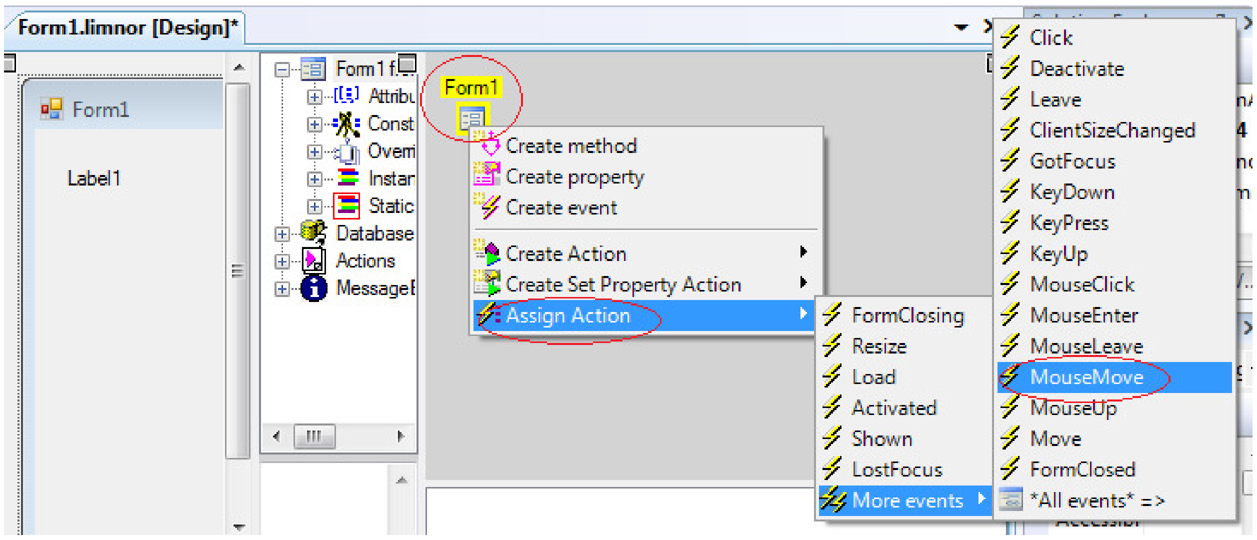
Right-click event MouseMove, Choose “Add handler method”:



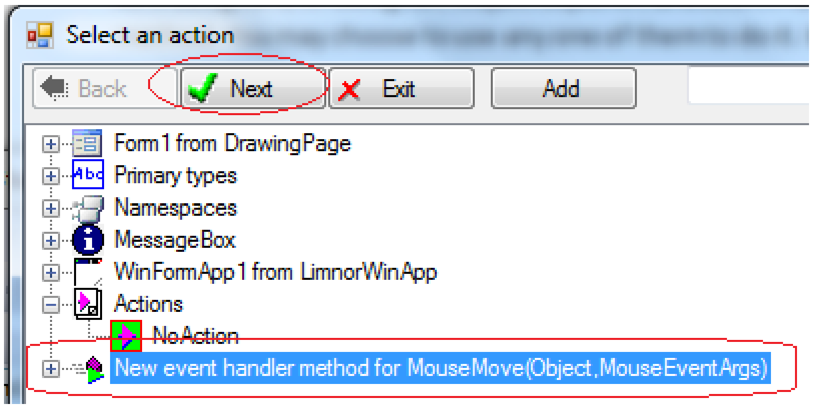
The method editor appears for developing the method.

