

# Handle Events for Many Objects

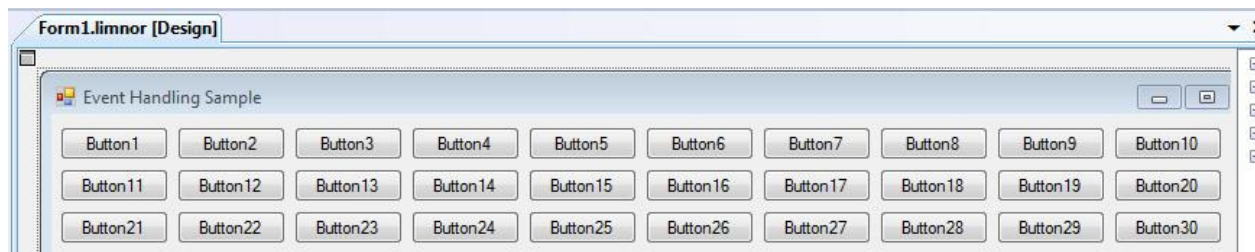
## Contents

Introduction .....	1
The Problem .....	1
The Solution .....	1
Handler Method Sample.....	2
Handle all types.....	6
Test .....	7

## Introduction

### The Problem

Suppose we have a Form with 30 buttons.



To handle one event of the buttons, at least 30 actions have to be created and assign the actions to each button respectively. These operations are tedious and hard to manage. Mistakes can happen.

### The Solution

Limnor Visual Programming invented a concept of “programming on types”. It allows you to do programming on one object and apply, very easily, to all objects of the same type. Using this solution to solve the above problem, only one action is needed and assigns the action once, instead of creating 30 actions and assigning actions 30 times.

Below we demonstrate this easy solution using the above sample.

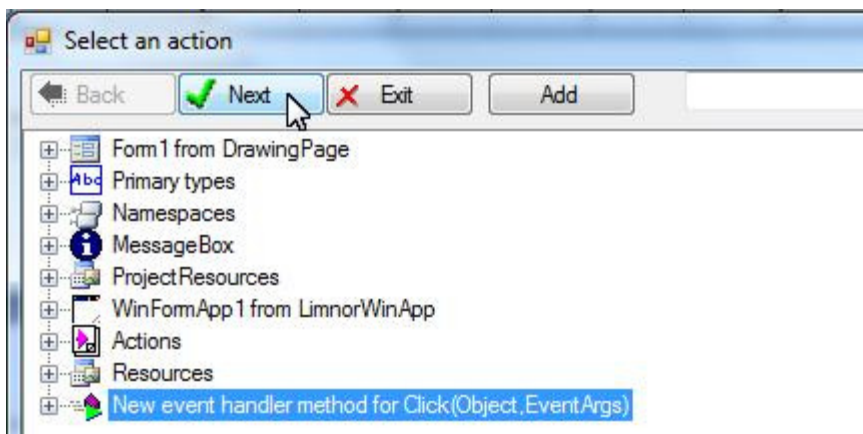
## Handler Method Sample

If you are not new to Limnor Studio then you probably already created event handler methods. An event handler method can be created through the UI Designer, the Object Explorer, and the Event Path Explorer. Here we use the UI Designer to do it.

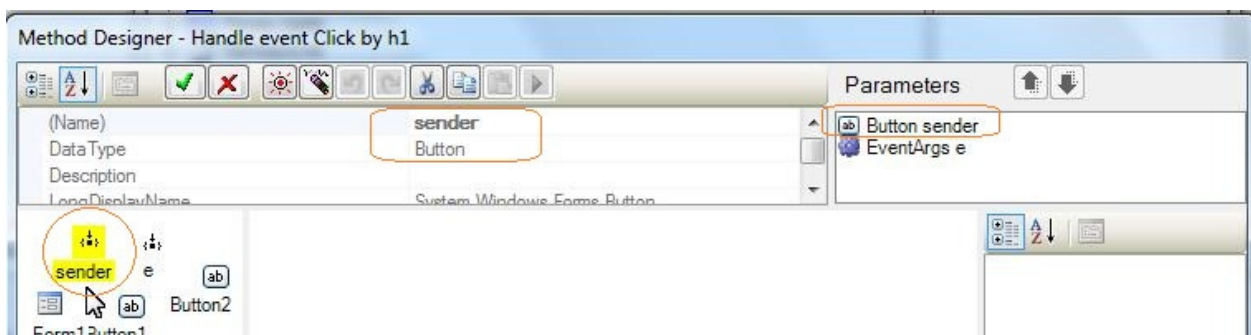
Right-click any one button; choose “Assign Action”; choose “Click” event:



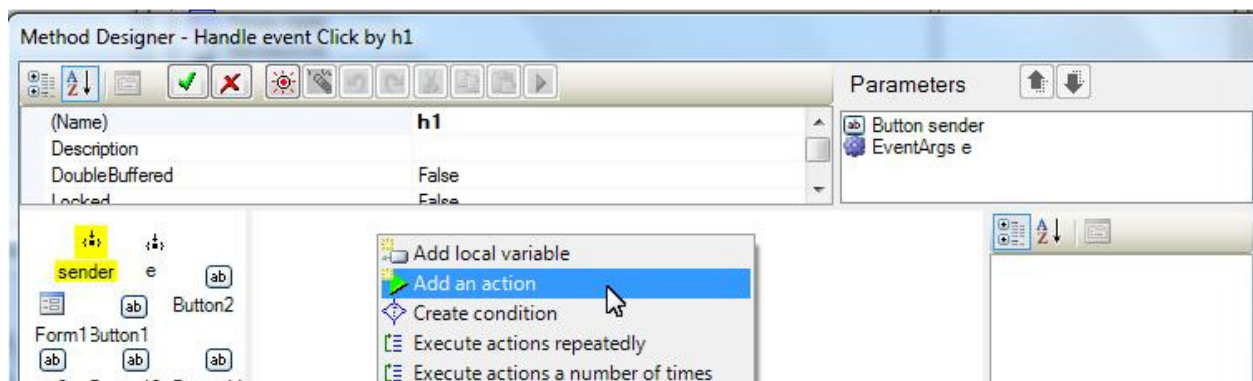
Choose “New event handler method for Click”



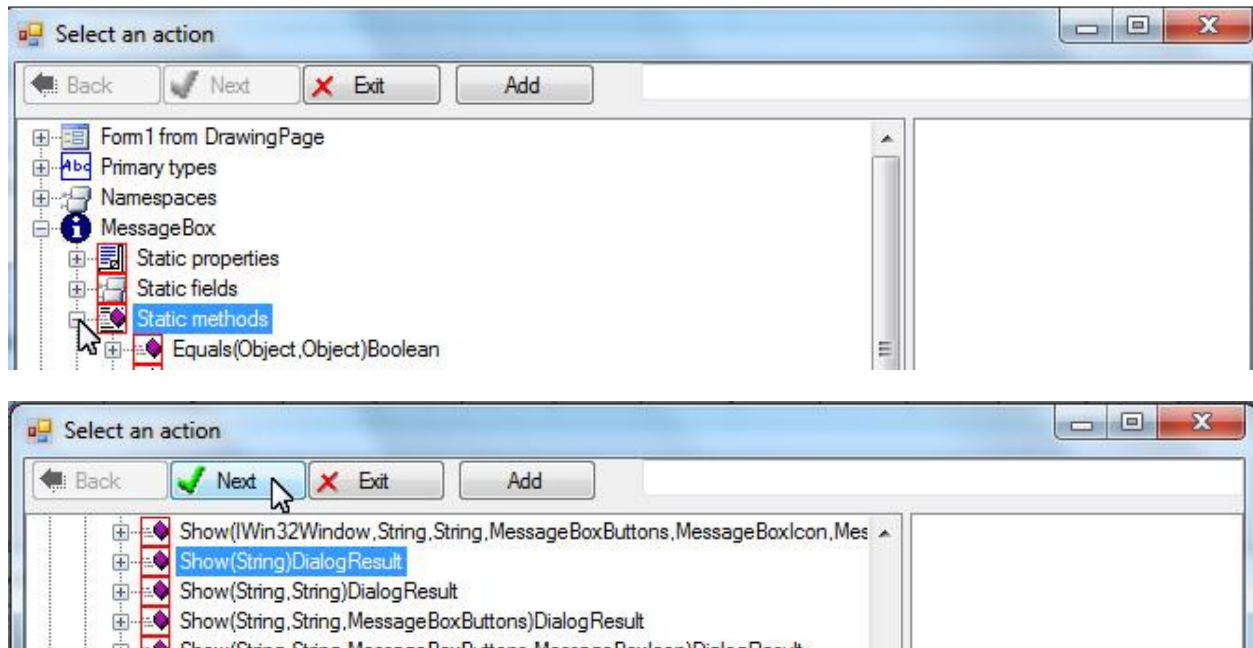
A Method Editor appears to let us add actions and arrange actions. When the Click event occurs the actions will be executed. Note the “sender” parameter. It indicates the button generating the event.



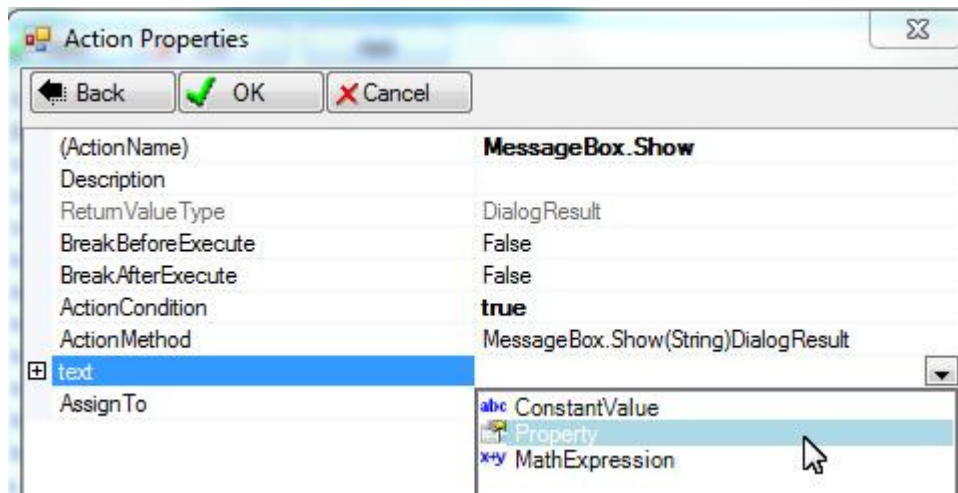
In this sample, let's add an action to display the button caption. Right-click the Action Pane (the middle pane); choose "Add an action":