2.2.3 Create event-handler method via Event Path

Right-click icon Form1, choose “Assign Action”. Choose “MouseMove” event:



Choose “New event handler method for MouseMove(Object, MouseEventArgs)”:

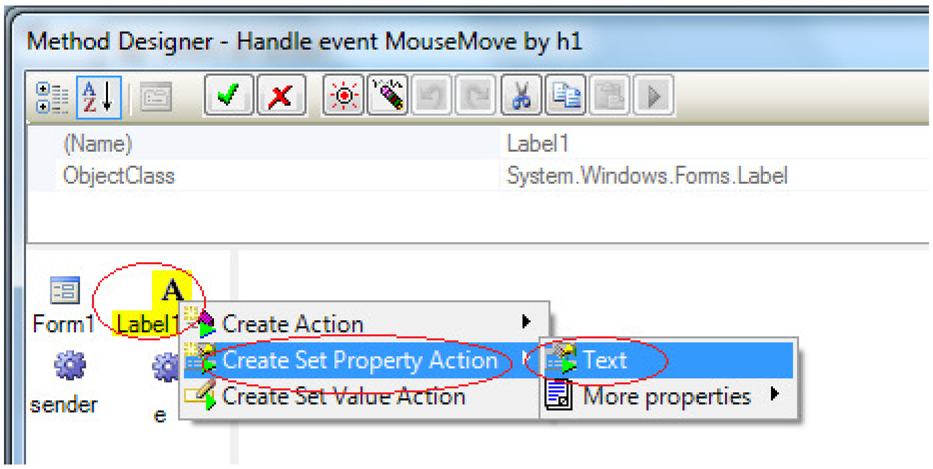


The method editor appears for developing the method.

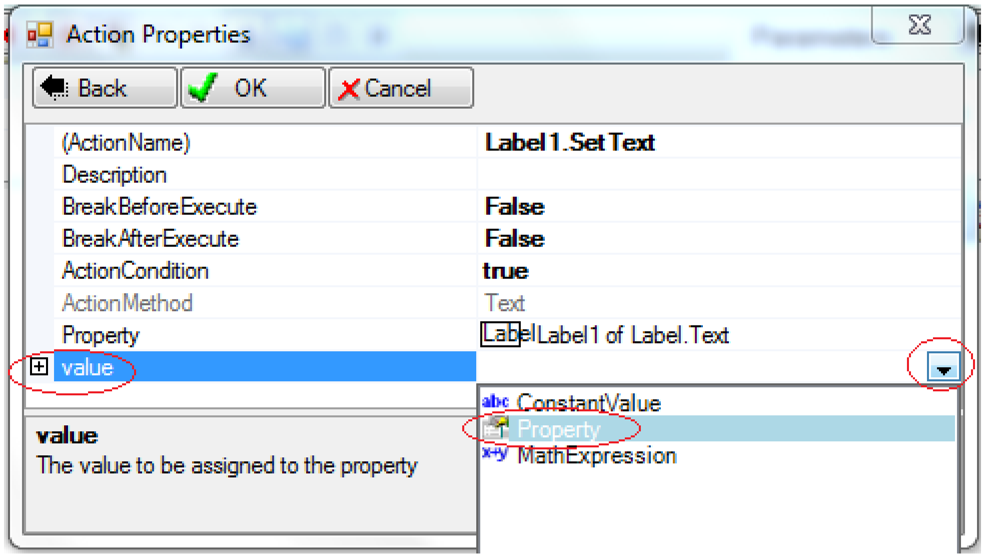
2.2.4 Developing event-handler method

We want to create a “Set Property” action to set the Text property of the Label to the mouse pointer location.

Right-click the icon for the Label, choose “Create Set Property Action”. Choose Text property:


Method Designer - Handle event MouseMove by h1
(Name)
Label1
ObjectClass
System.Windows.Forms.Label
Form1
Label1
sender
e
Create Action
Create Set Property Action
Create Set Value Action
Text
More properties