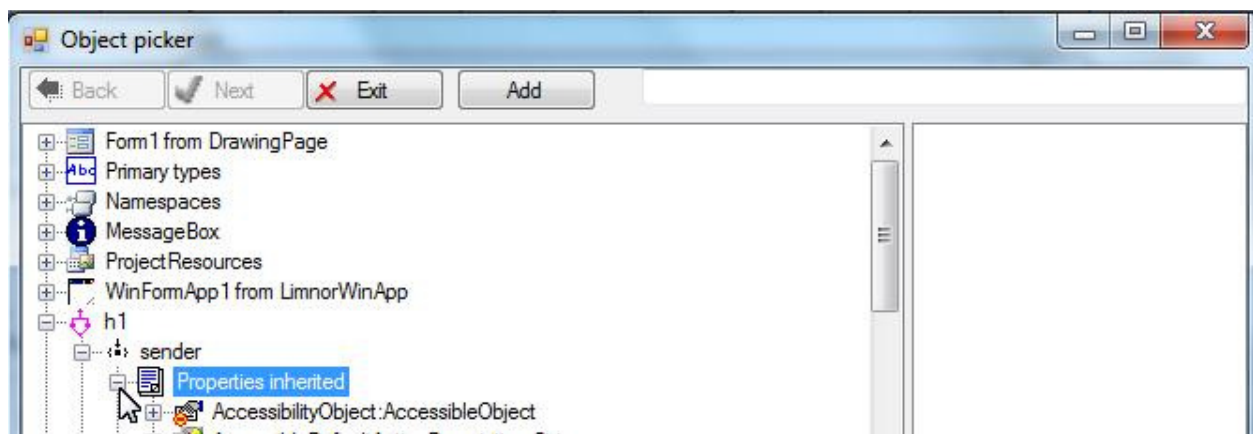
Choose Show method of the MessageBox class:



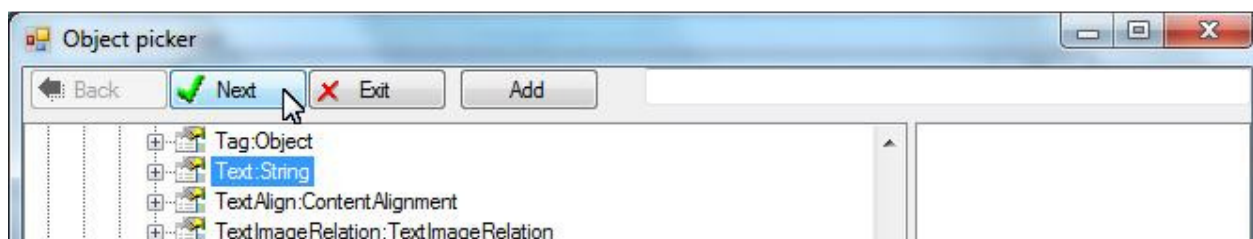
Select "Property" for the "text" parameter because we want to show button caption:



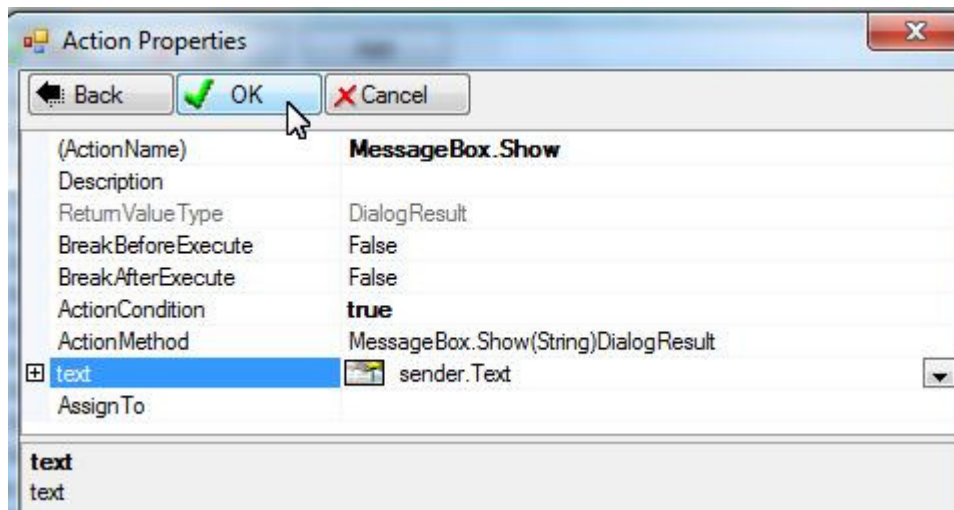
Locate the properties for “sender”:



Select the Text property of the “sender”:

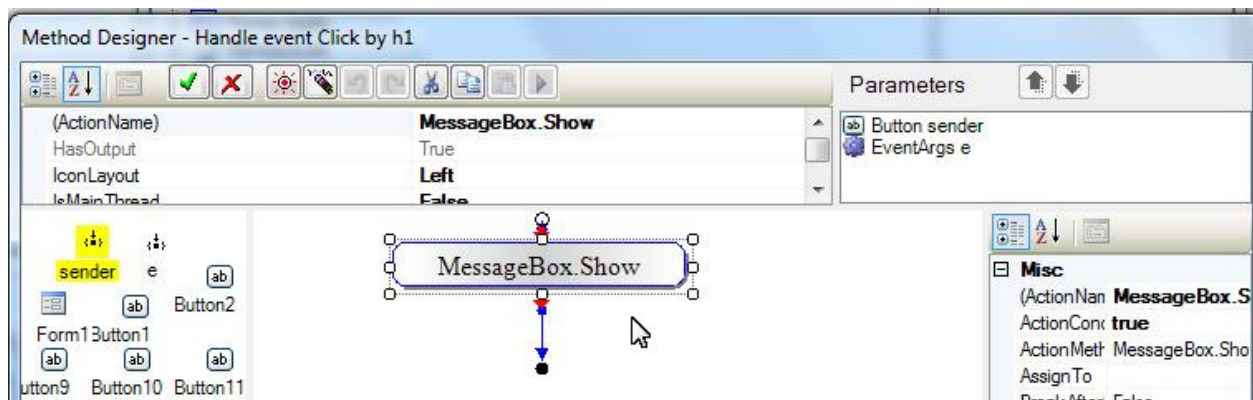


This is the action we want:



Note the use of “sender” in this action. We do not want to use the Text of any specific button. We want to use the Text of the “sender”.

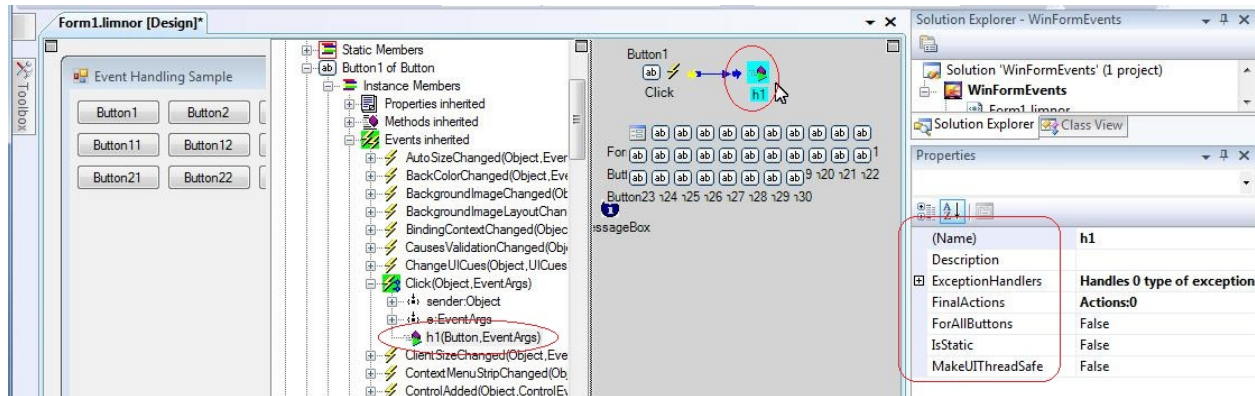
The action appears in the Action Pane:



We may add many actions and arrange the actions into a diagram. For this sample, we just use this single action. Click . The event handler method is created.

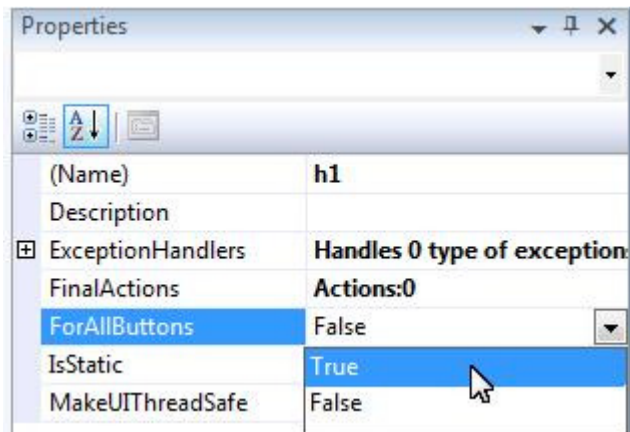
The event handler method appears in the Object Explorer and the Event Path Explorer:





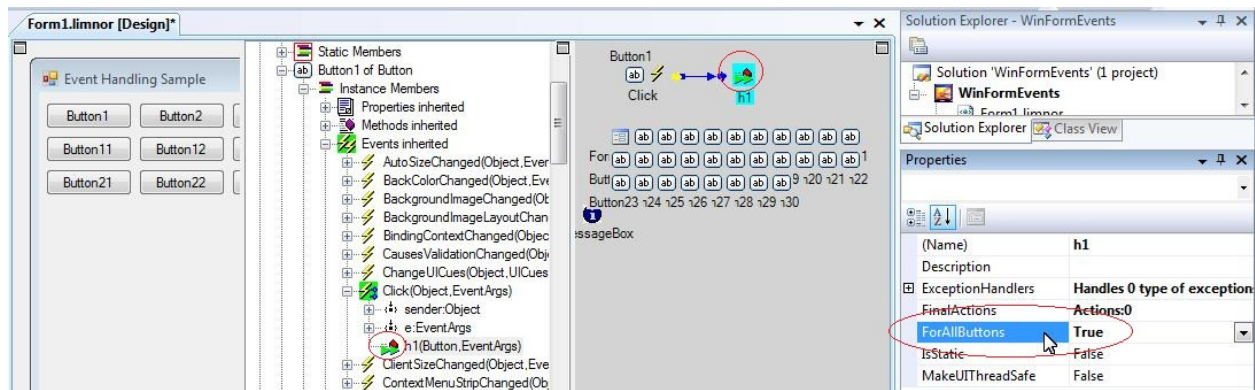


## Handle all types

By default an event handler method only responds to the object it is created for. You can let it response to all the objects of the same type, by simply set its `ForAll{types}` property to true. In this sample, since we created the above event handler method for a button, you will find a `ForAllButtons` property. Set it to True to make it response to all buttons:

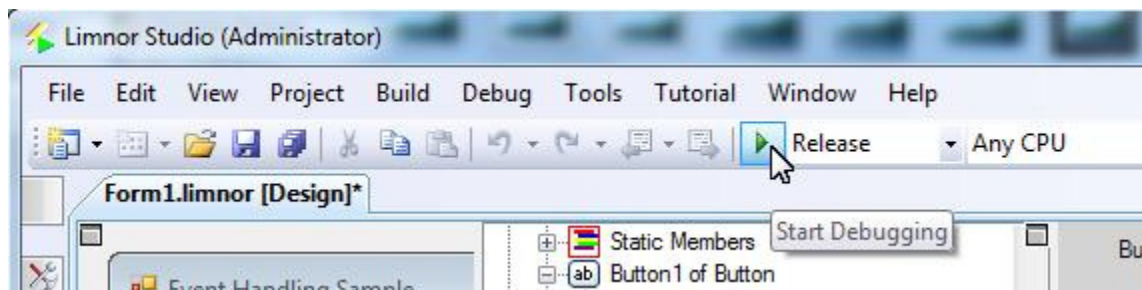


Note that on doing so, the icon for the event handler method changed from  to , indicating that it will respond to all the objects of the same type.

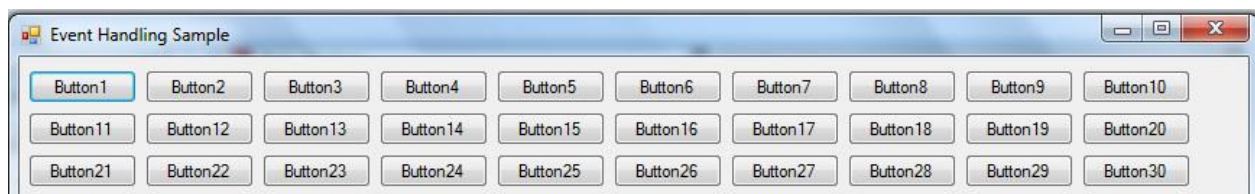


## Test

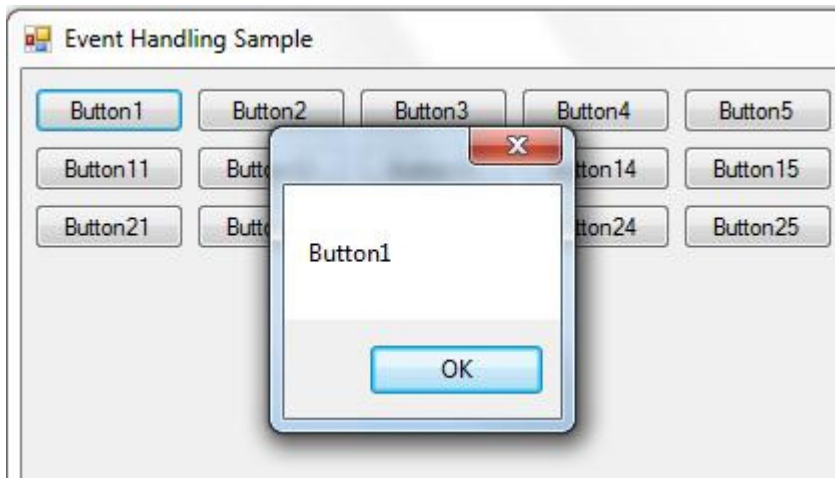
We may test the application now.