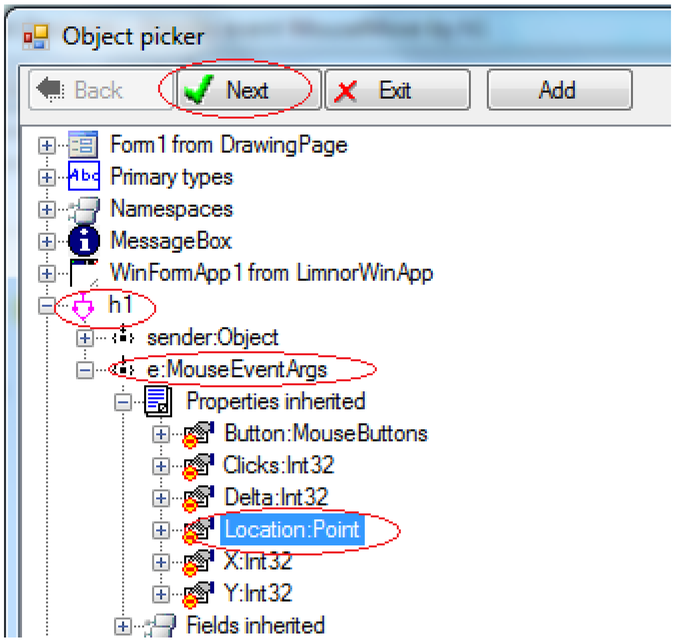
For the action parameter “value”, choose “Property” because we want to use the mouse pointer location which is available as a property of the event parameter:



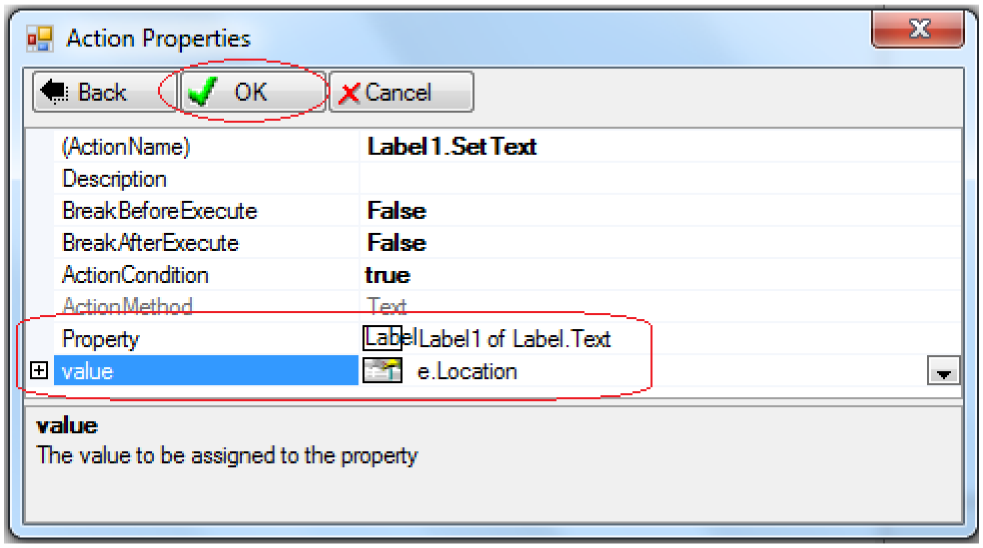
(ActionName)	Label1.SetText
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	Text
Property	Label1 of Label.Text
value	

value
The value to be assigned to the property

The event parameters are listed under the event-handler method h1 . Parameter e contains the properties for the mouse pointer. Select Location property. Click Next:



The action set the Text property of the Label to the mouse location. Click OK to finish creating this action.

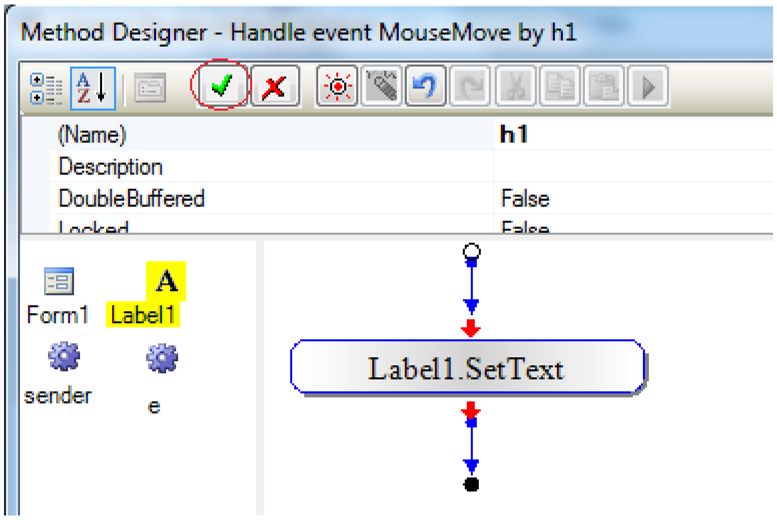


(ActionName)	Label1.SetText
Description	
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	Text
Property	Label1 of Label.Text
value	e.Location