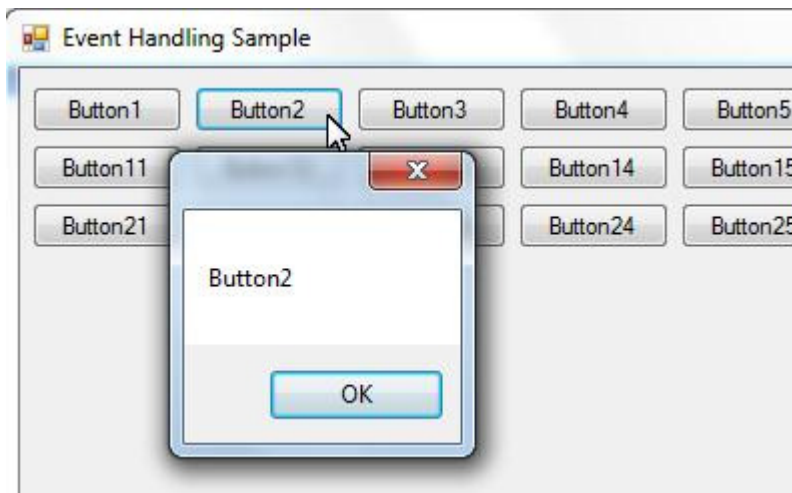
The form appears:



Click the first button. A message box appears, showing the caption of the first button, as we expected, since the event handler method was created for button 1.



Click another button, say, button 2. A message box appears too, showing the caption for button 2:



This is the effect of turning on "ForAllButtons".

- ✓ The event handler responded to the clicking of Button 2, even the event handler was not created for Button 2.
- ✓ The event handler correctly used the caption of Button 2, the event sender, not the caption of Button 1, even the event handler was created for Button 1.

Click each button; we see that every click triggers the correct message box.

## Exclude Objects from Event Handler

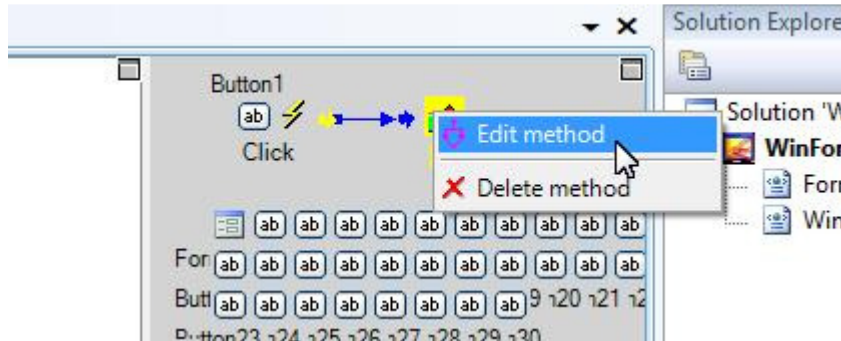
In many cases we do not want the event handler to be triggered for certain objects. For different types of objects, and for different business requirements, different ways can be used to exclude unwanted objects.

Here we show one simple way of doing it.

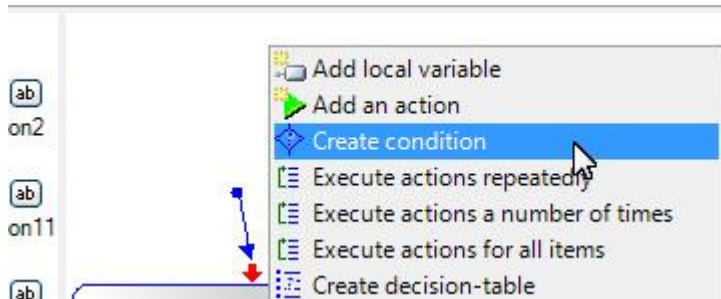


Every Control has a Name property. We may use such a logic that if a button's Name does not contains certain text then we exclude the button.