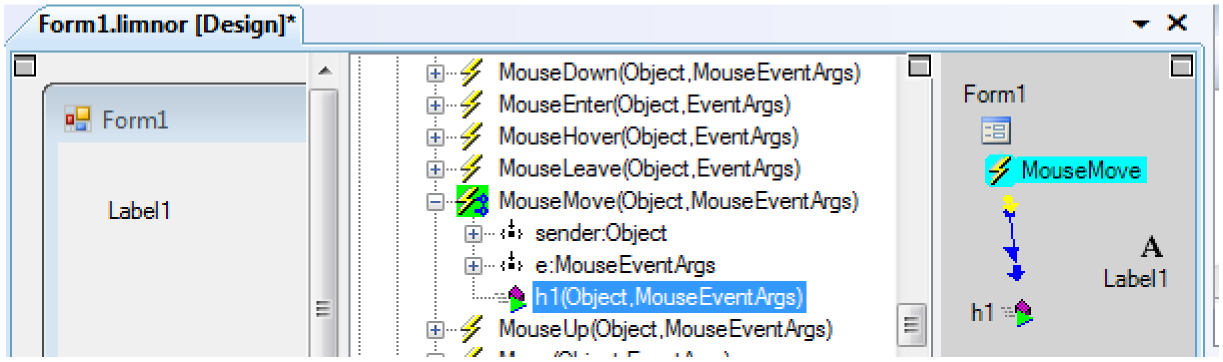
value
The value to be assigned to the property

The action appears in the method editor. For this event-handler method, we just need this one action.

Click ✓ to finish creating this event-handler method.

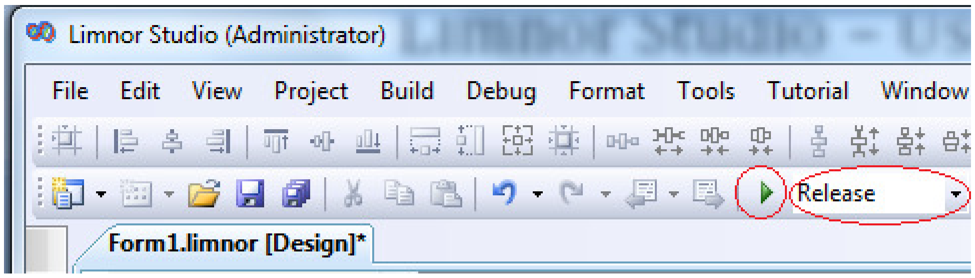


The event-handler method appears under the event node in the Object Explorer. In the Event Path, the event-handler method appears an icon. A line links the event icon to the event-handler icon.

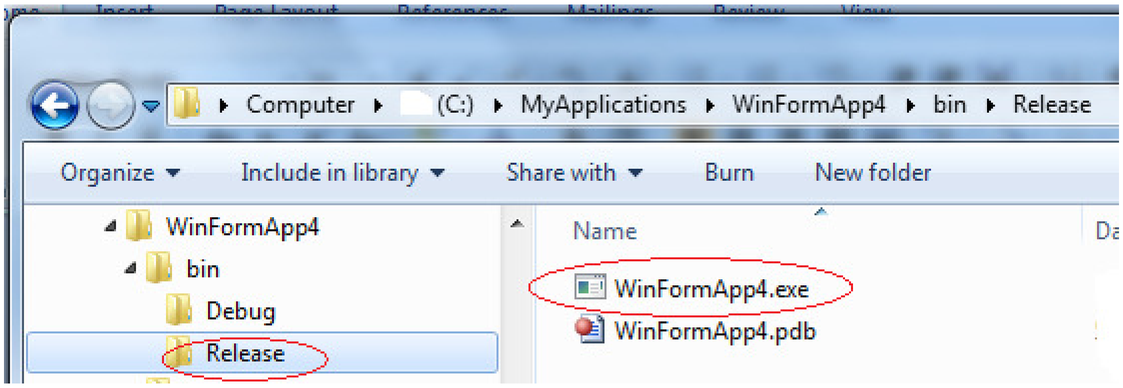

Form1.limnor [Design]*
Form1
Label1
MouseDown(Object,MouseEventArgs)
MouseEnter(Object,EventArgs)
MouseHover(Object,EventArgs)
MouseLeave(Object,EventArgs)
MouseMove(Object,MouseEventArgs)
sender:Object
e:MouseEventArgs
h1(Object,MouseEventArgs)
MouseUp(Object,MouseEventArgs)
Form1
MouseMove
Label1
h1

2.3 Compile and test the project

Click ▶ to compile and test the project.

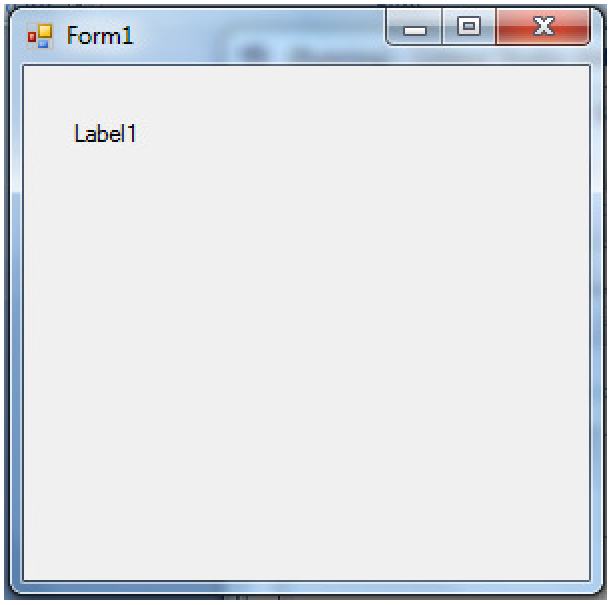

Limnor Studio (Administrator)
File Edit View Project Build Debug Format Tools Tutorial Window
Release
Form1.limnor [Design]*

If the compilation succeeds then the software is created under bin\Release folder under the project folder:

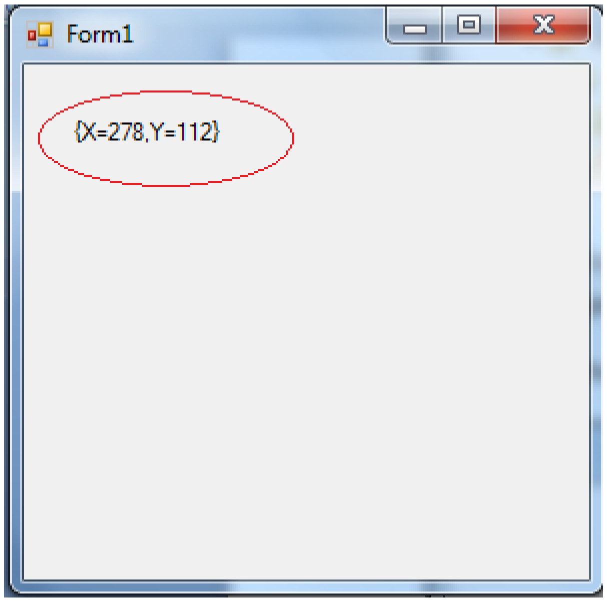


If Debug build is used then the software will be created in the bin\Debug folder under the project folder.

On compiled the software, Limnor Studio starts the software for debugging. The first form appears:



Move mouse pointer inside the form, we will see that the label text keeps changing when the mouse pointer moves:



{X=278,Y=112}