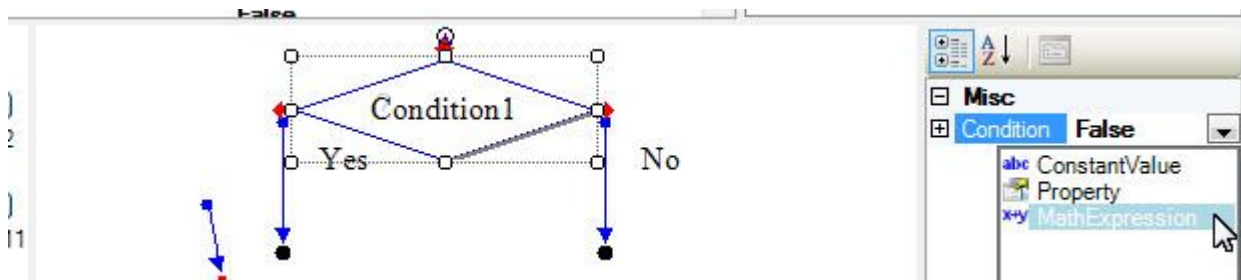
Open the event handler:



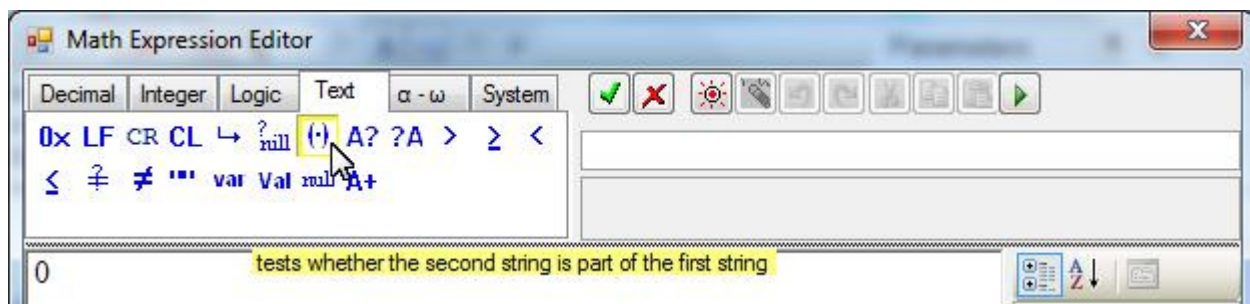
Create a "Condition":



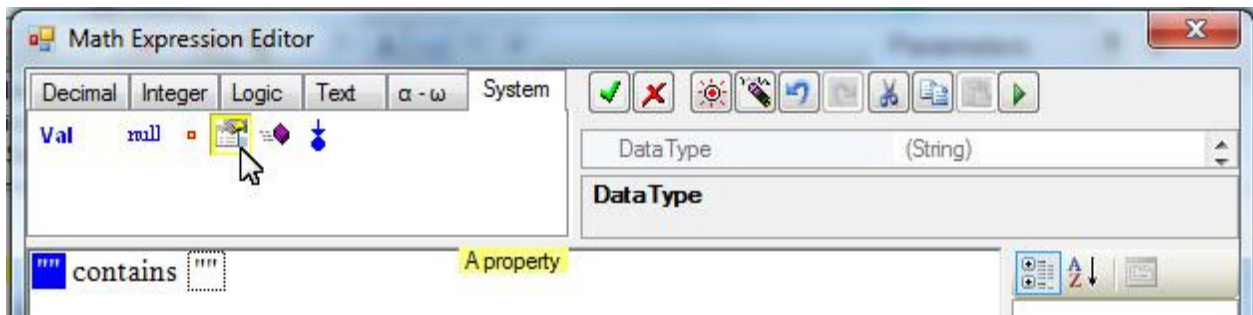
Set the condition to a Math Expression:



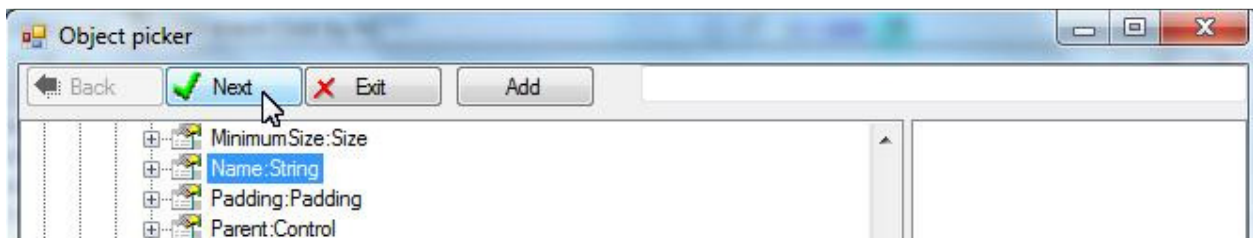
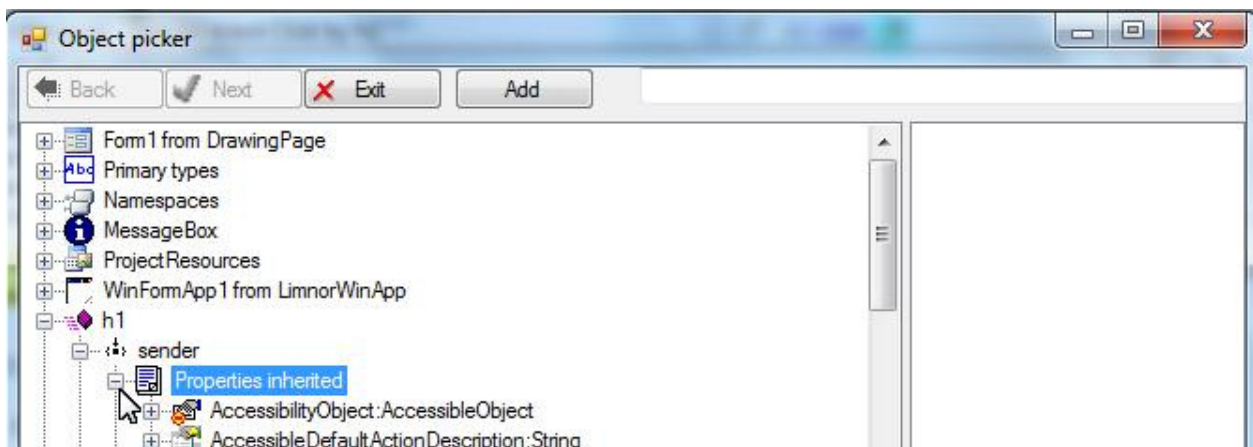
Click "(.)" icon:



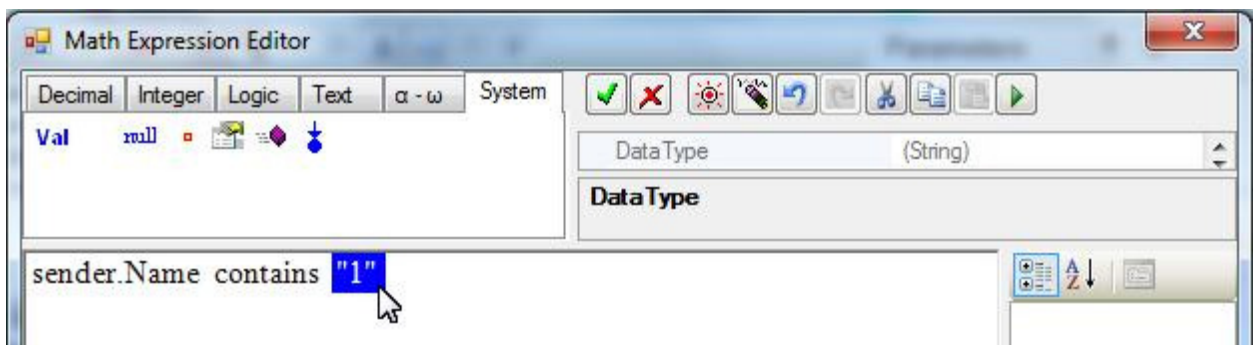
Select the first item and click the Property icon:



Select the Name property of the “sender”:

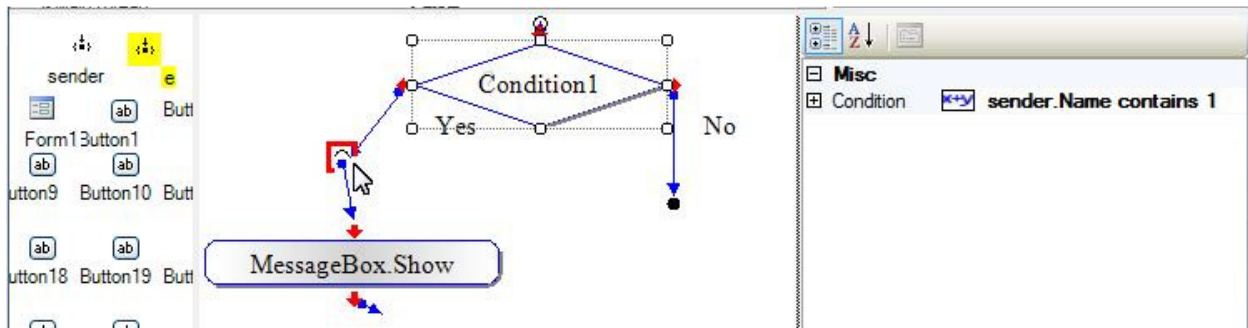


Select the last item; type 1:

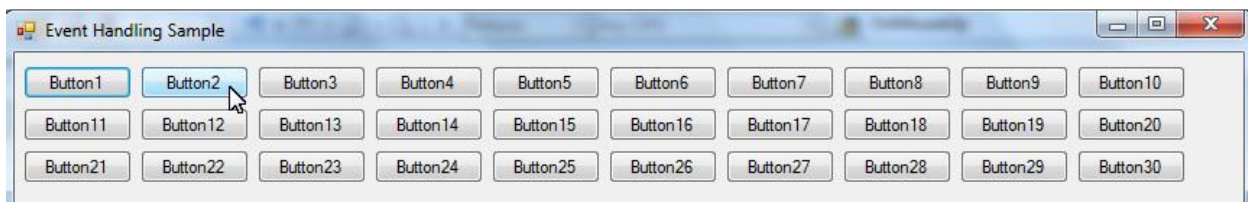


This expression gives a condition that the Name contains text “1”.

We may link the rest of the actions to the Yes port or to the No port, depending on our business logic. For this example, we link the action to the Yes port:



Let's test the application again. Click Button 2, nothing happens:



Click Button 12, a message box appears:

