Create Tabbed Web Browser

Contents

Introduction

When a web browser control is used in a Windows Form application, if a web page link leads to a popup window then a default web browser is launched to show the popup window. This behavior is not desirable in some cases, because a popup window in a default web browser is out of control of the Windows Form application.

This sample provide such a solution: a tab control is used to host web browser controls in each tab page. When a popup window is going to launch, we cancel the popup, and create a new tab page with a new web browser control on the tab page to show the web page for the original popup window.

The sample is a class library project to create a DLL. The project contains two classes. One class is named TabControlWebBrowser, which is derived from class TabControl. The other class is named TabPageWebBrowser, which is derived from class TabPage.

TabControlWebBrowser

We create two methods on this class.

AddNewWebPage

This method takes a parameter for web page address. It adds a new tab page of type TabPageWebBrowser and loads the web page on the tab page.

GotoURL

This method takes a parameter for web page address. It loads the web page into the currently selected tab page. If there is not a tab page then call AddNewWebPage to create a new tab page to load the web page.

TabPageWebBrowser

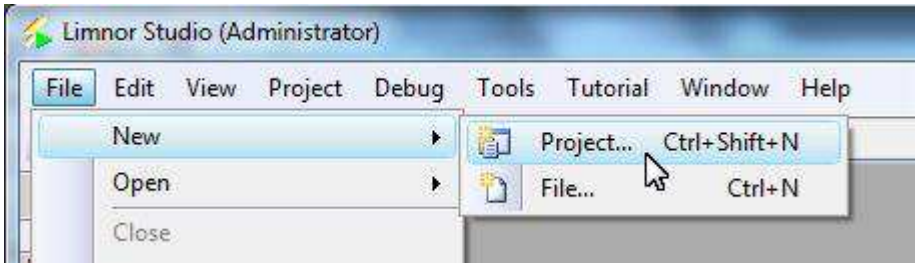
We add a web browser control on this class.

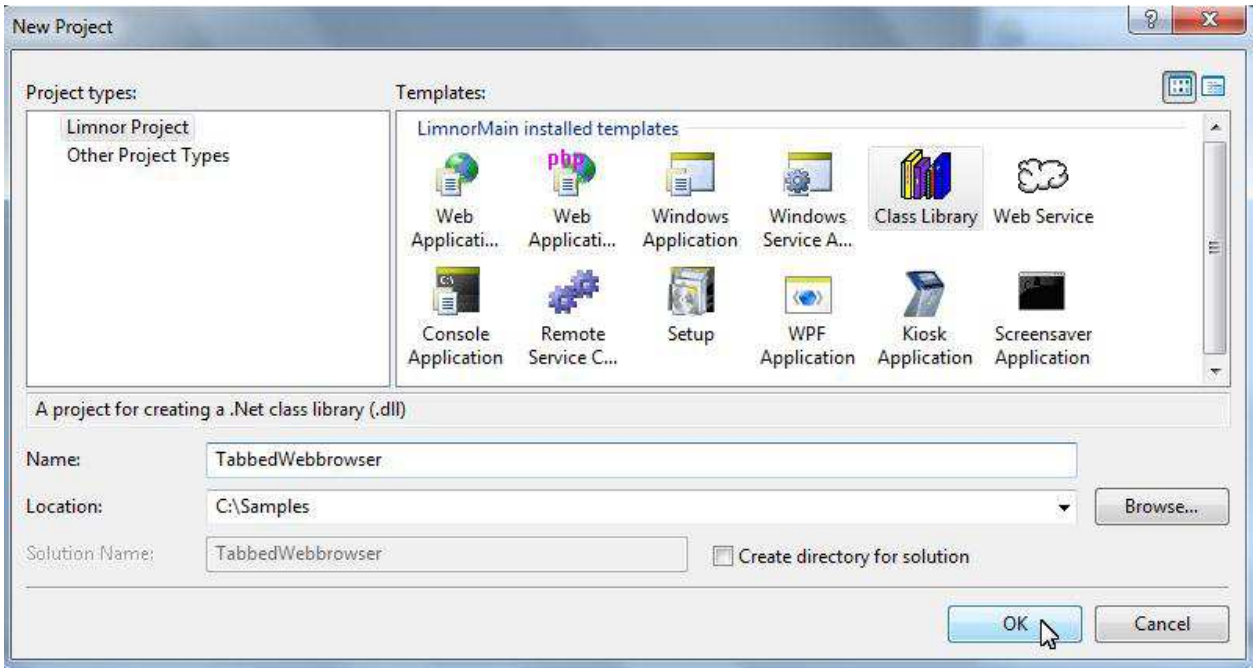
We create one method named GotoURL on this class. This method takes a parameter for web page address. It executes a Navigate action with the web page address to load the web page into the web browser control.

We handle event StartNewWindow to cancel the popup and execute an AddNewWebPage action to show the popup web page in a new tab page.

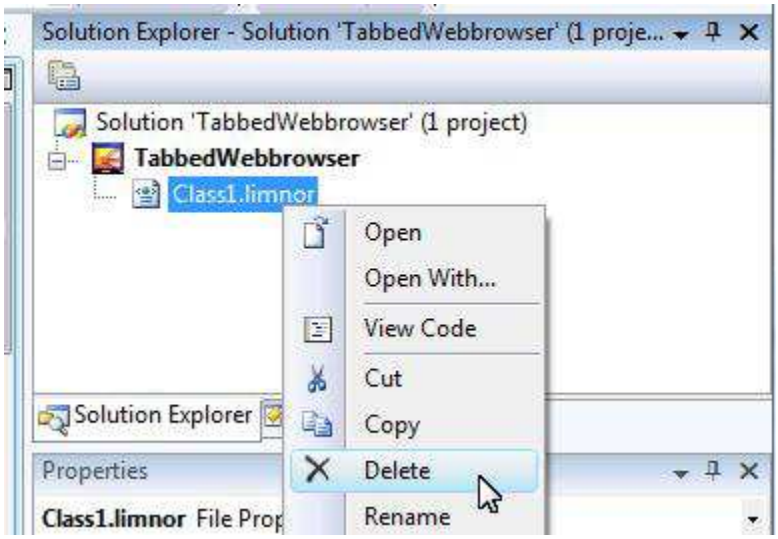
Create Class Library Project

Create a new class library project:



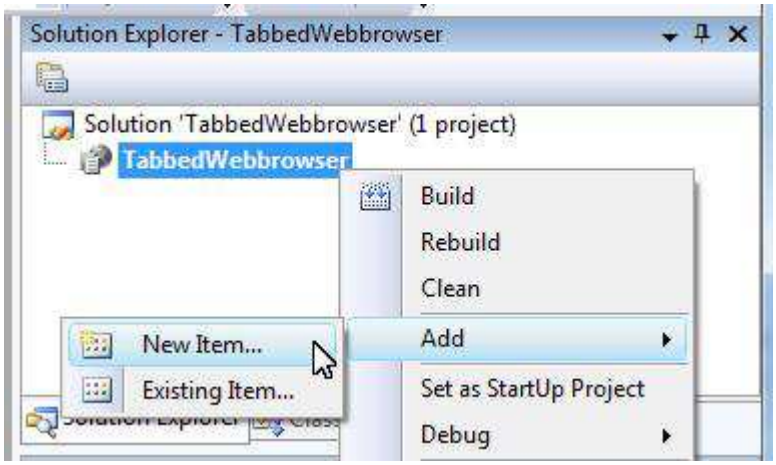


Delete Class1 because we are not using it:

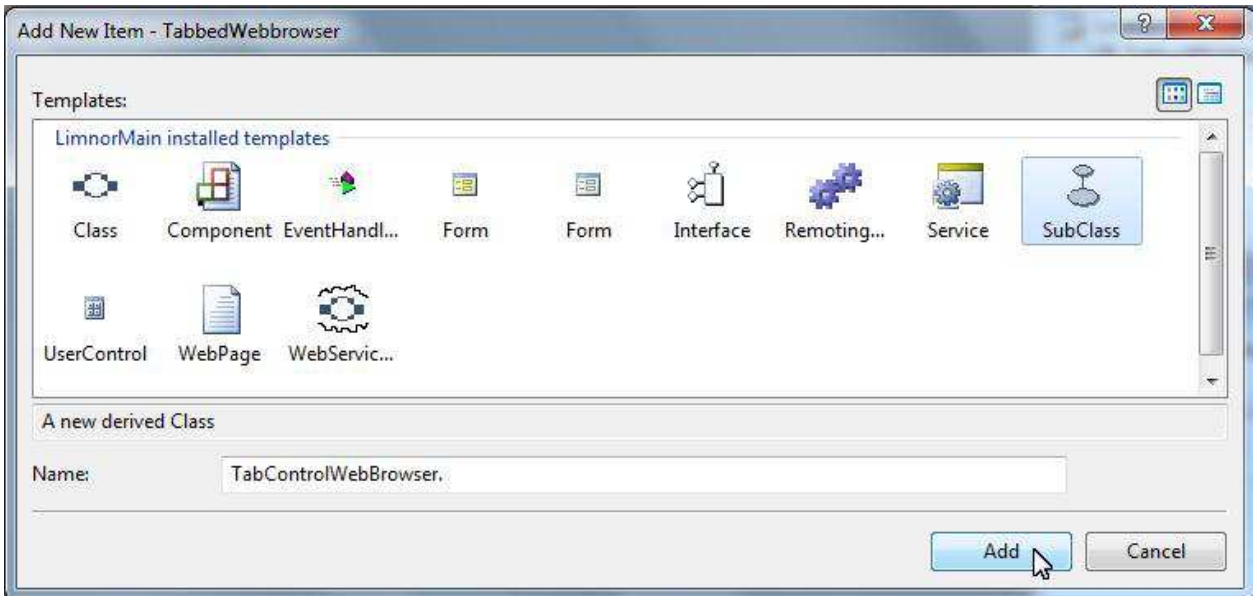


Create TabControlWebBrowser

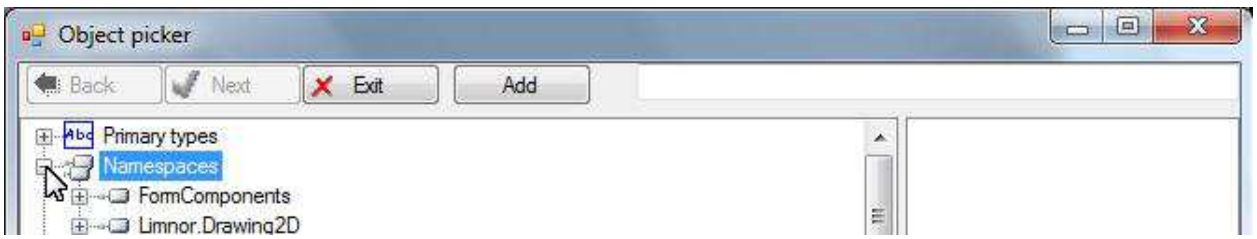
Add a new class:

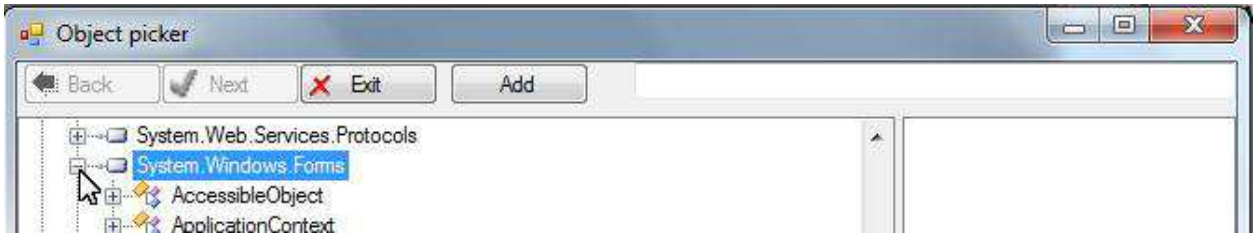


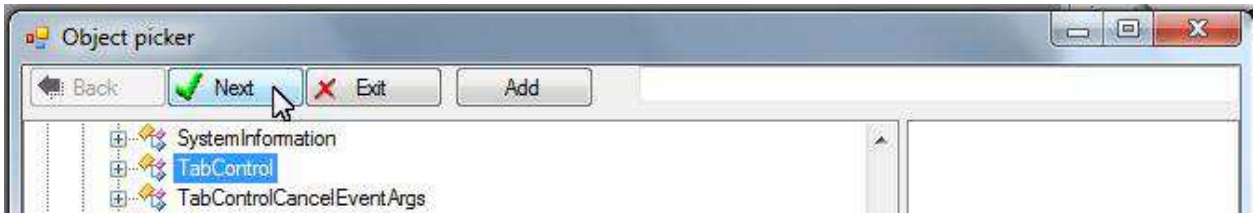
Select SubClass so that we may choose a base class:



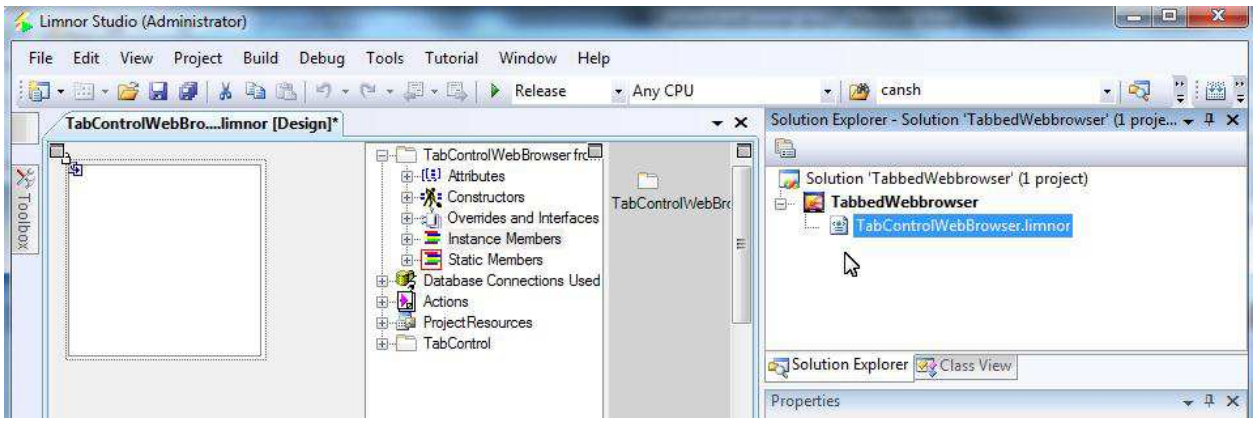
Select TabControl as the base class:





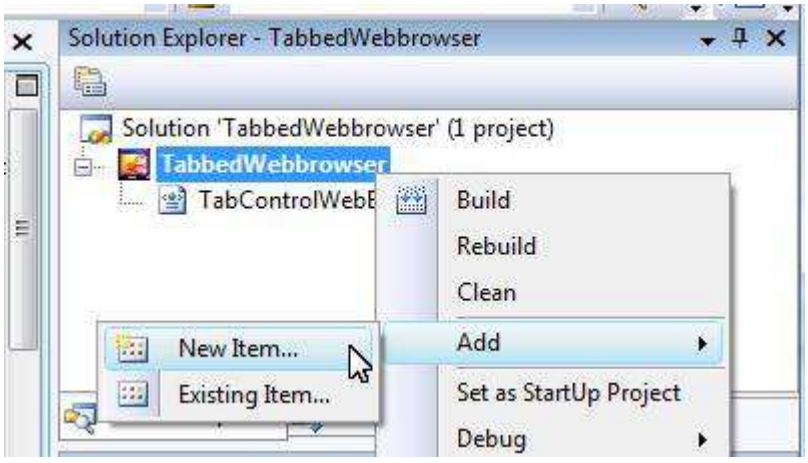


The new tab control class is created:

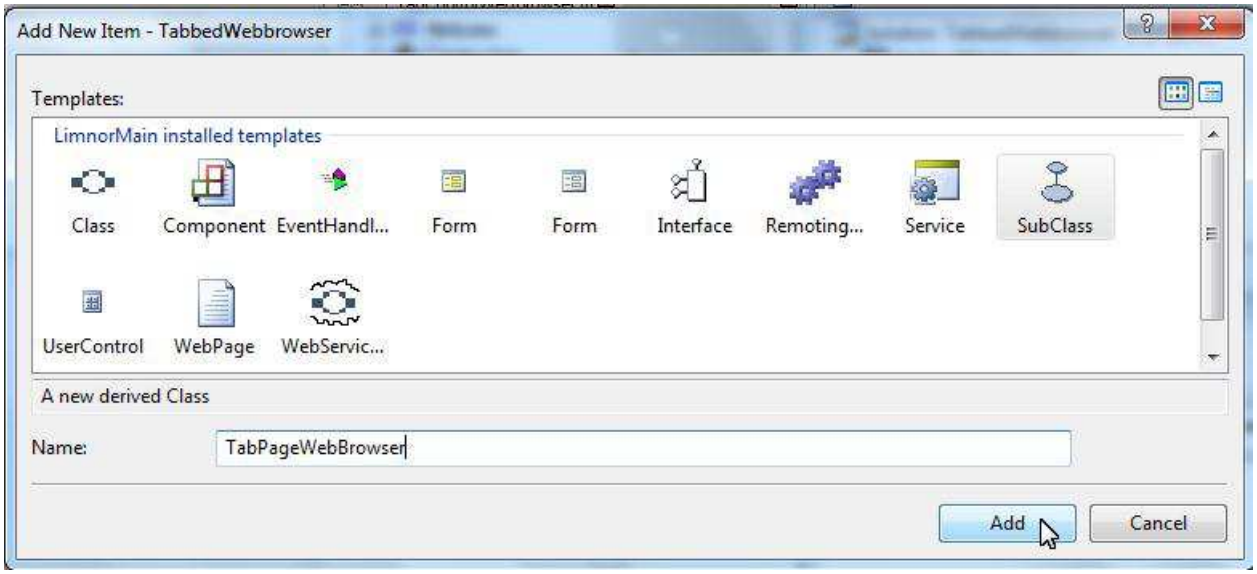


Create TabPageWebBrowser

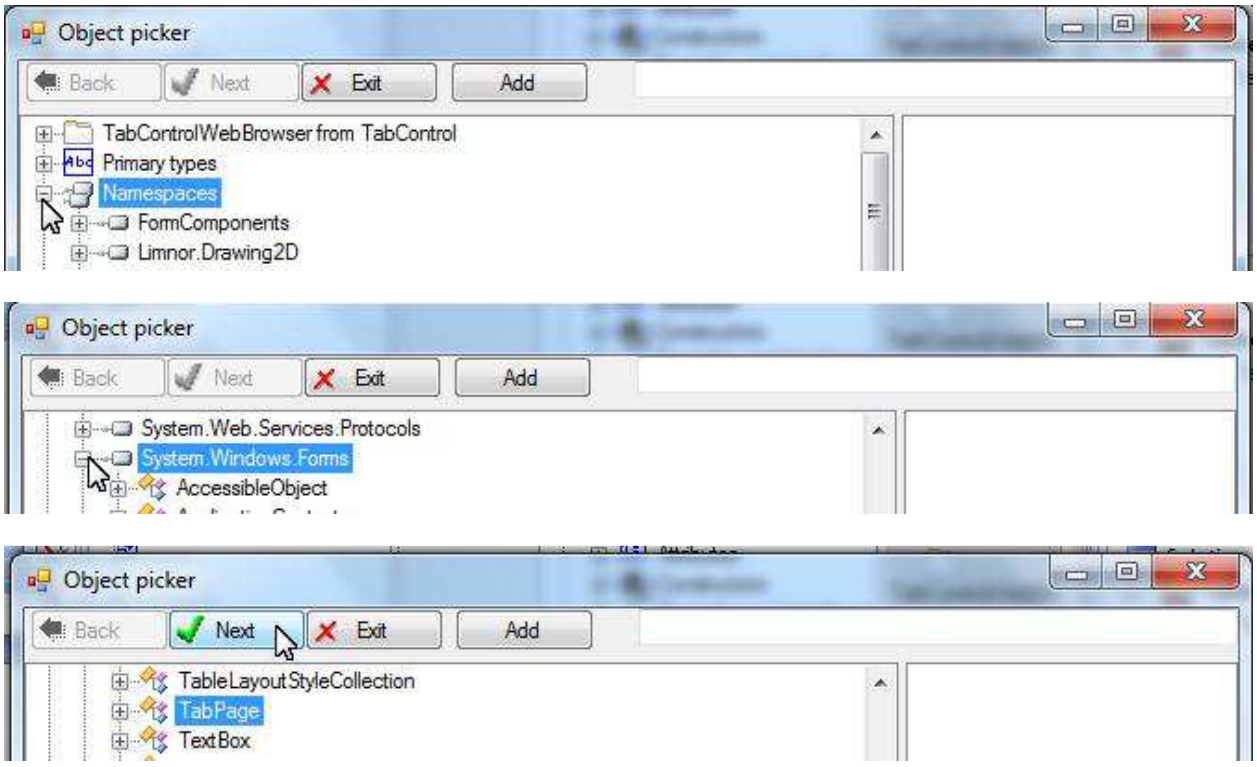
Add a new class:



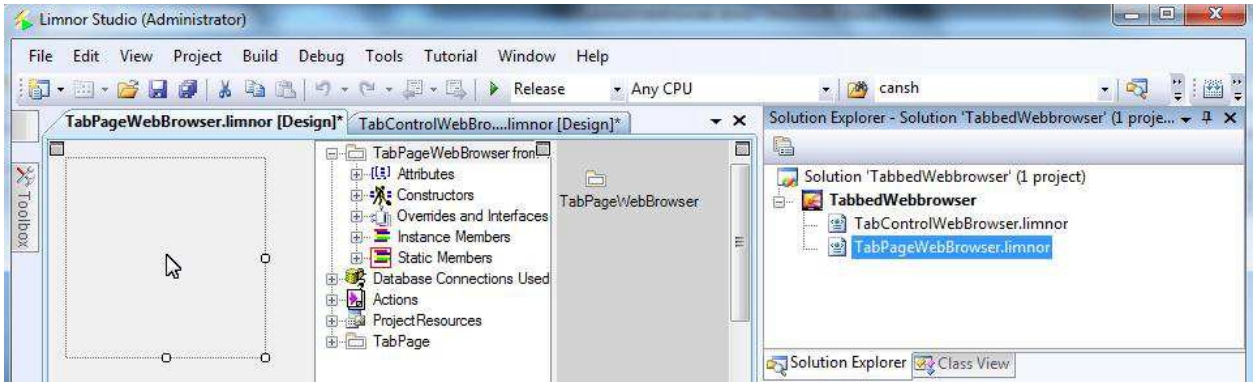
Select SubClass so that we may choose a base class:



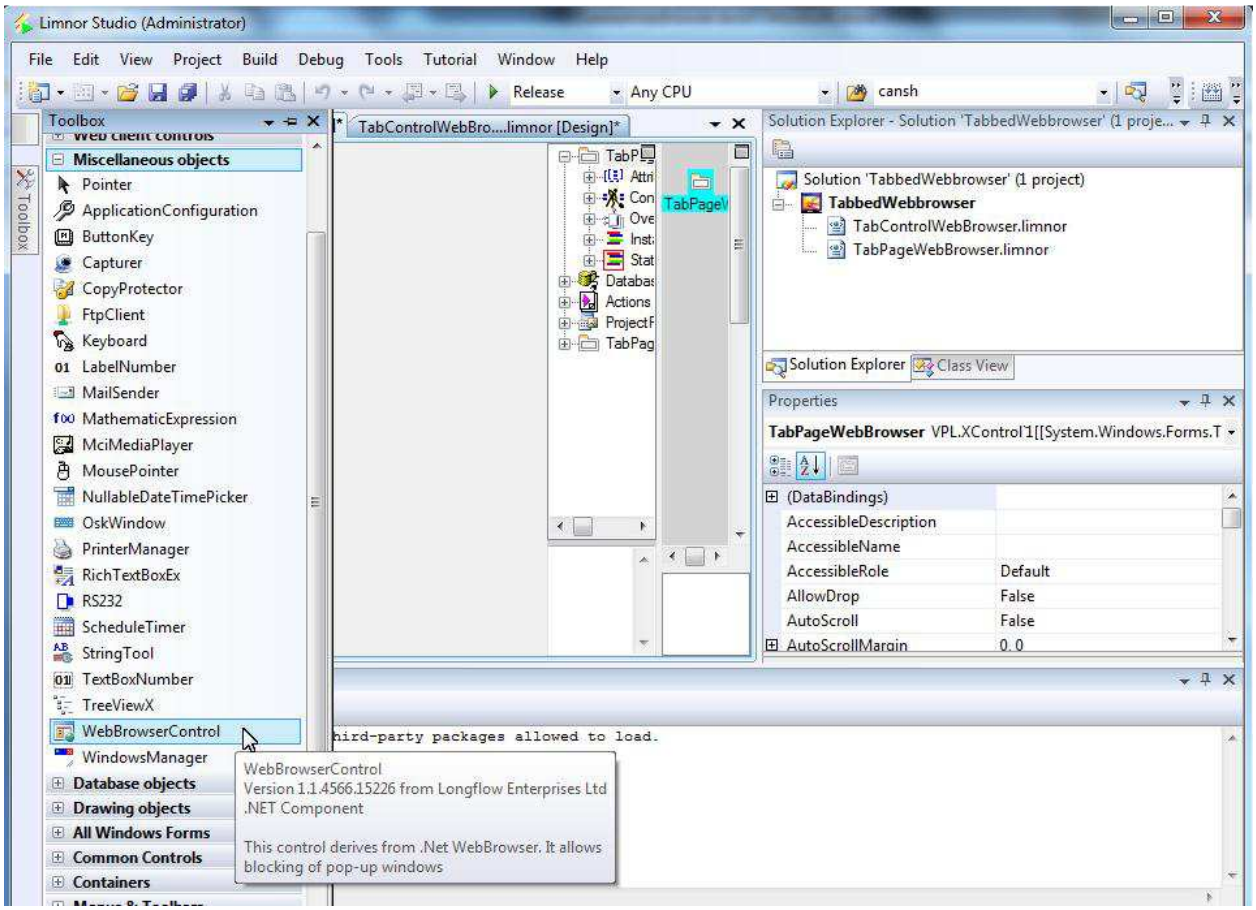
Select TabPage as its base class:



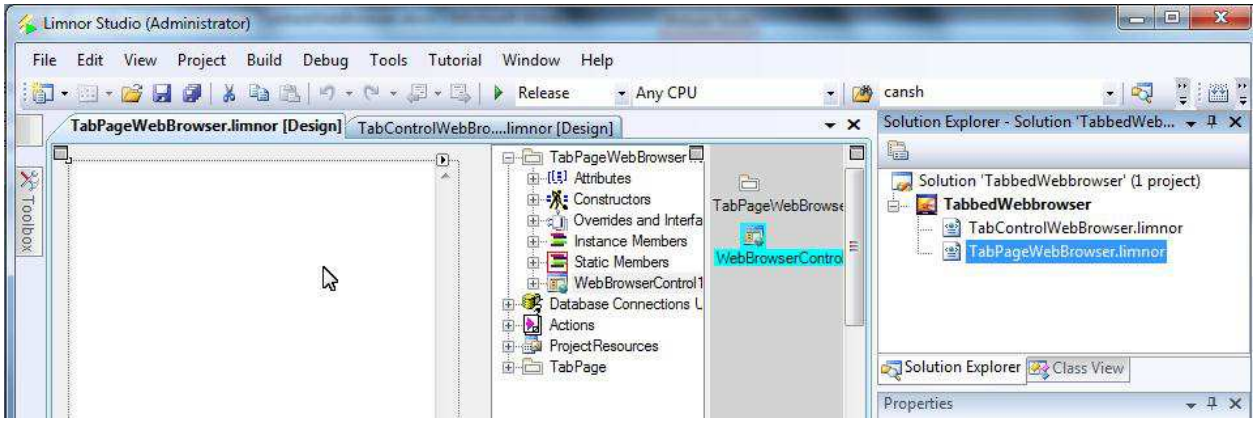
A new tab page control is created:



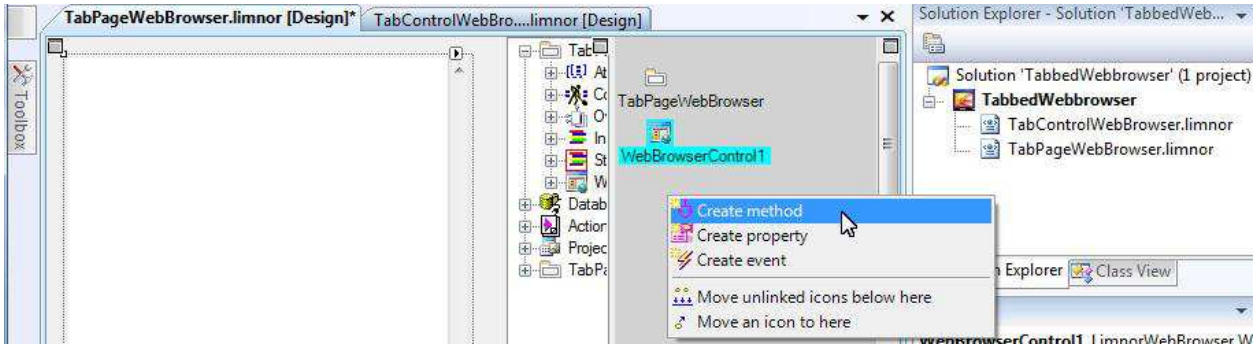
Add a web browser control to the new tab page class:



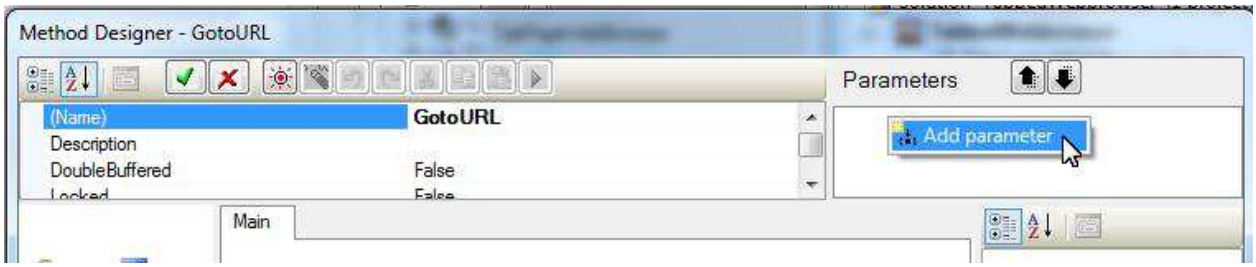
A web browser control appears in the tab page control:



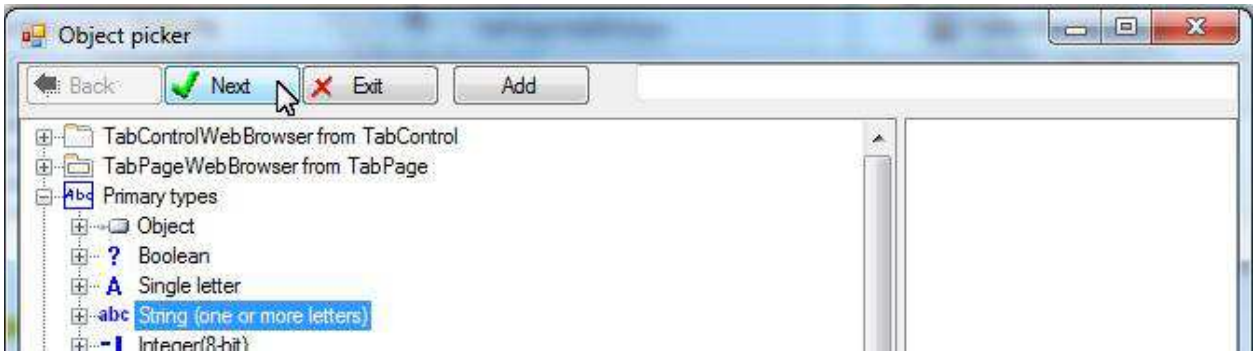
Create TabPageWebBrowser method – GotoURL



Change its name to GotoURL and add a parameter:

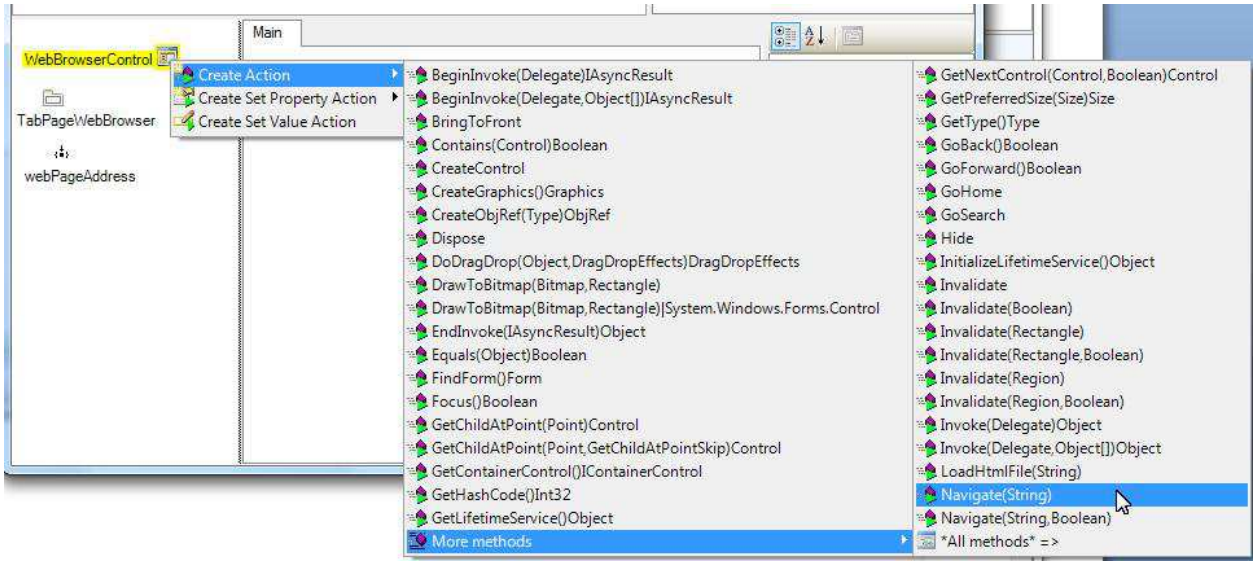


Select “String” as the parameter type:

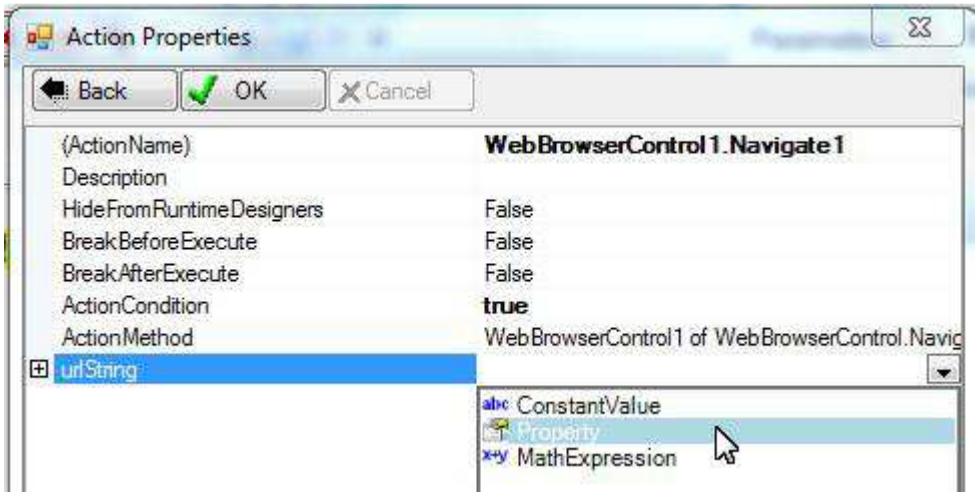


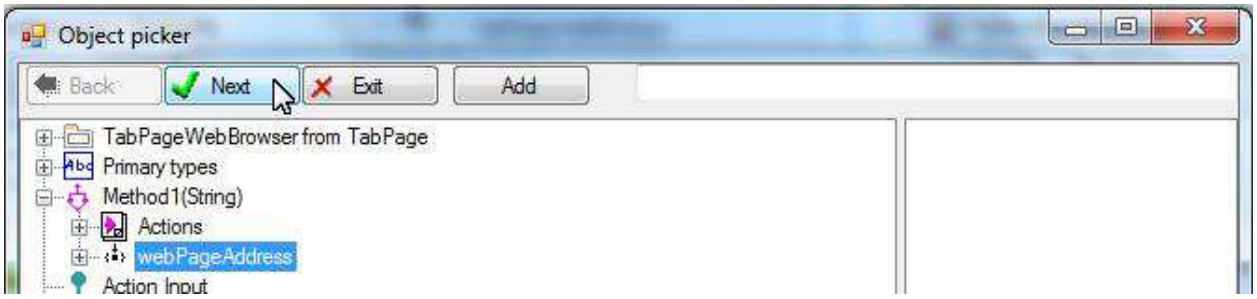
Rename the parameter to webPageAddress

Create a Navigate action for this method:

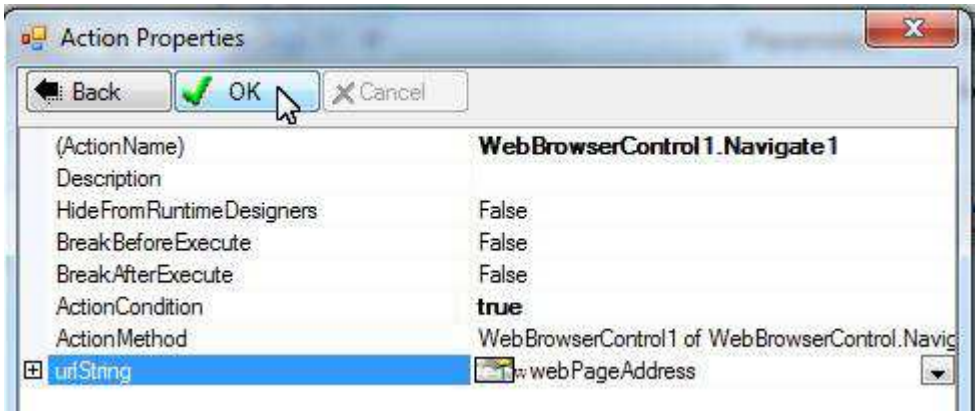


Use the webPageAddress for the action:



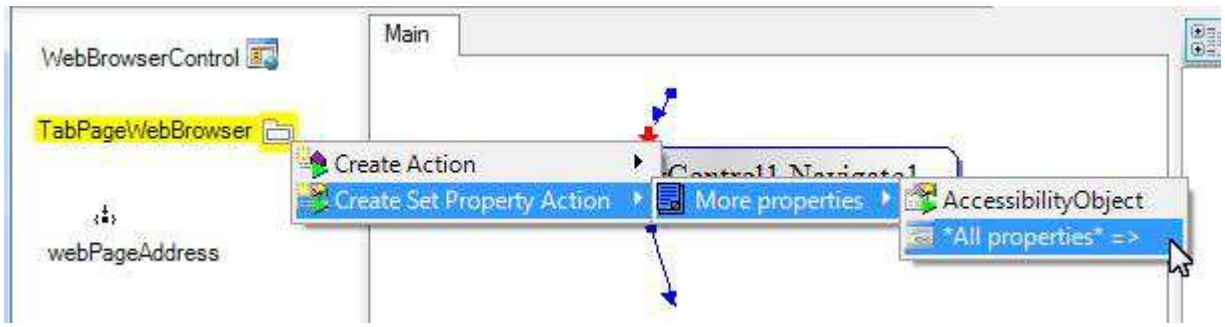


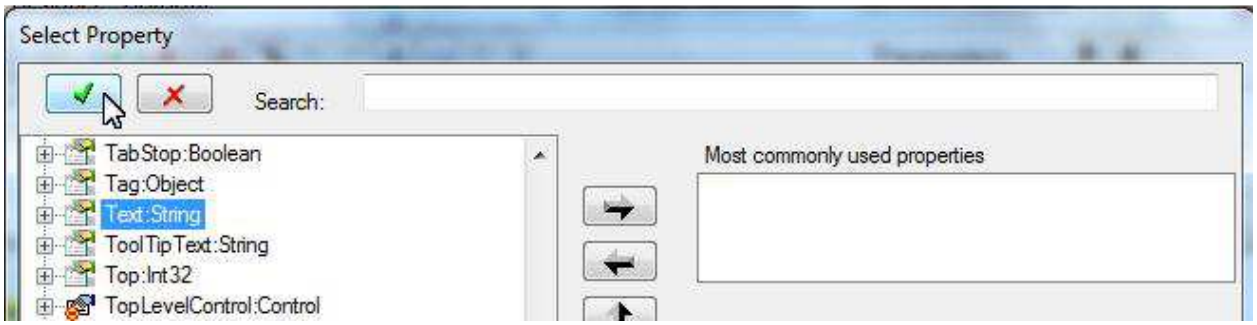
Click OK:



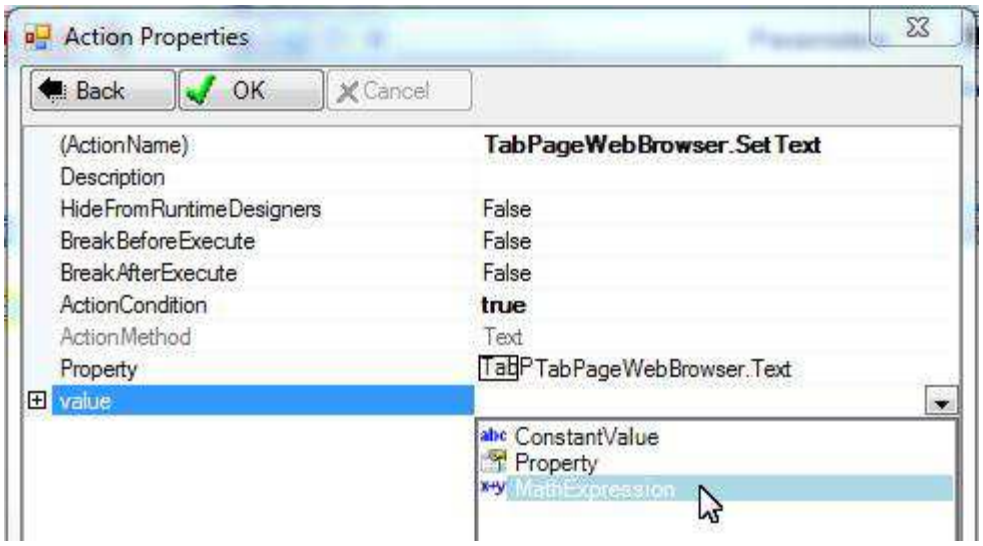
The action appears.

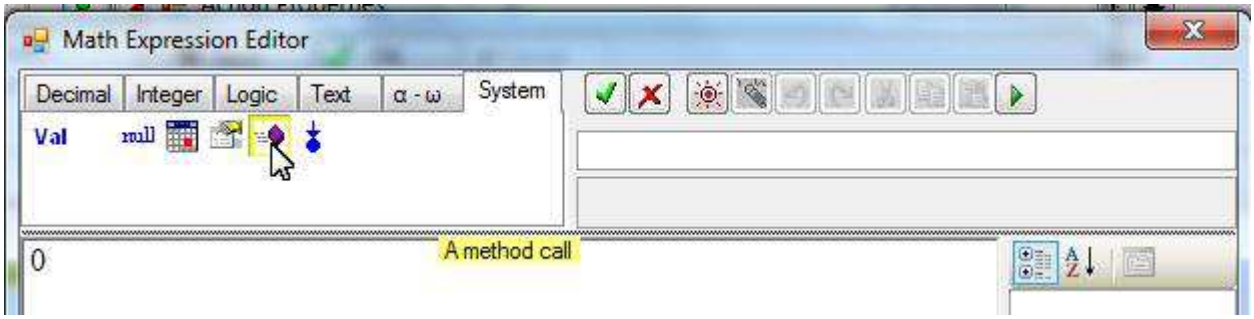
Add an action to show the web page address to the tab caption, which is the Text property:

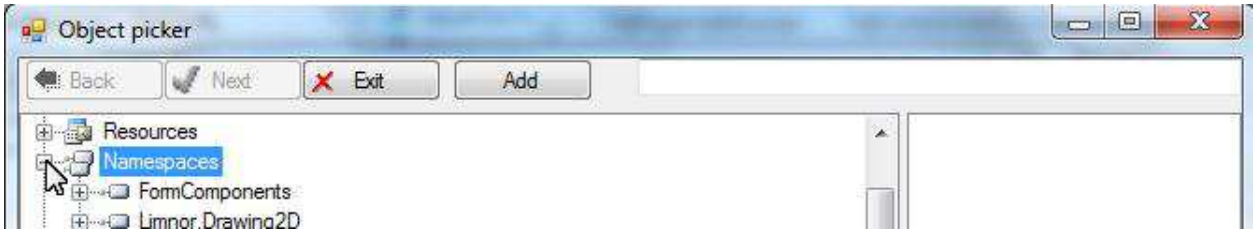


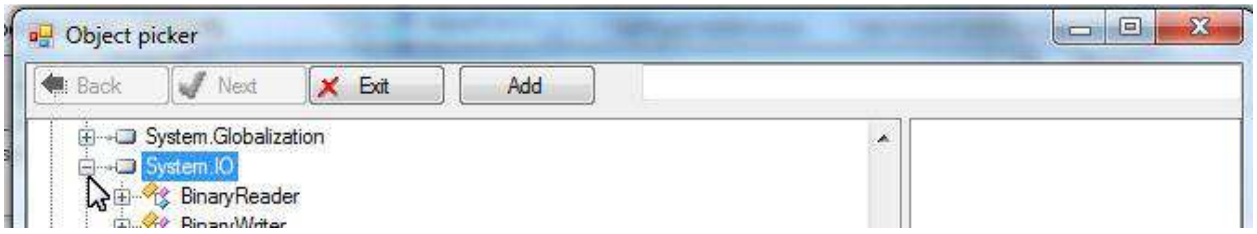


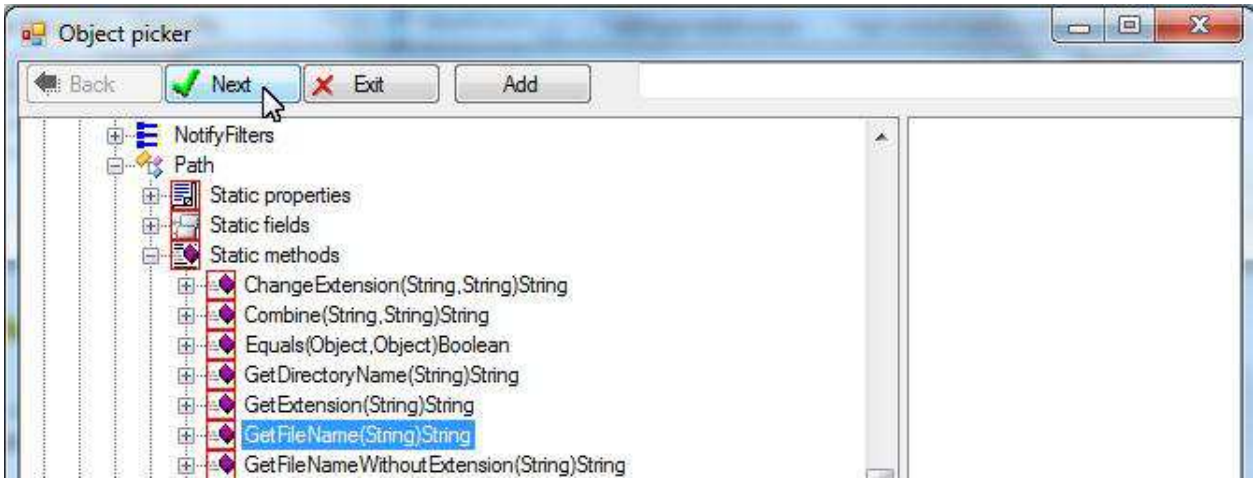
Use Path class' GetFileName method to get only the last part of the web page URL:

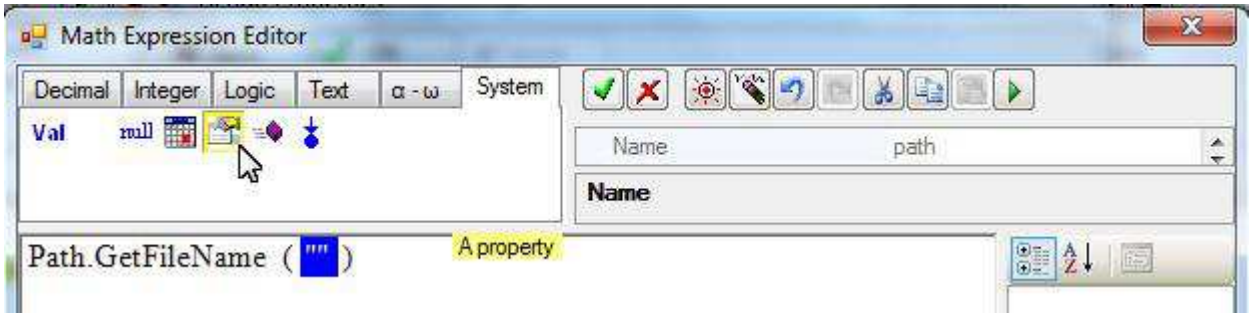

Action Properties
Back
OK
Cancel
(ActionName)
TabPageWebBrowser.SetText
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
Text
Property
TabPTabPageWebBrowser.Text
value
ConstantValue
Property
MathExpression

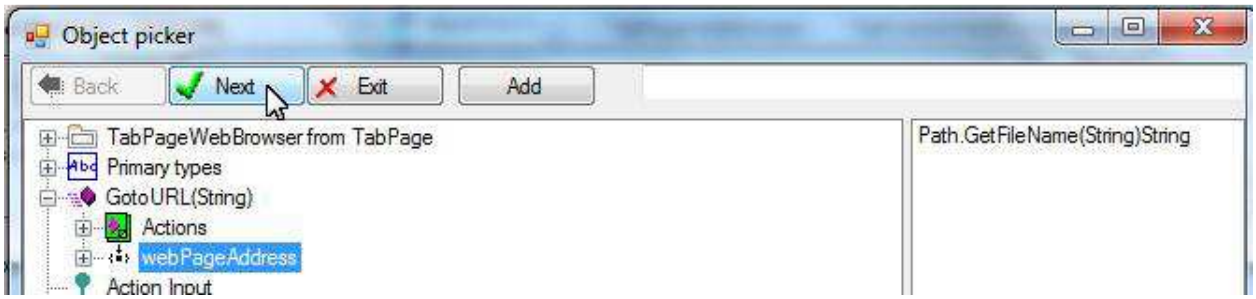

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
A method call
0

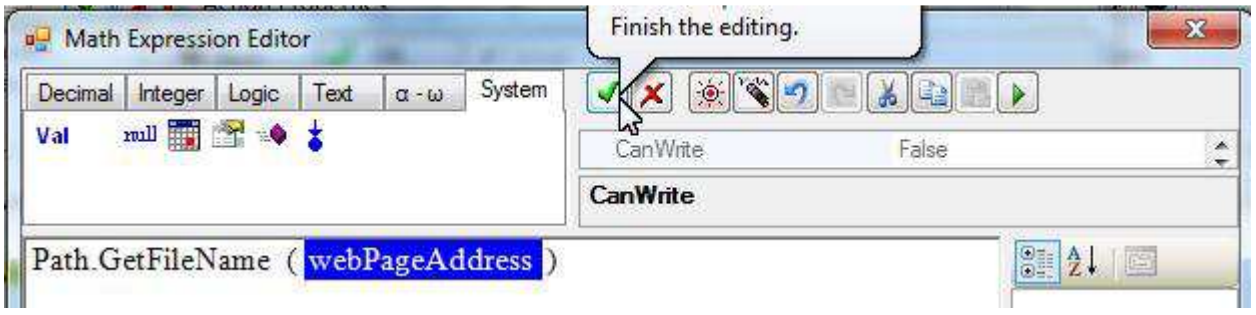

Object picker
Back
Next
Exit
Add
Resources
Namespaces
FormComponents
Limnor.Drawing2D

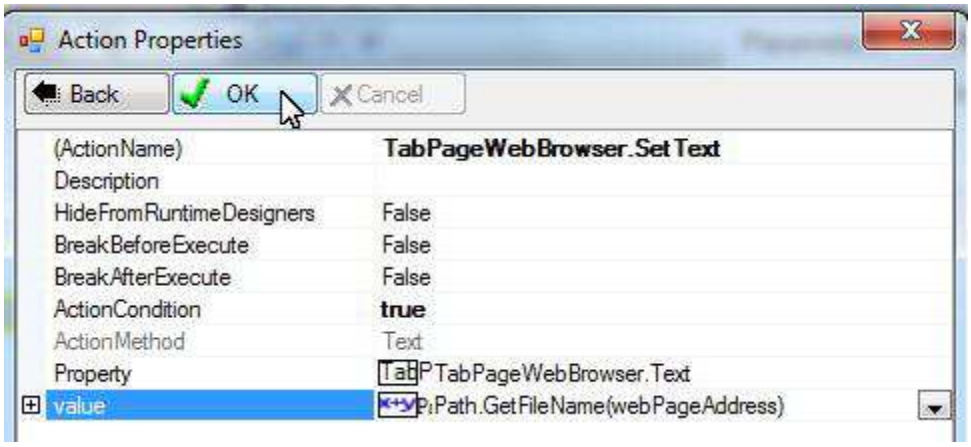

Object picker
Back
Next
Exit
Add
System.Globalization
System.IO
BinaryReader


Object picker
Back
Next
Exit
Add
NotifyFilters
Path
Static properties
Static fields
Static methods
ChangeExtension(String,String)String
Combine(String,String)String
Equals(Object,Object)Boolean
GetDirectoryName(String)String
GetExtension(String)String
GetFileName(String)String
GetFileNameWithoutExtension(String)String

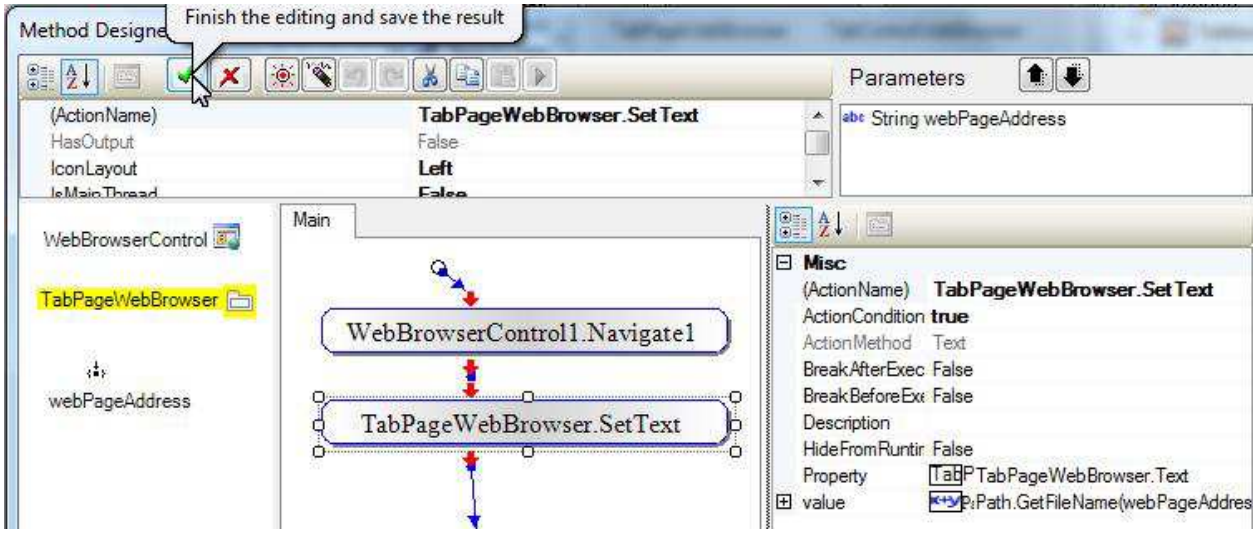

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
Name
path
Name
Path.GetFileName ("")
A property


Object picker
Back
Next
Exit
Add
TabPageWebBrowser from TabPage
Primary types
GotoURL(String)
Actions
webPageAddress
Action Input
Path.GetFileName(String)String

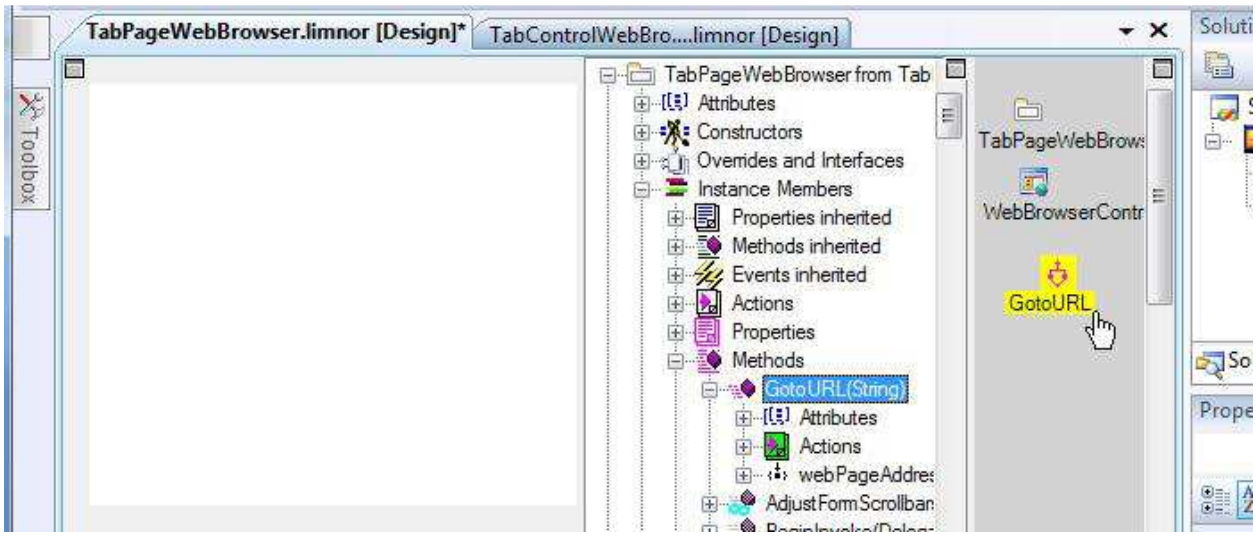

Finish the editing.
Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
CanWrite
False
CanWrite
Path.GetFileName (webPageAddress)



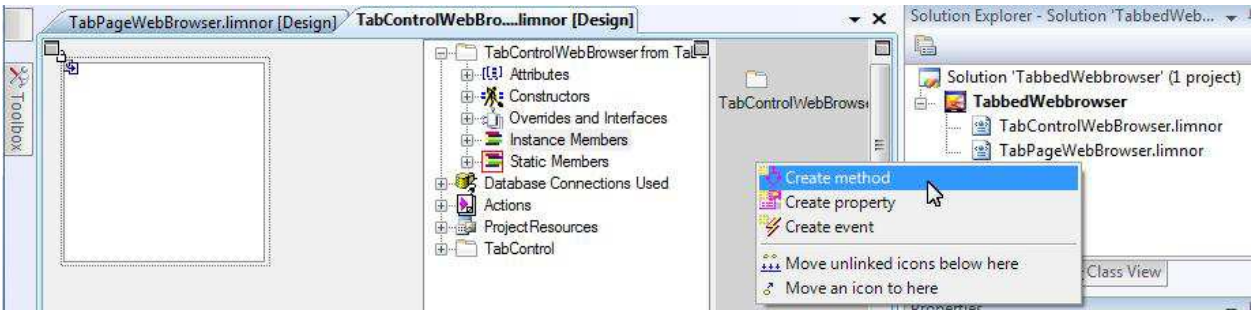
Link it to the last action. That is all for this method:



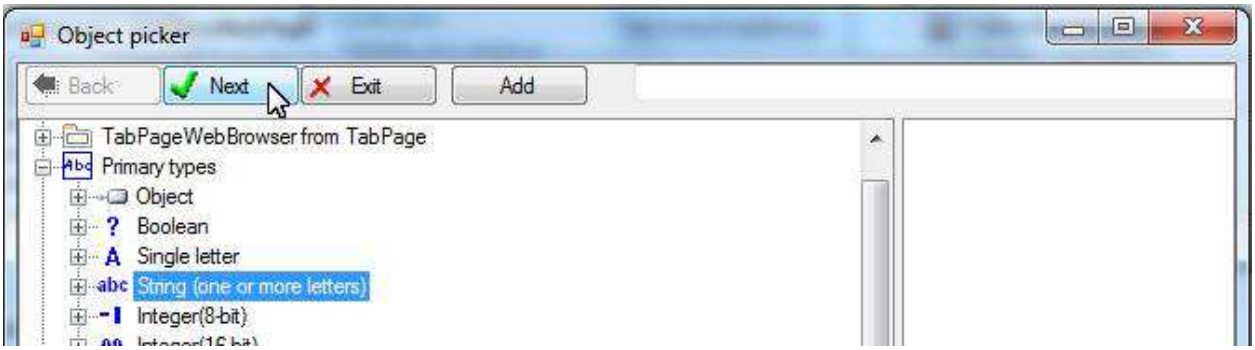
The new method appears:



Create TabControlWebBrowser method – AddNewWebPage

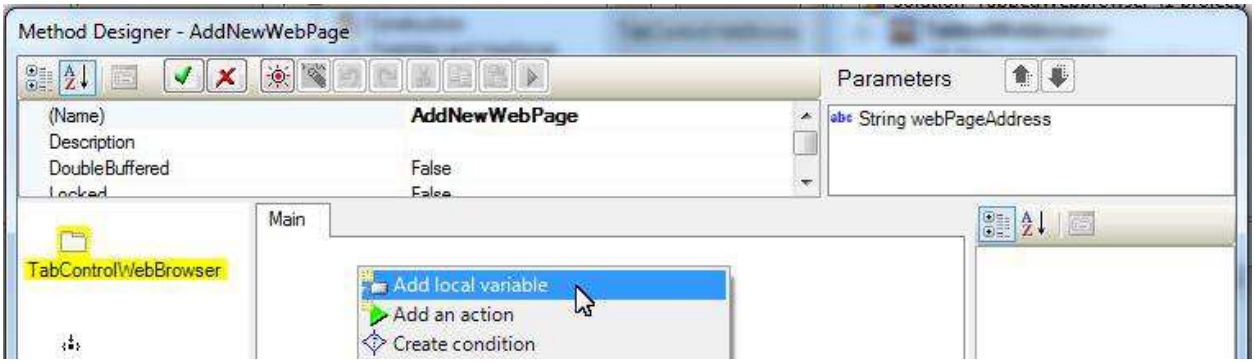


Name it AddNewWebPage and add a new String parameter:

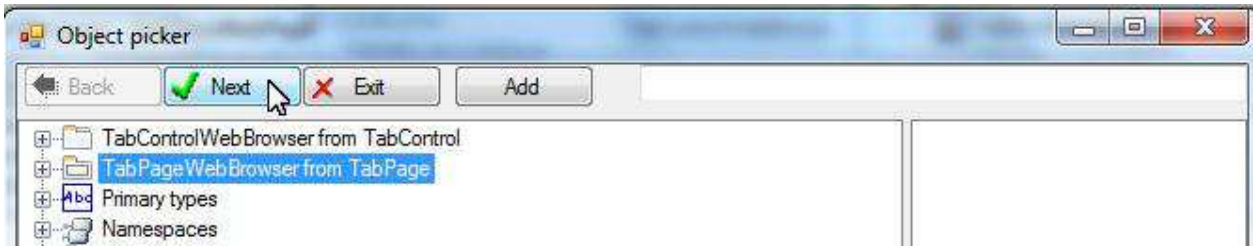


Rename the parameter to webPageAddress:

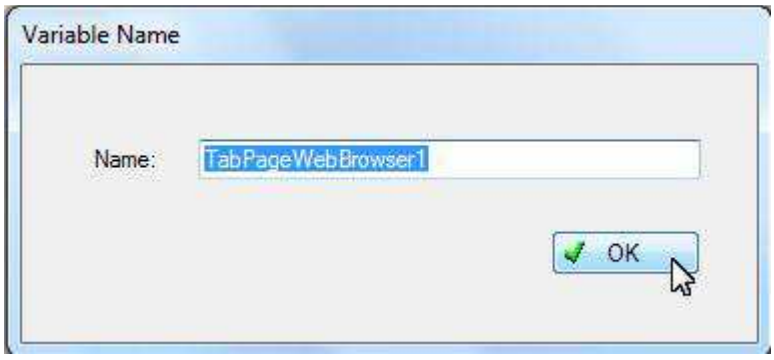
Create a new TabPageWebBrowser to be added to the tab control:



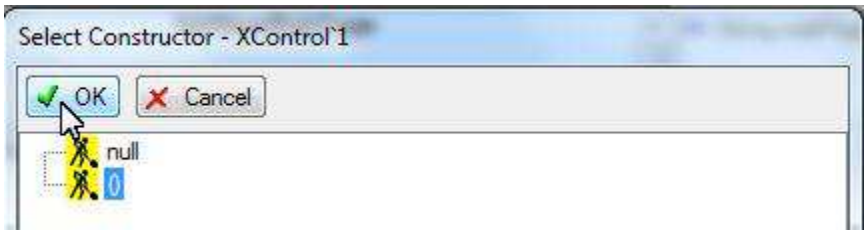
Select TabPageWebBrowser:



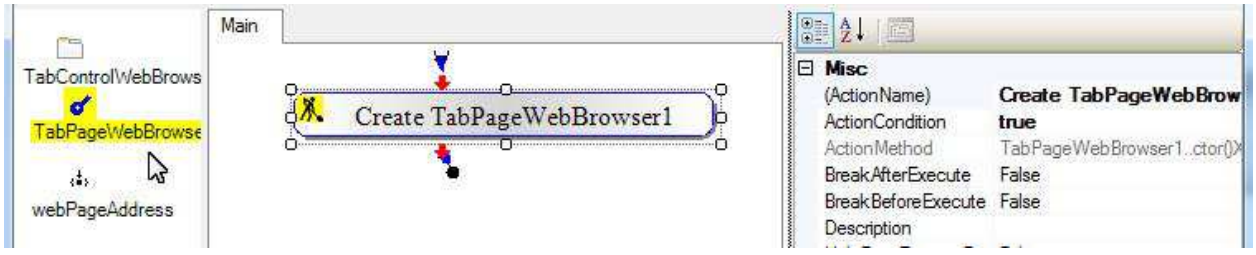
Choose a desired variable name



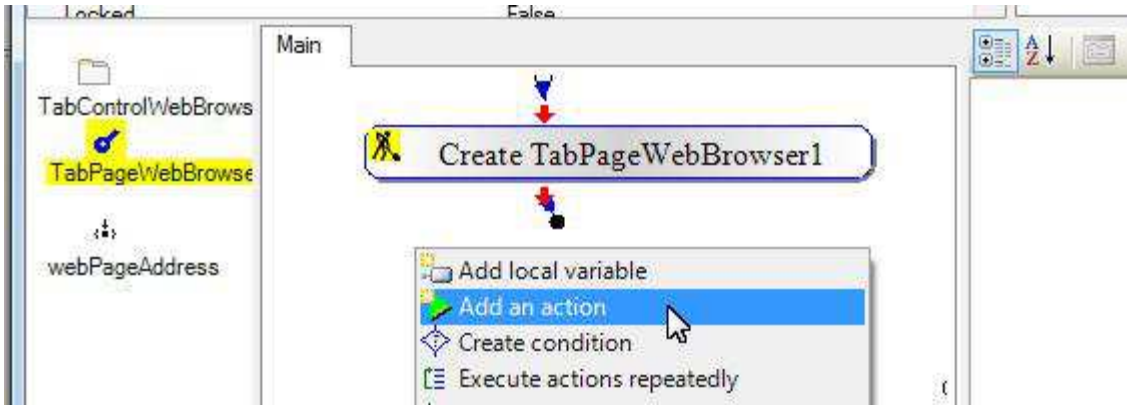
Select "()" to create a new instance of TabPageWebBrowser:



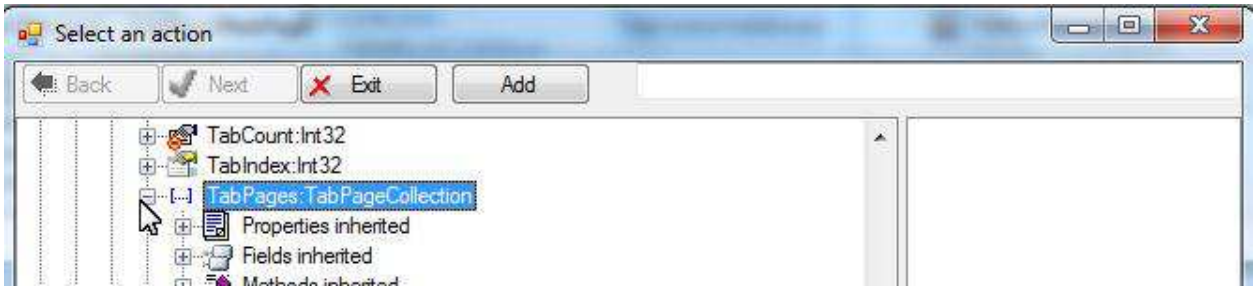
An action appears which creates a new instance of TabPageWebBrowser. An icon for the new variable also appears:



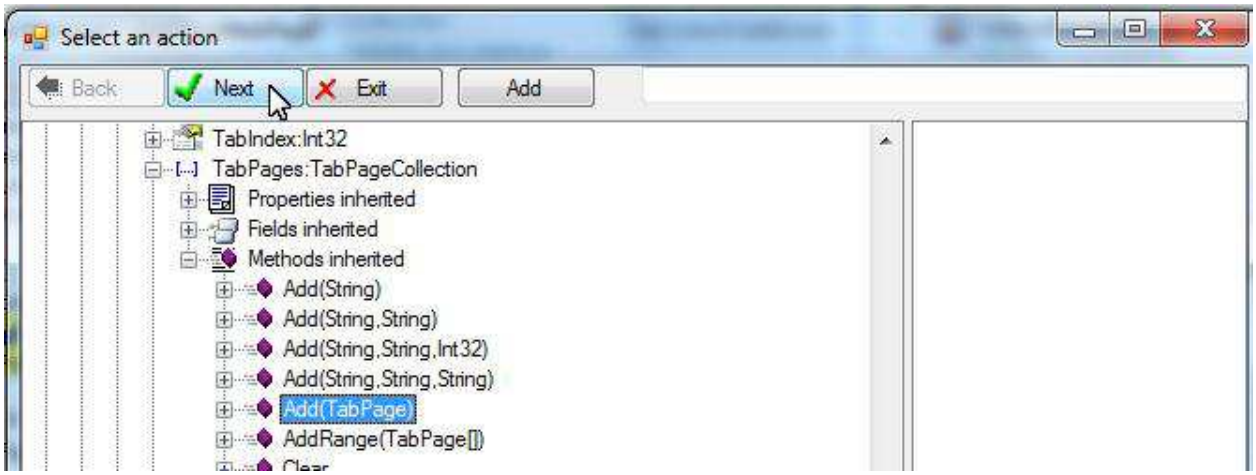
Create an action to add the new TabPageWebBrowser to the TabPages collection:



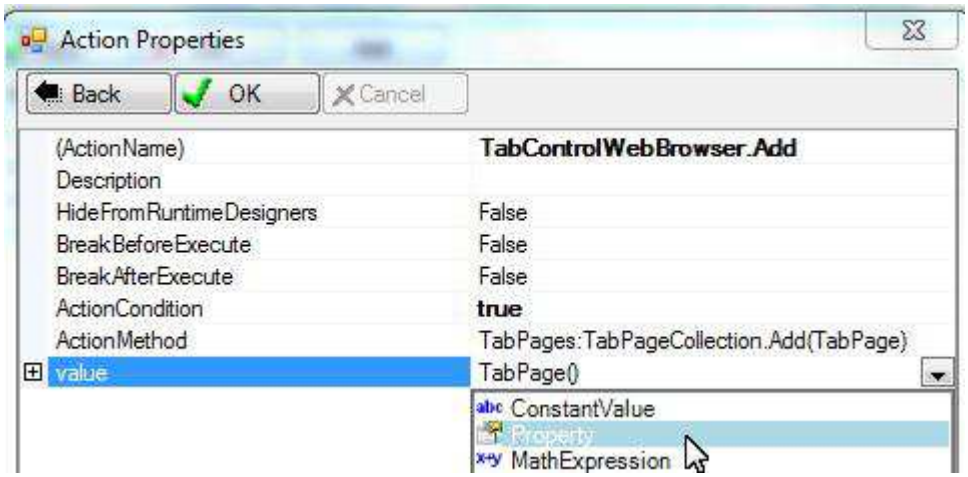
Locate the TabPages collection:

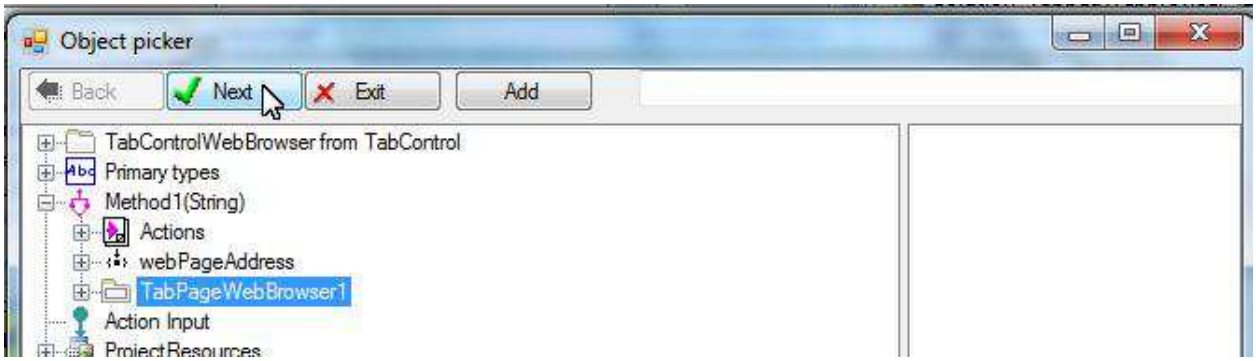


Select its Add method:

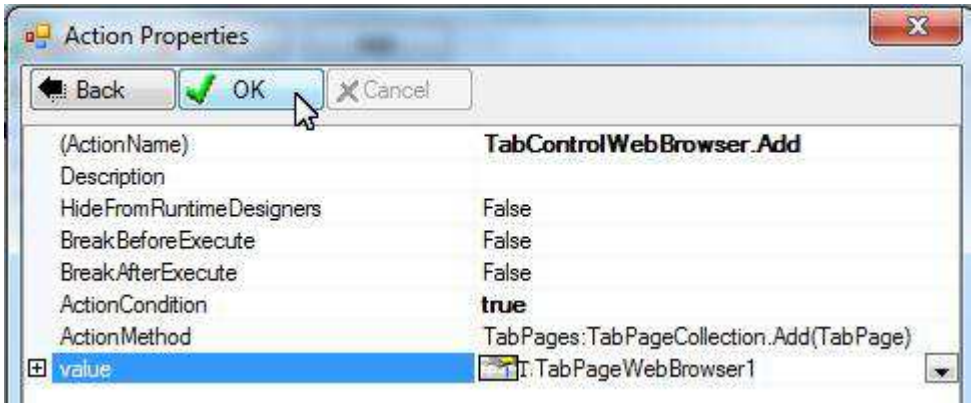


Select the variable TabPageWebBrowser1:



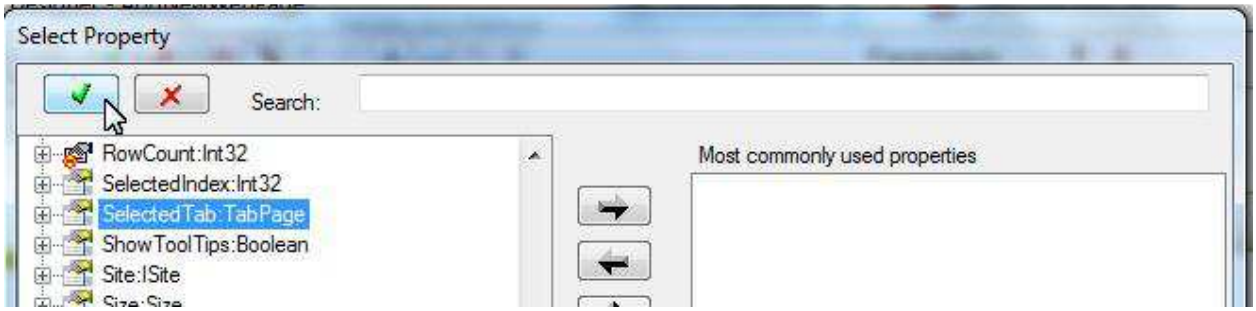


Click OK:

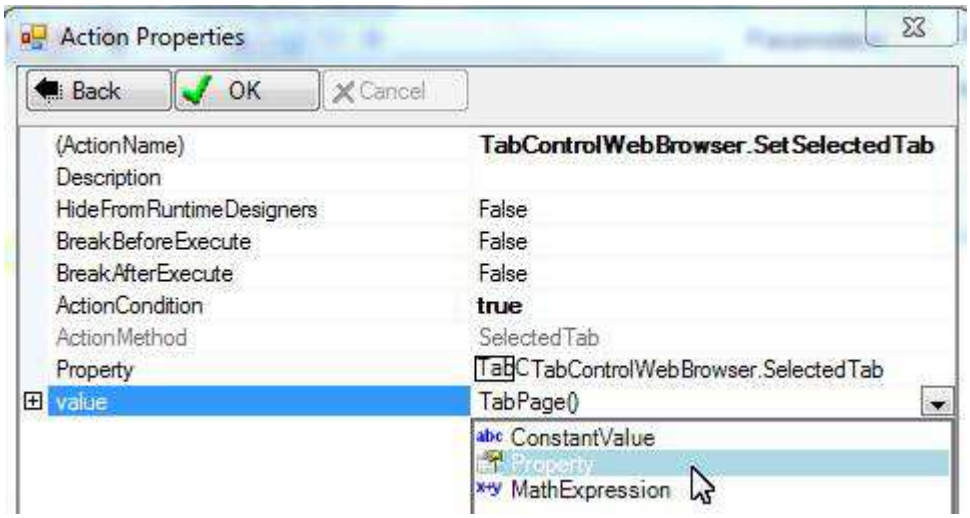


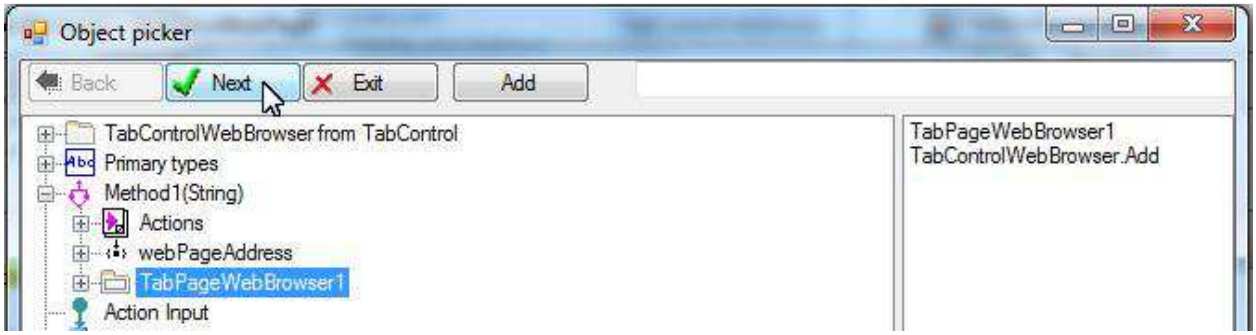
The action appears. Link it to the previous action.

Create another action to set the SelectedTab to the new tab page:

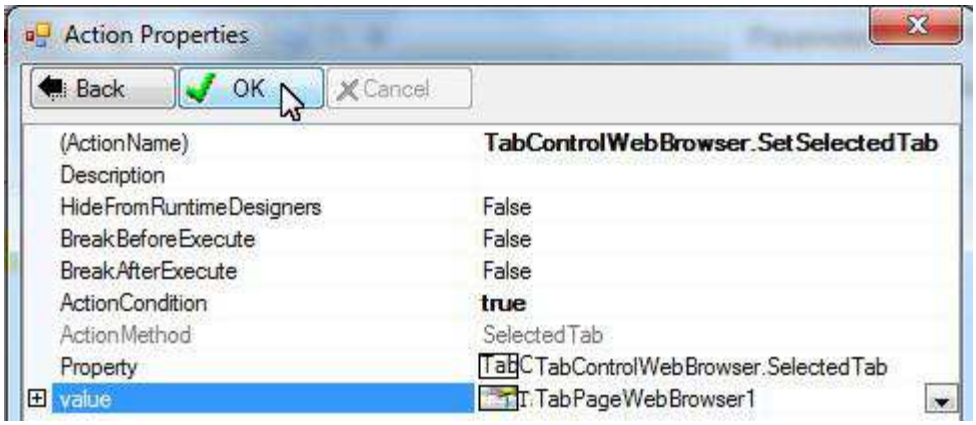


Select the variable TabPageWebBrowser1:



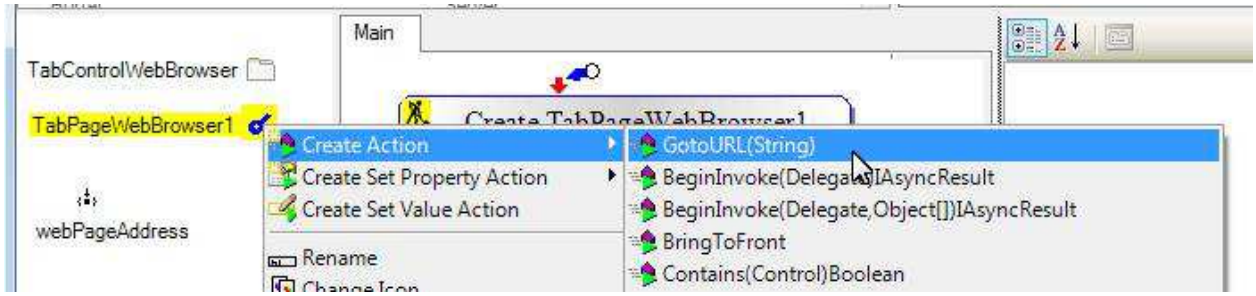


Click OK:

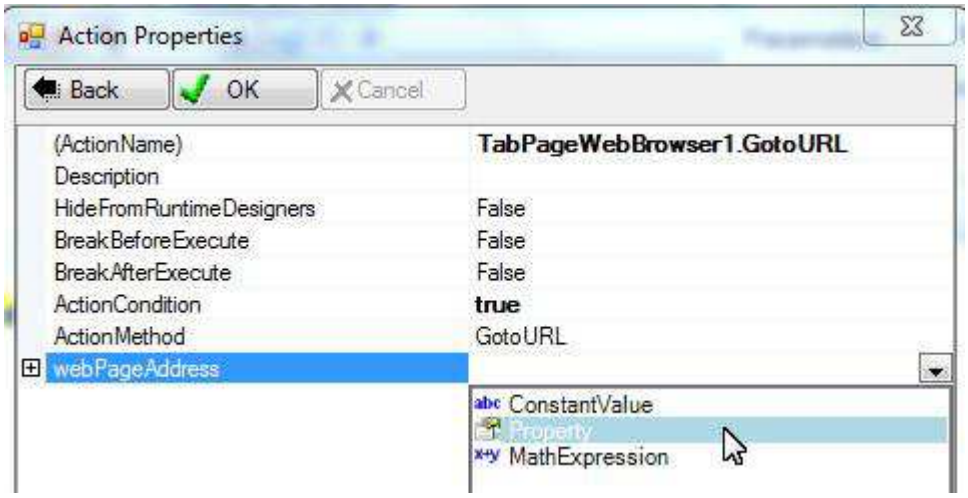


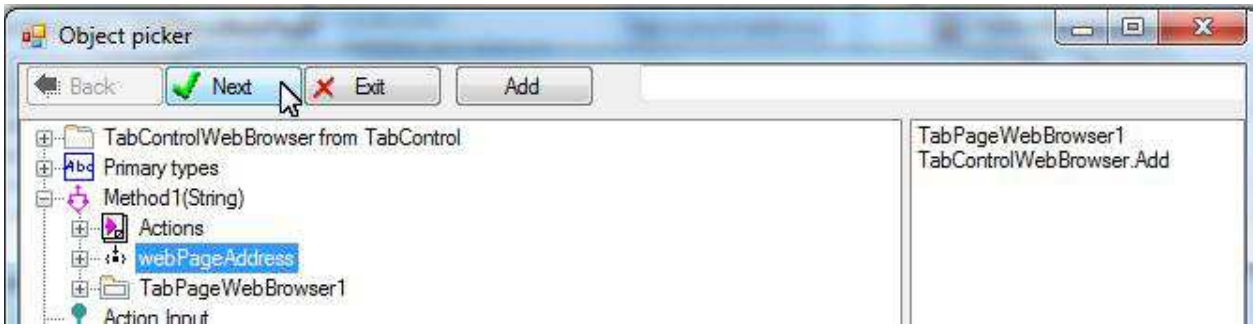
The action appears. Link it to the last action.

Create an action to execute GotoURL method of TabPageWebBrowser1:

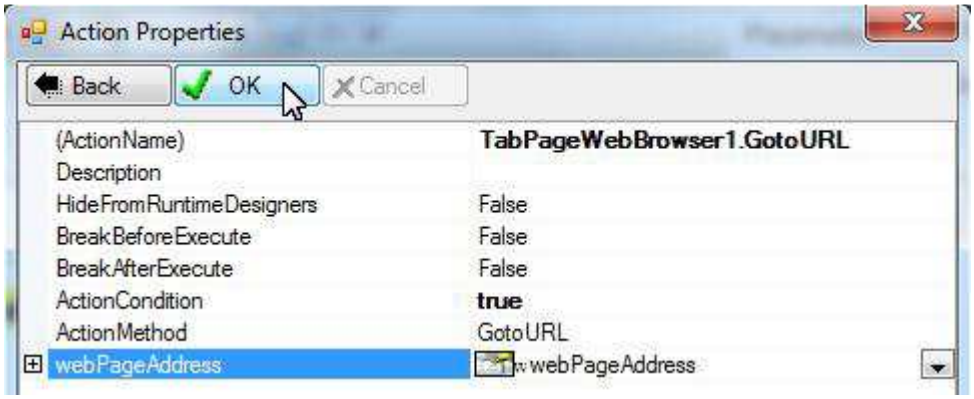


Use webPageAddress for the action:



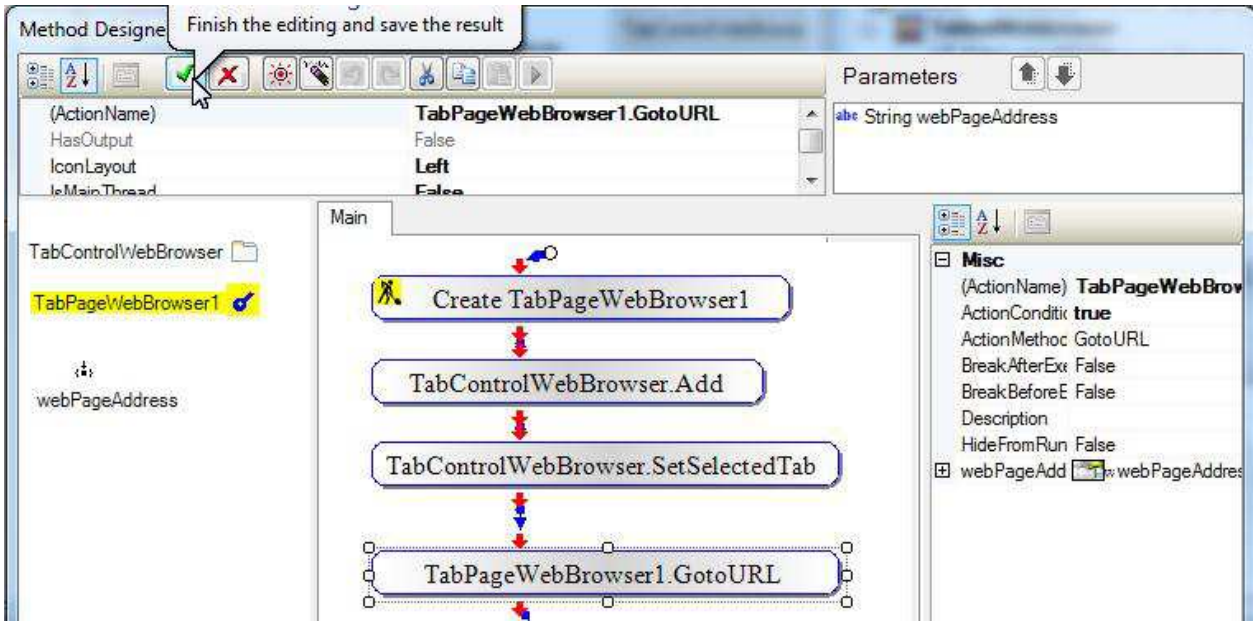


Click OK:



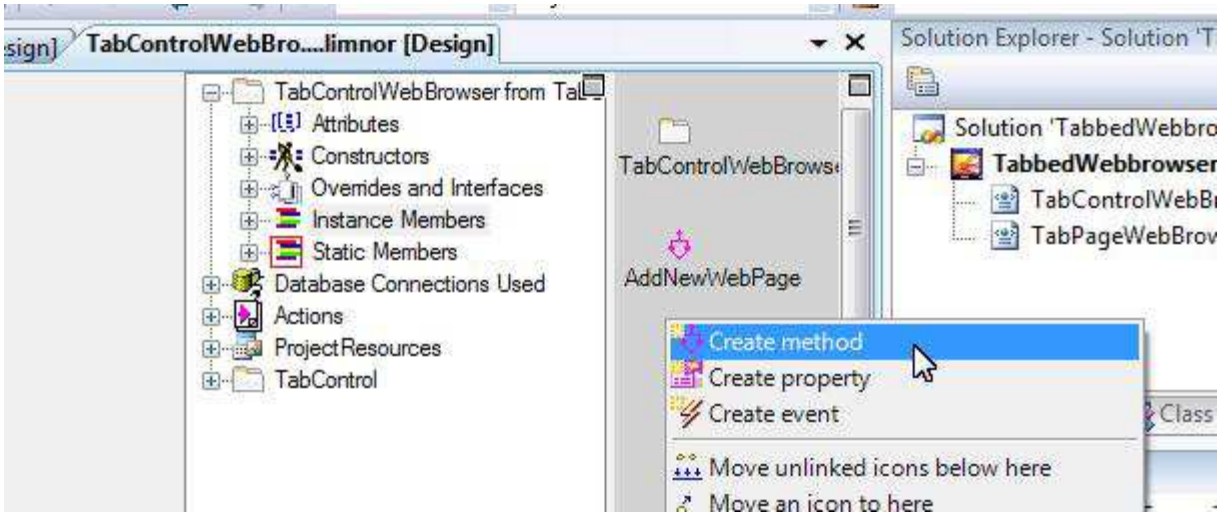
The action appears. Link it to the last action.

That is all for this method:

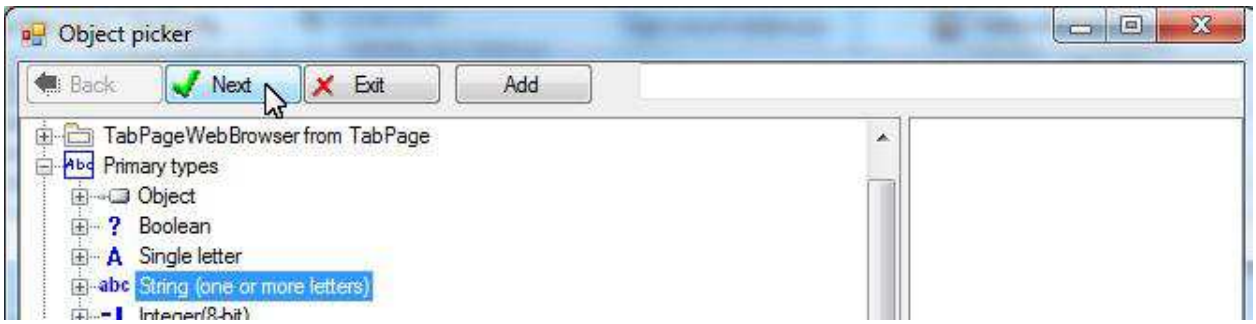


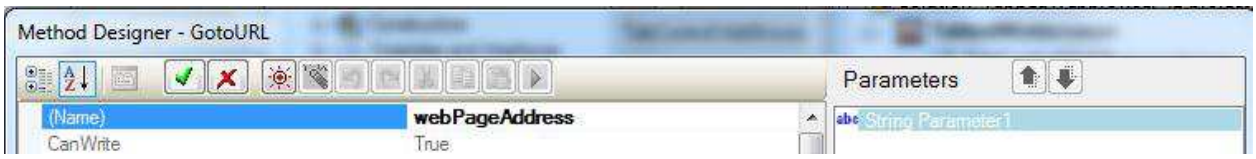
Create TabControlWebBrowser method – GotoURL

This method executes GotoURL method of the selected tab page. If there is not a selected tab page then it executes AddNewWebPage instead. This method is supposed to be used by the programs using this DLL.

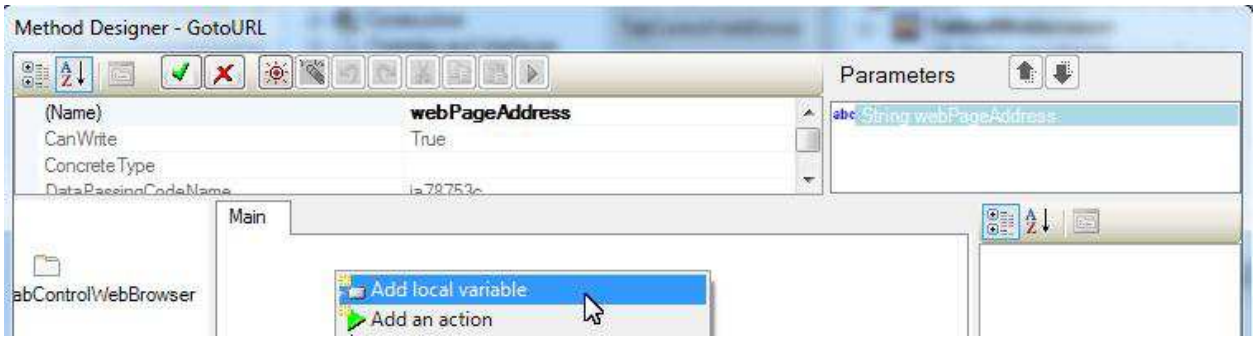


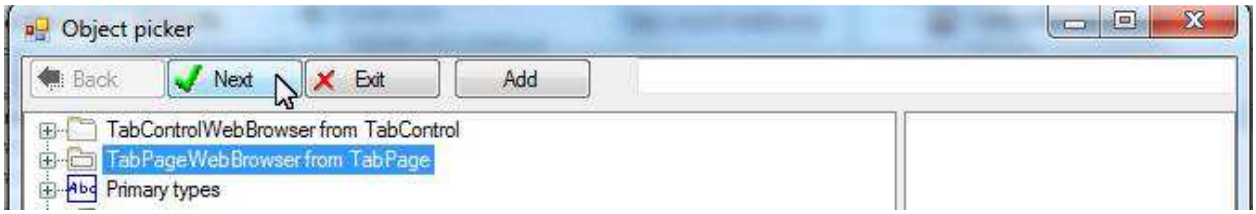
Rename it to GotoURL. Add a String parameter:

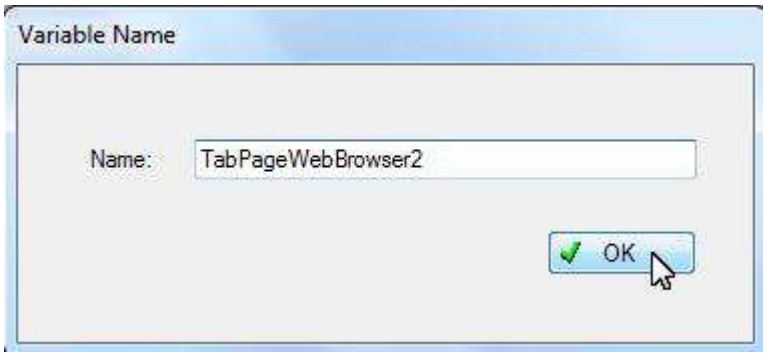




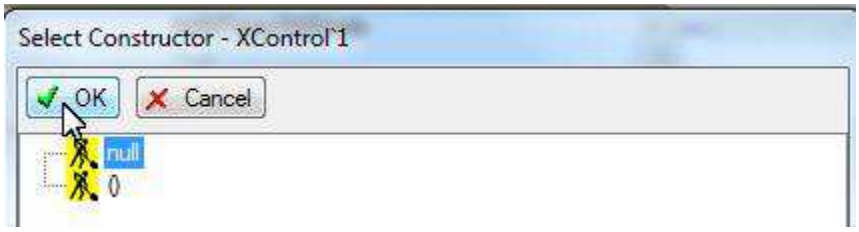
Create a TabPageWebBrowser variable:



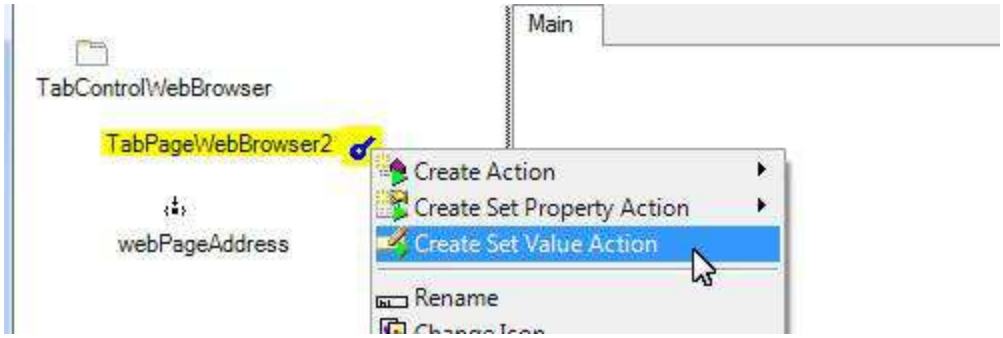




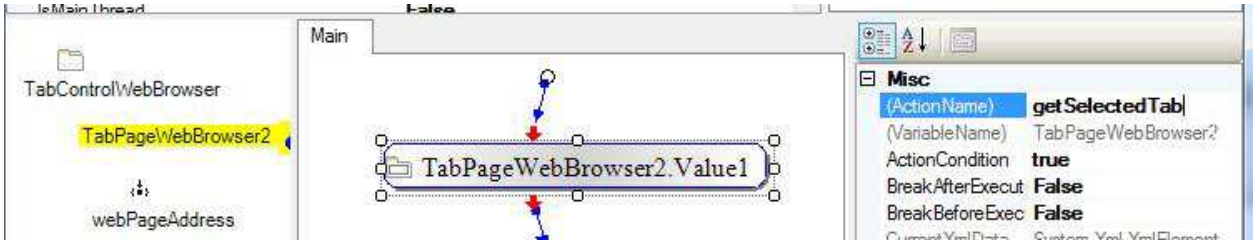
Select “null” because we do not want to create a new instance:



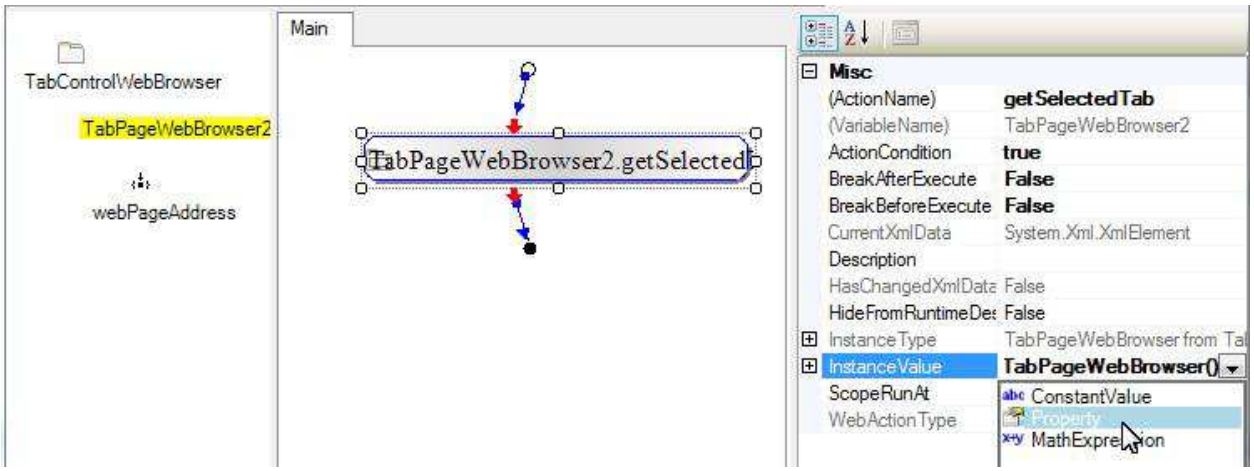
The variable appears. Set it to the SelectedTab:

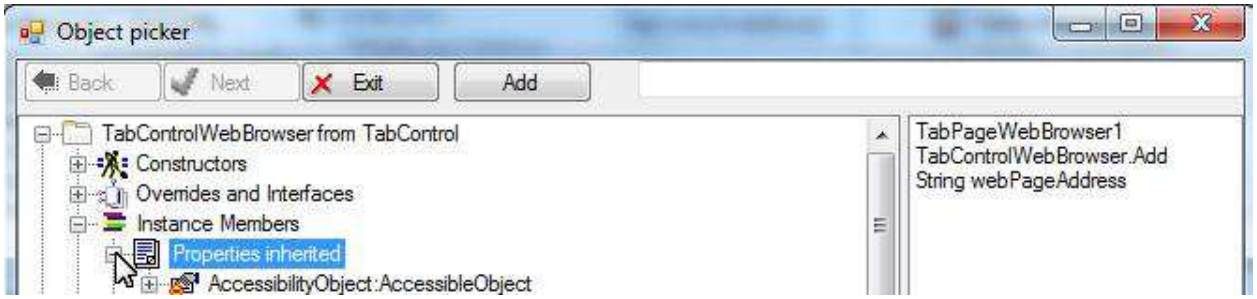


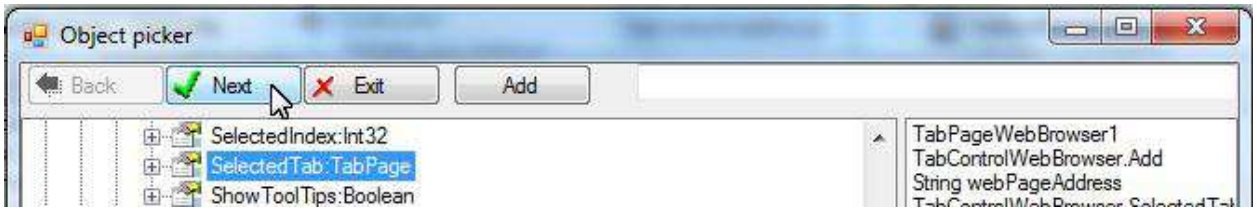
Name the action “getSelectedTab”:



Set its value to SelectedTab:

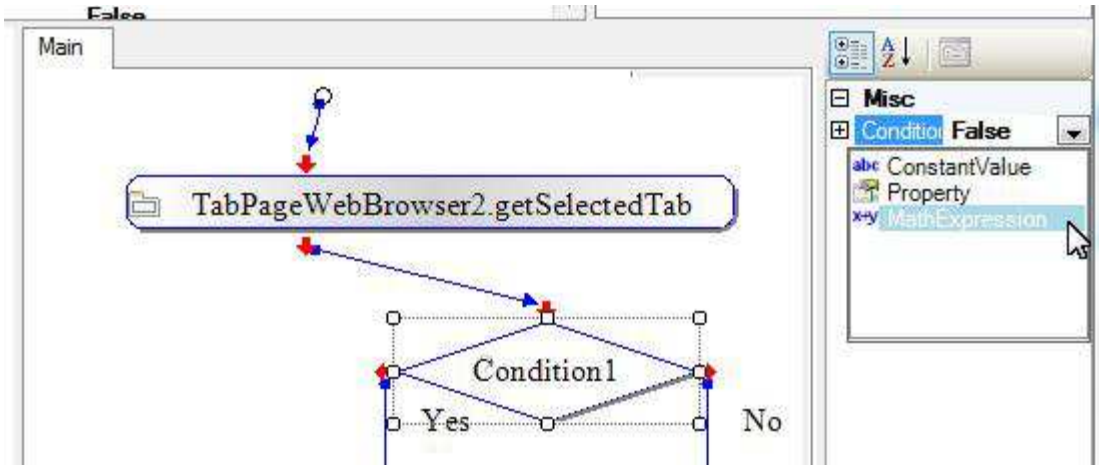




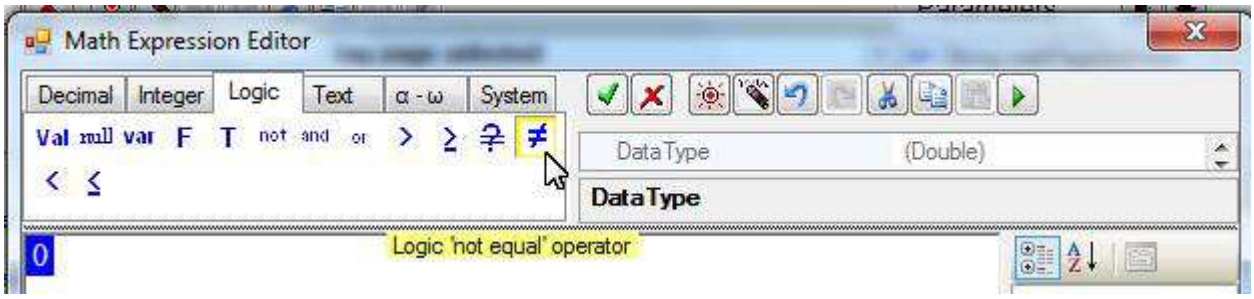


Add a condition action to check to if the variable is null:

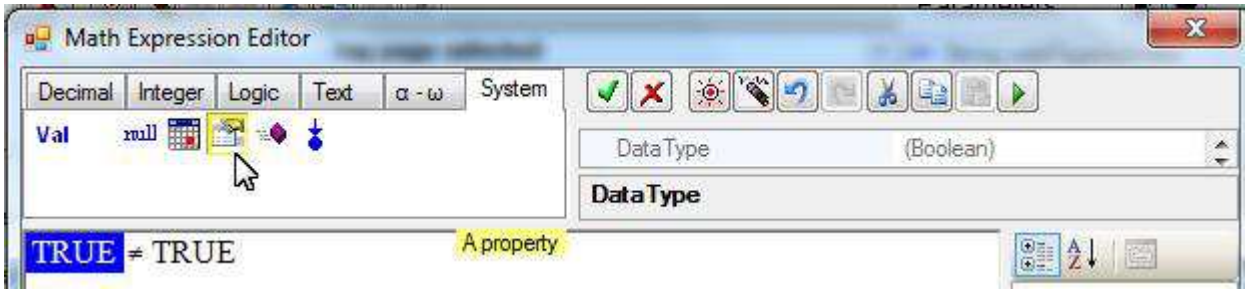
Link the action to the previous action. Set its Condition:

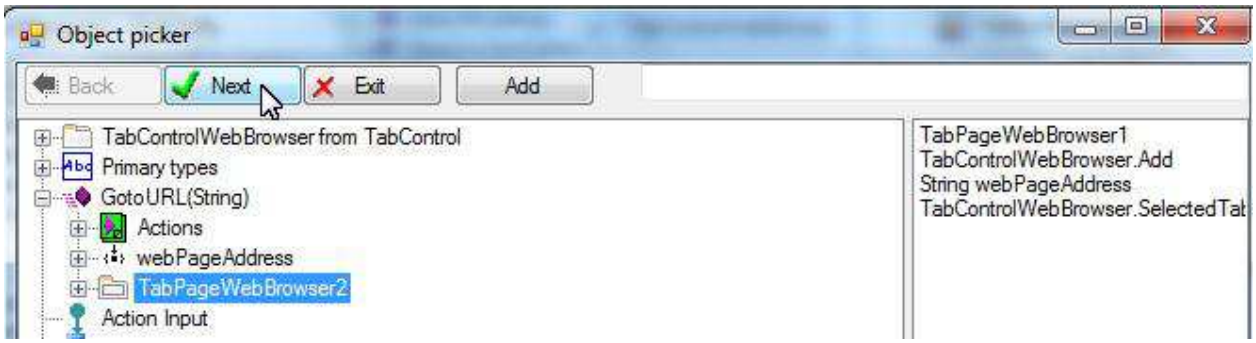


Select “not equality” icon:

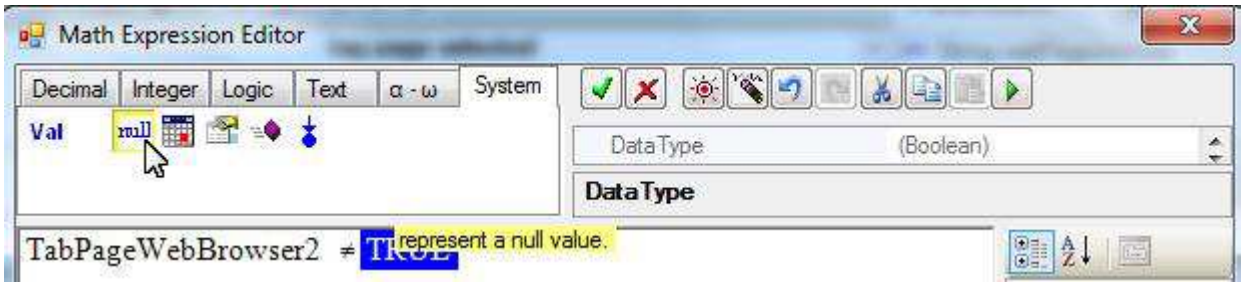


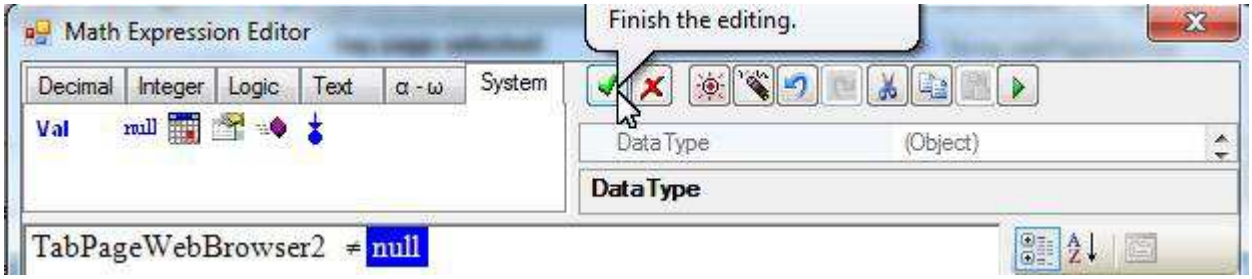
Select variable TabPageWebBrowser2 into the first item:



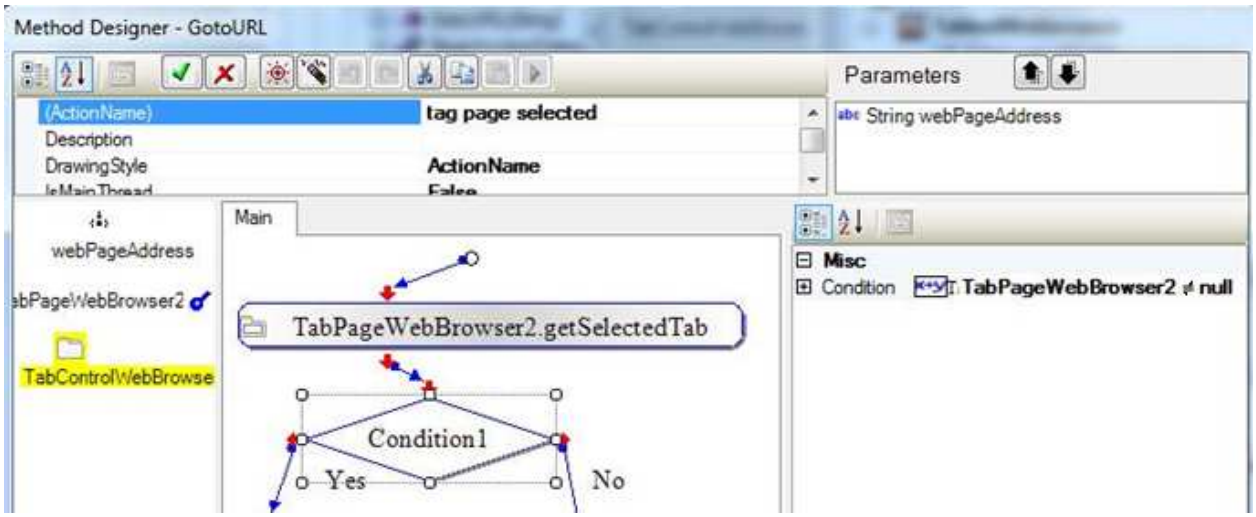


Select “null” into the second item:

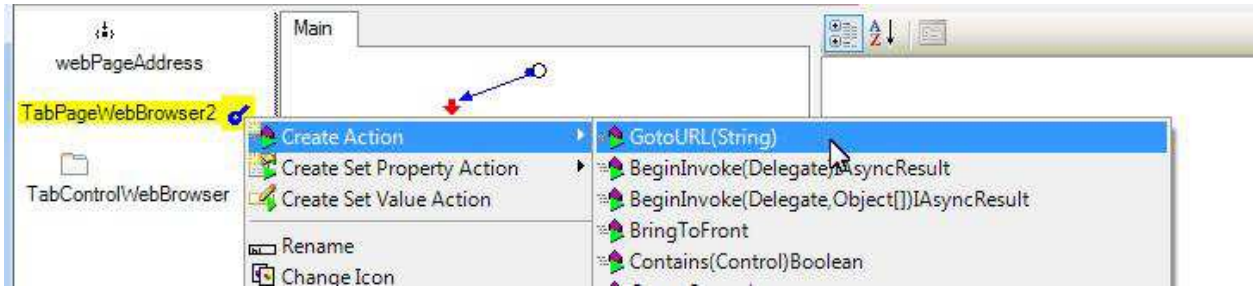




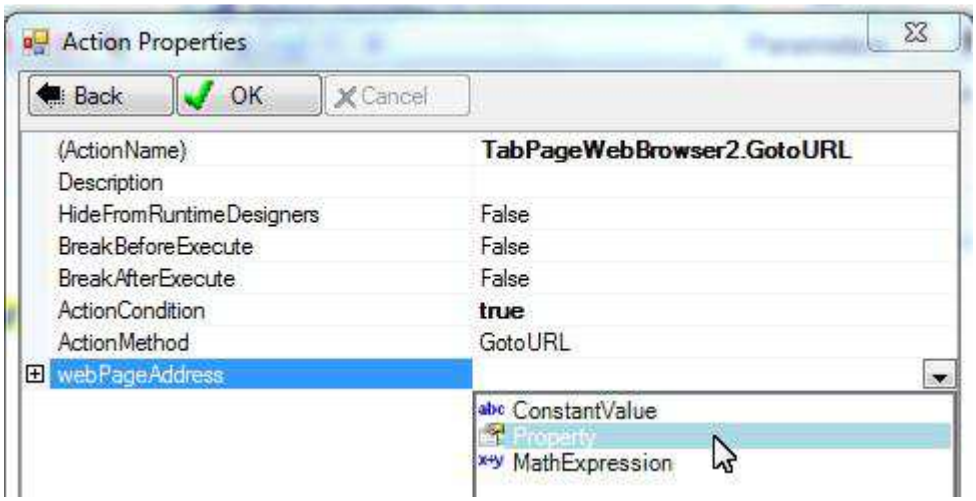
Name the condition “tab page selected”:

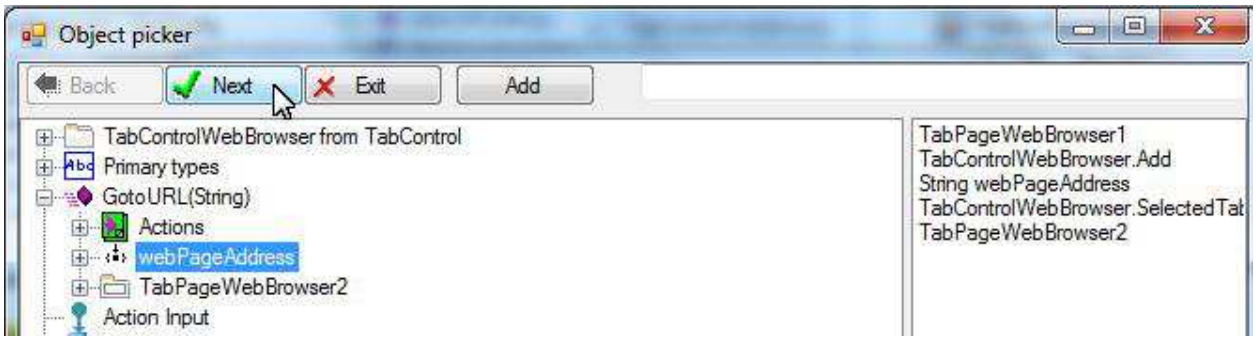


Create an action to execute GotoURL on the selected tab page and link it to “Yes”:

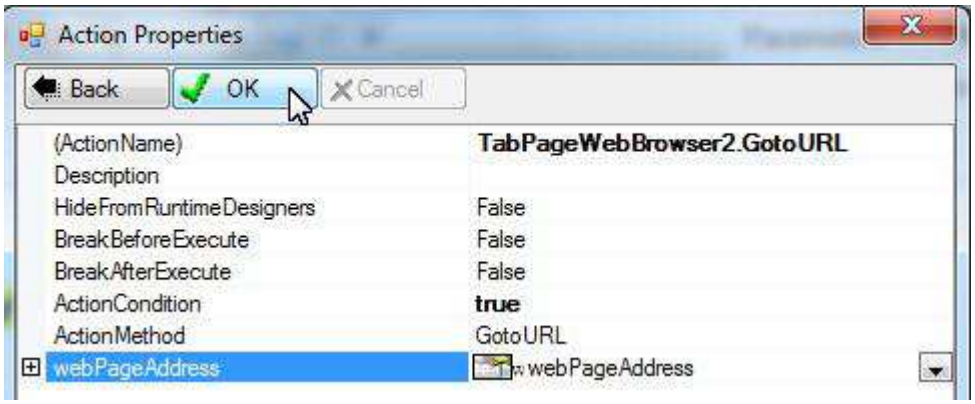


Use method parameter for the action:

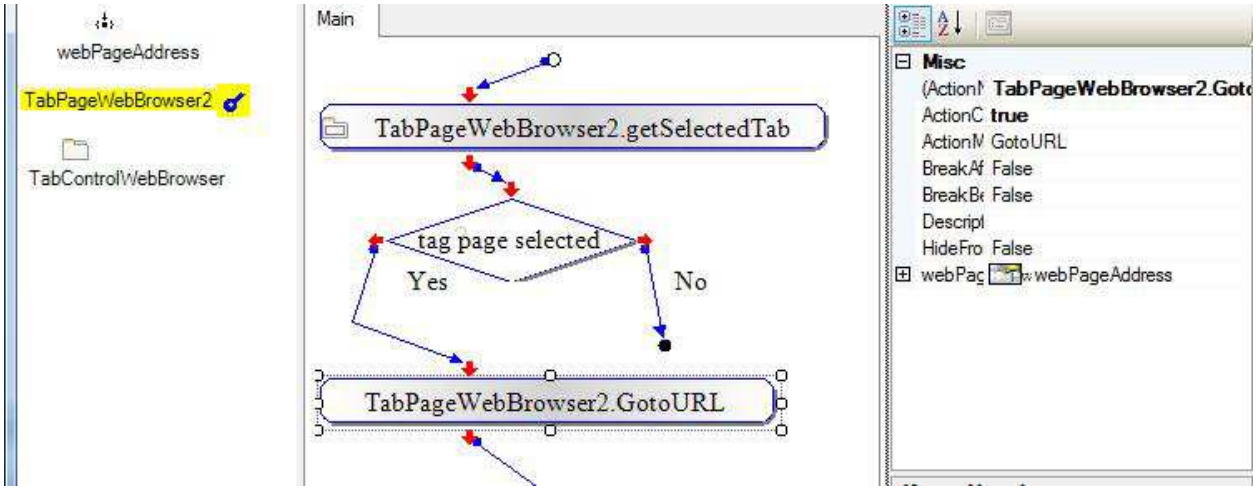




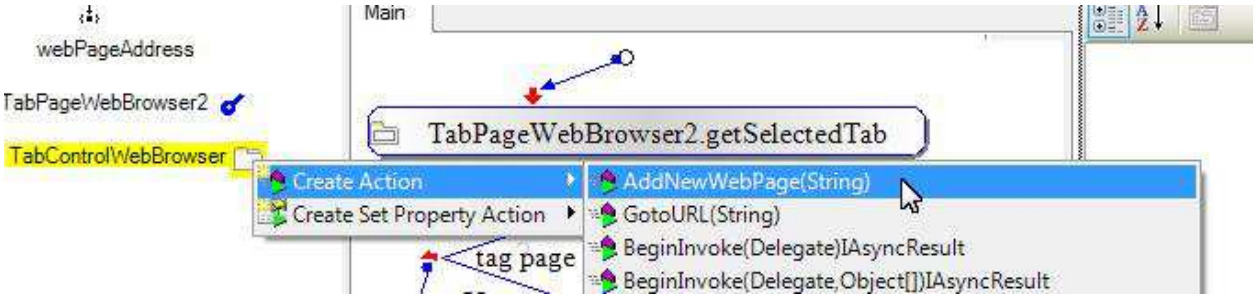
Click OK:



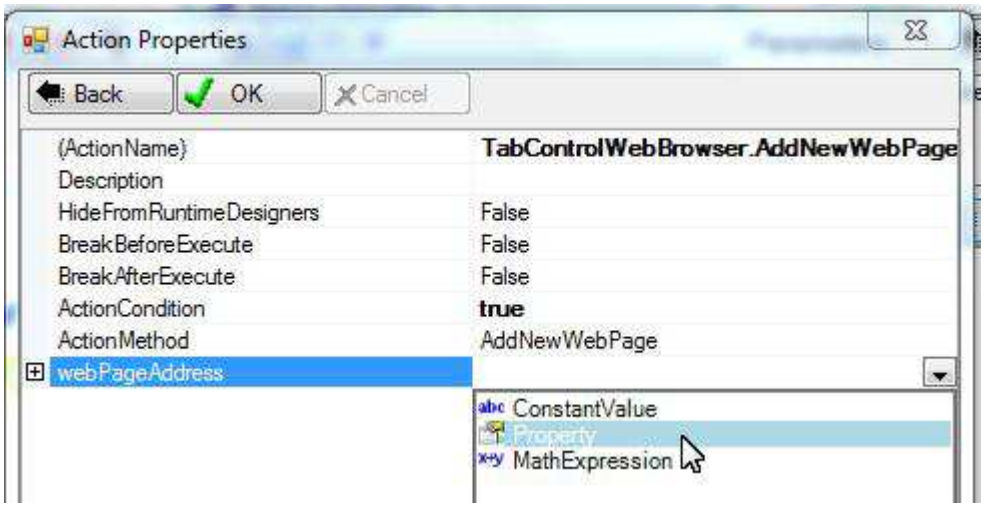
Link it the “Yes”:

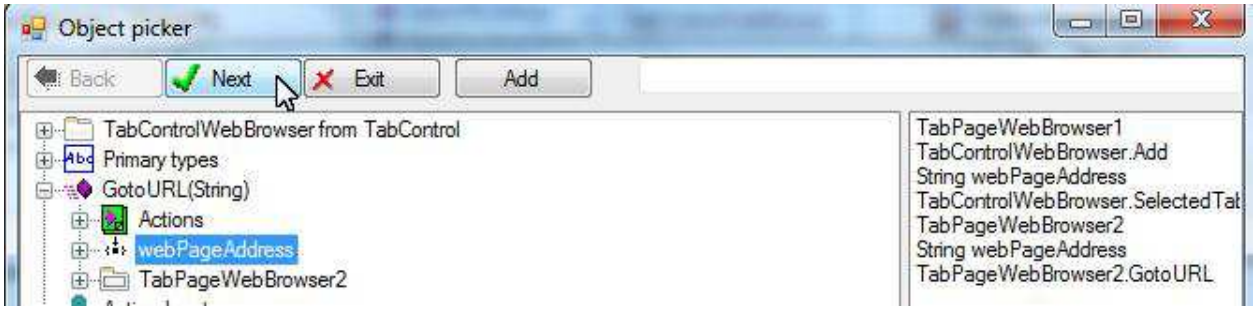


Create an AddNewWebPage action:

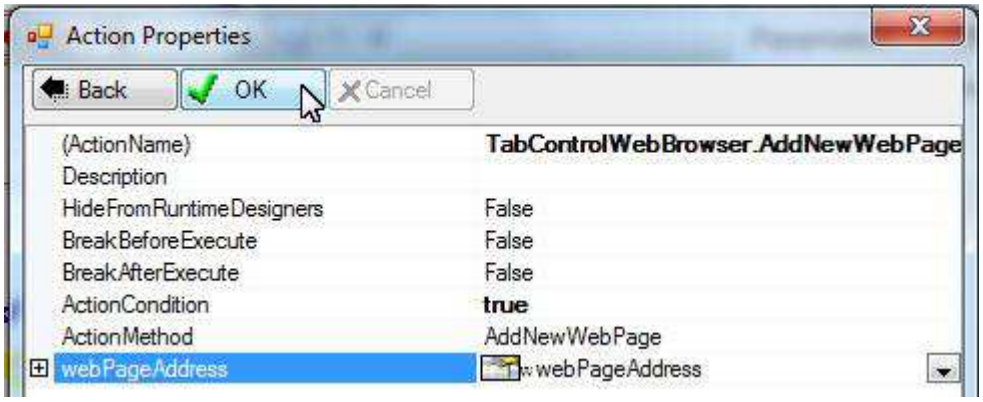


Use the webPageAddress for the action:

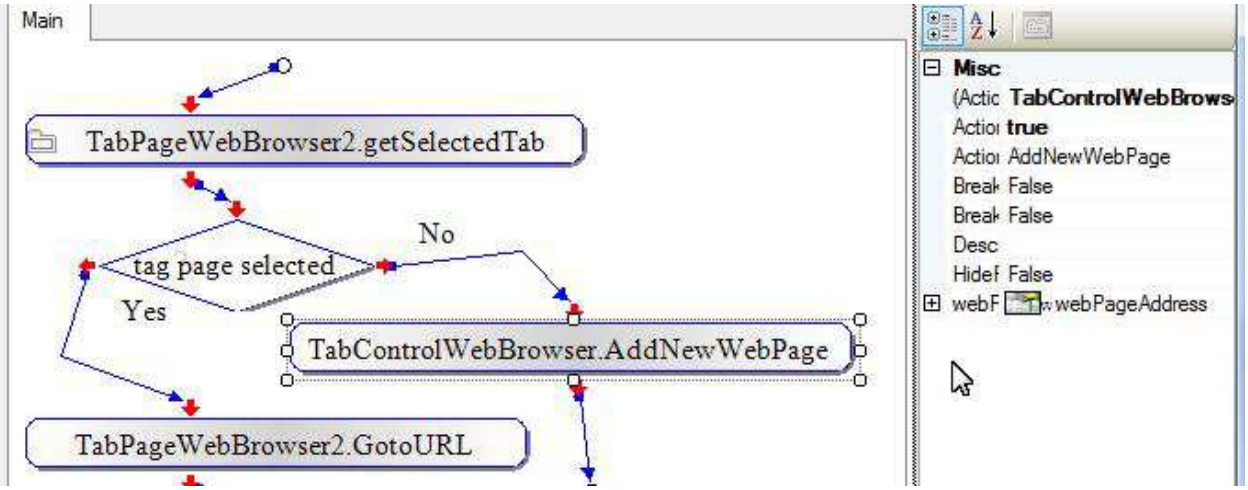




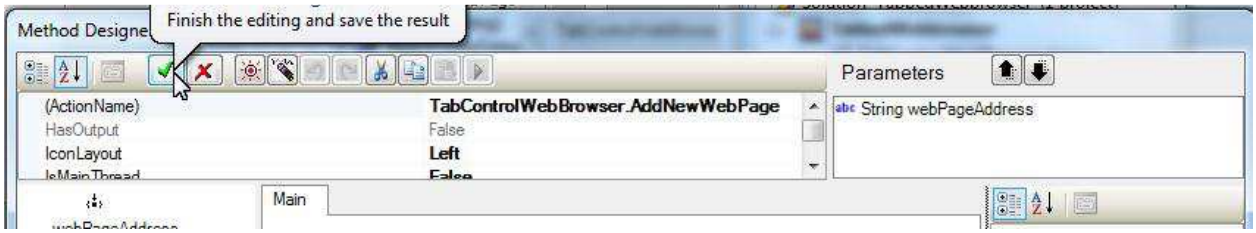
Click OK:



Link it to “No”:



That is all for this method:

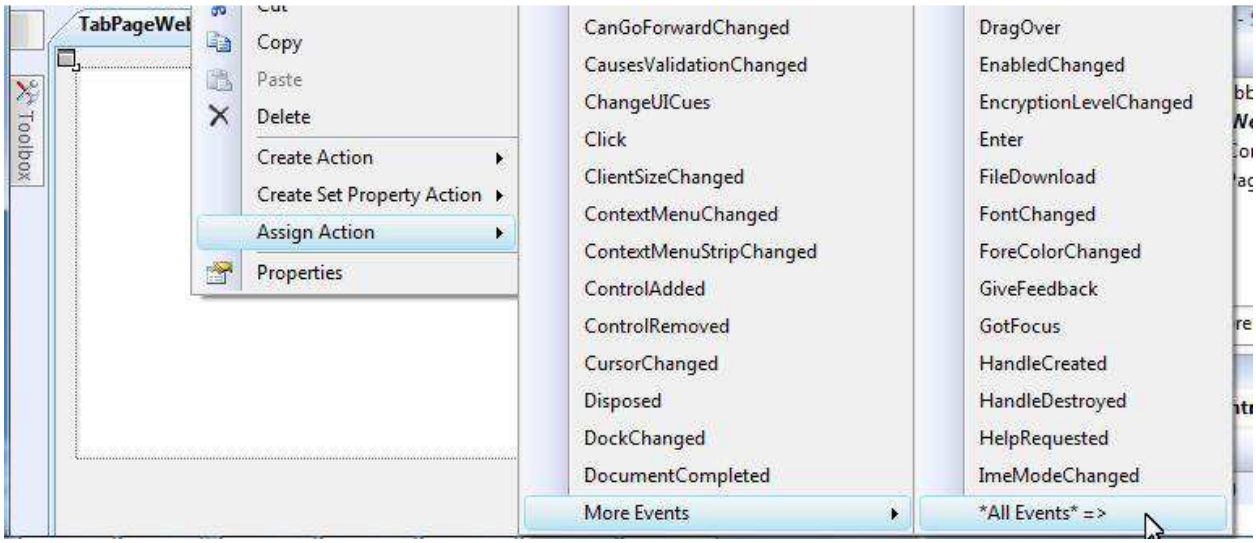


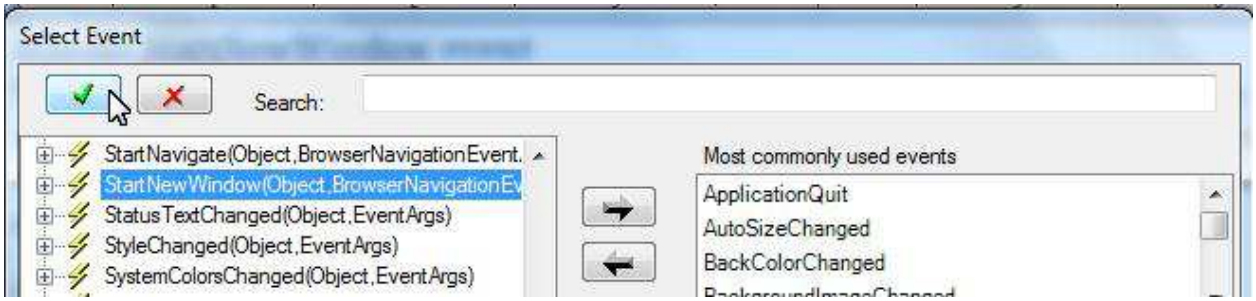
Handle StartNewWindow event

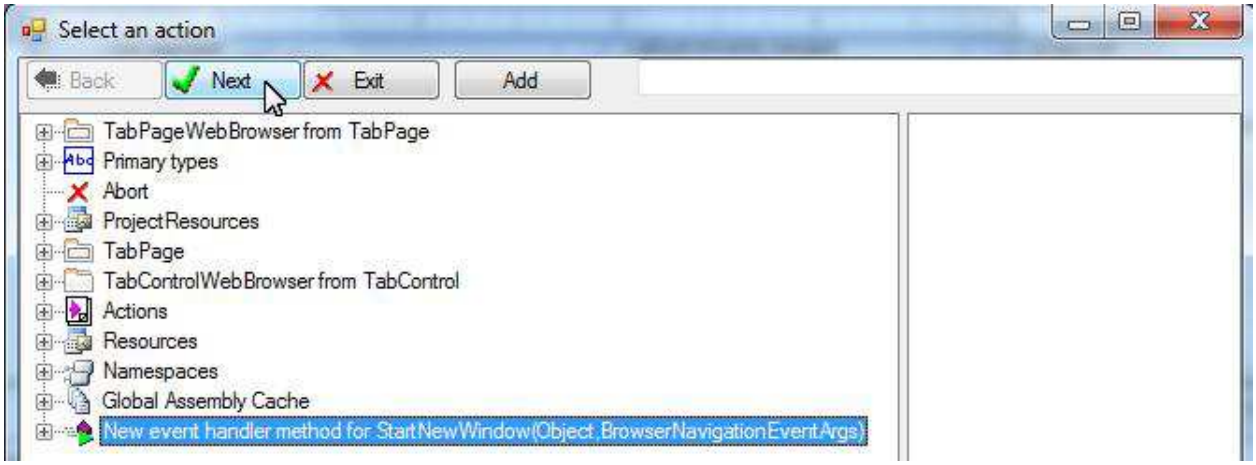
The key point of this sample is to handle StartNewWindow event. The event occurs before a popup window is launched. For our sample, we want to cancel the event so that the web browser control will not open the default web browser. We want to execute an AddNewWebPage action to show the web page on a new tab page.

Create handler method

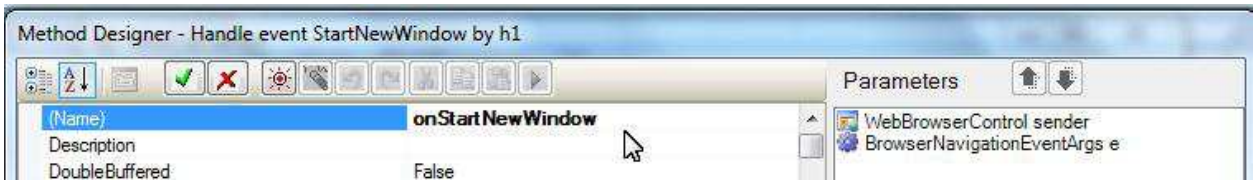
Create an event handler method for event StartNewWindow:





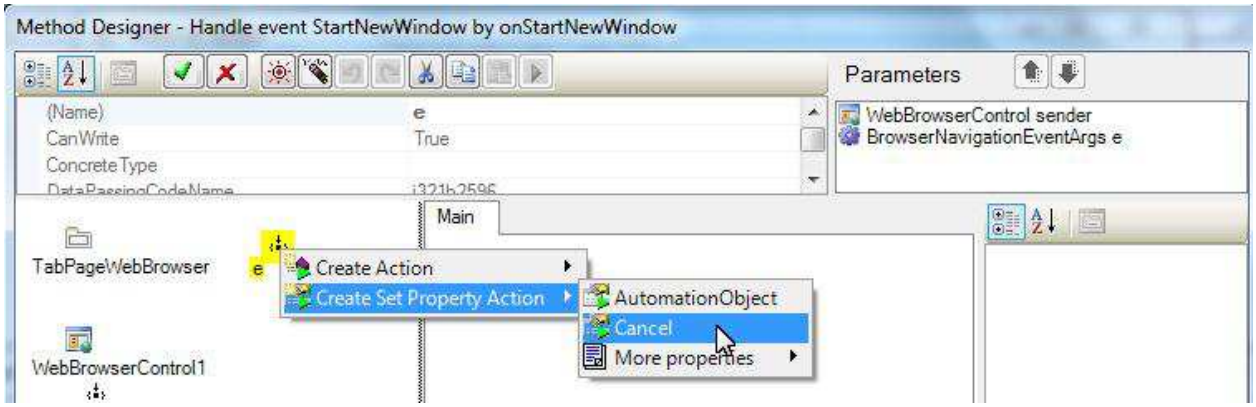


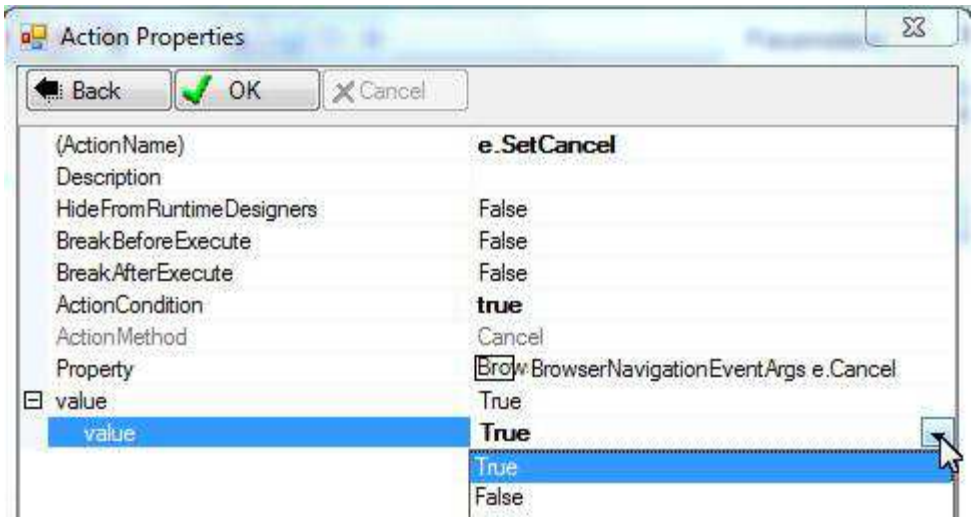
Rename the handler method to “onStartNewWindow”:



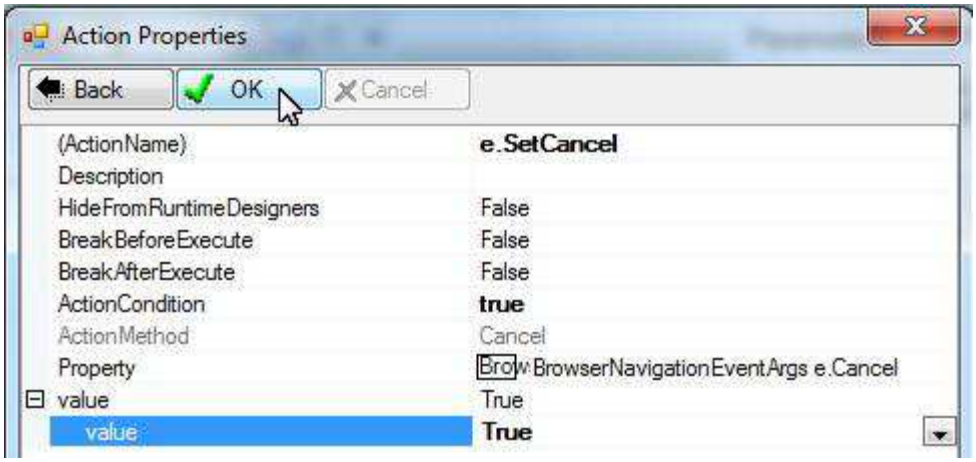
Cancel the event

We want to cancel the event so that the default web browser will not be launched.

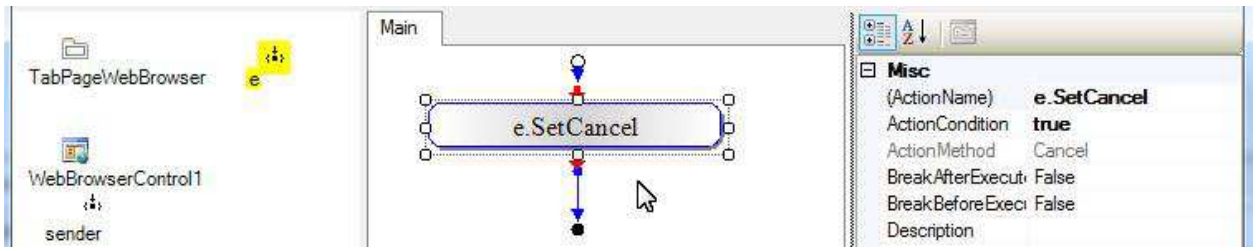




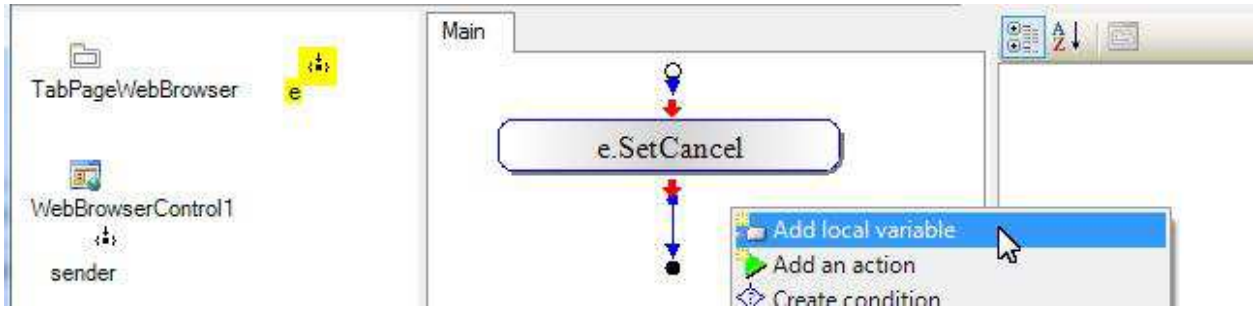
Click OK:

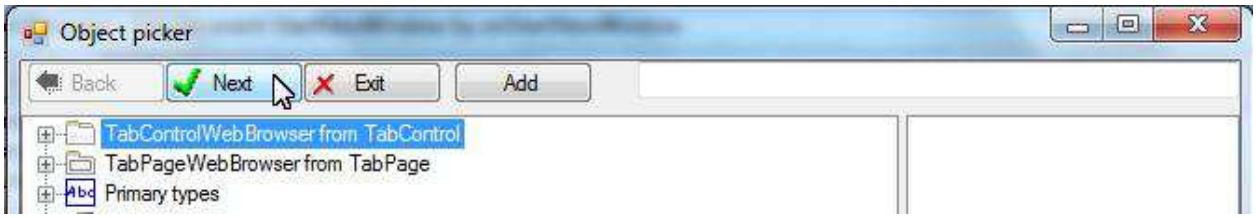


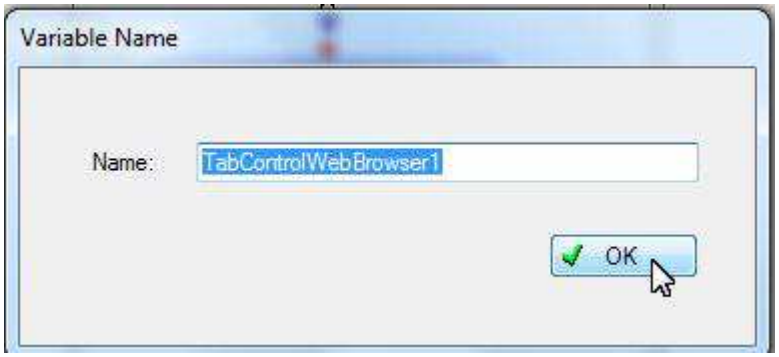
The action appears:



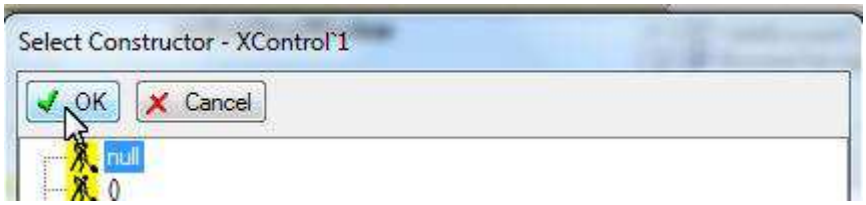
We use a variable of type TabControlWebBrowser to get the parent of the current tab page:



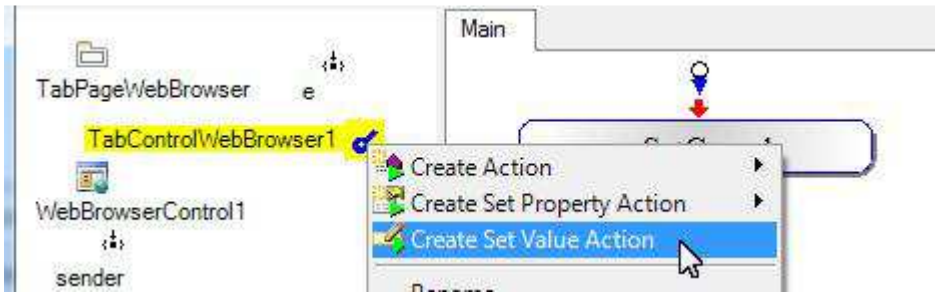




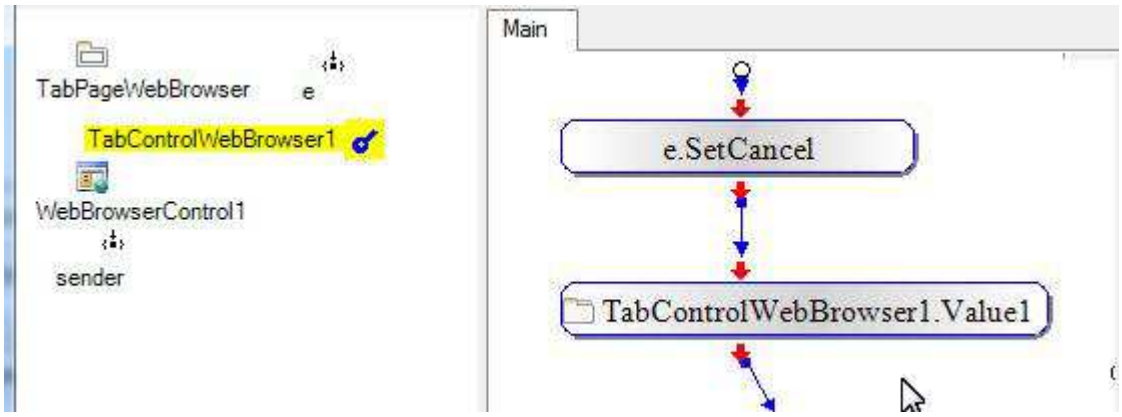
Select “null” because we do not want to create a new instance:



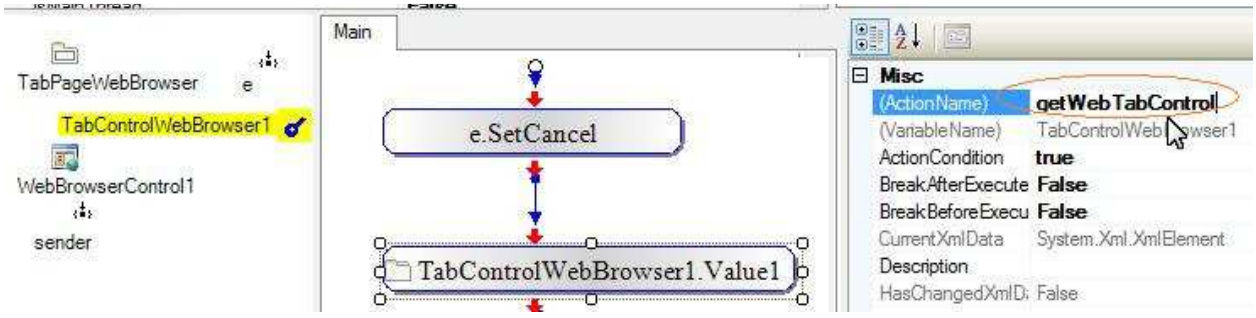
The variable appears. Create an action to set the variable to the Parent:



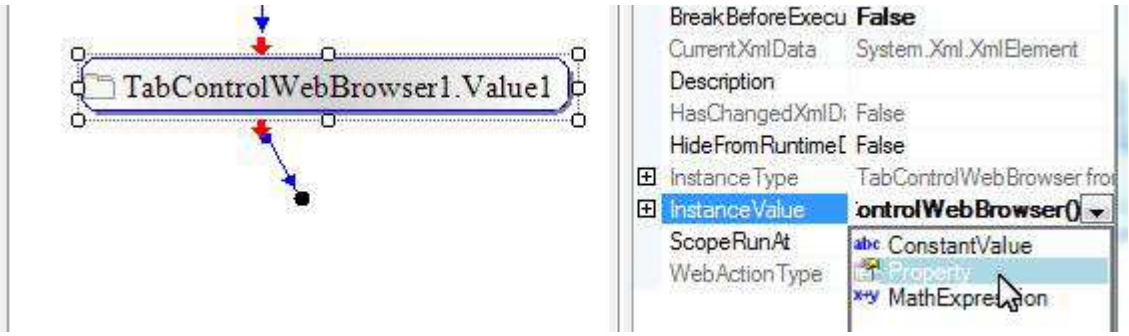
The action appears. Link it to the last action:

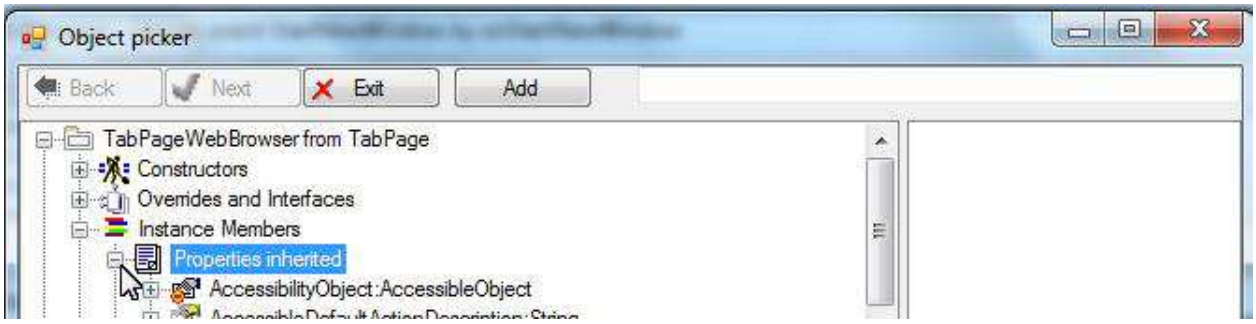


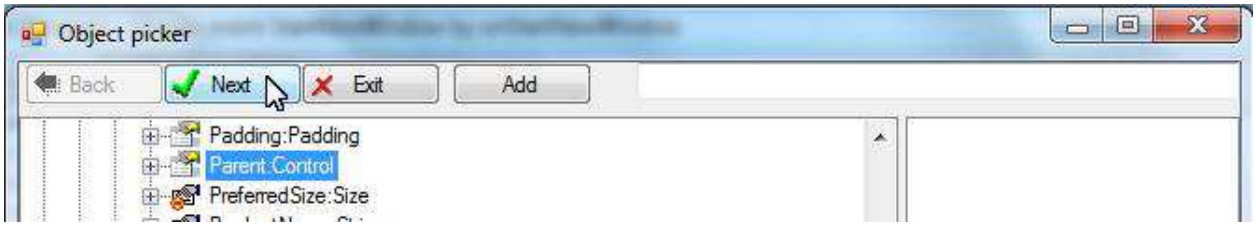
Rename the action to “getWebTabControl”:



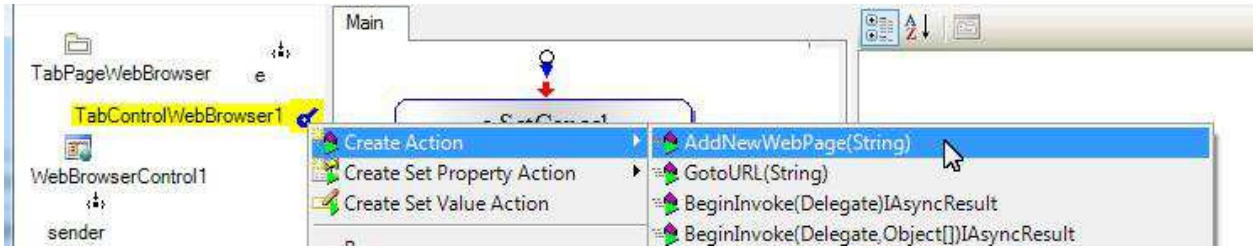
Assign Parent property to the “InstanceValue” of the action:



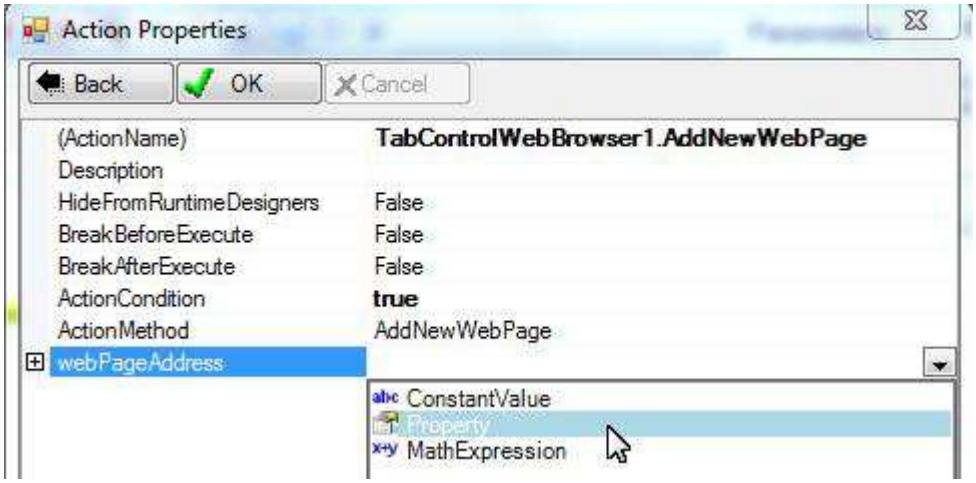


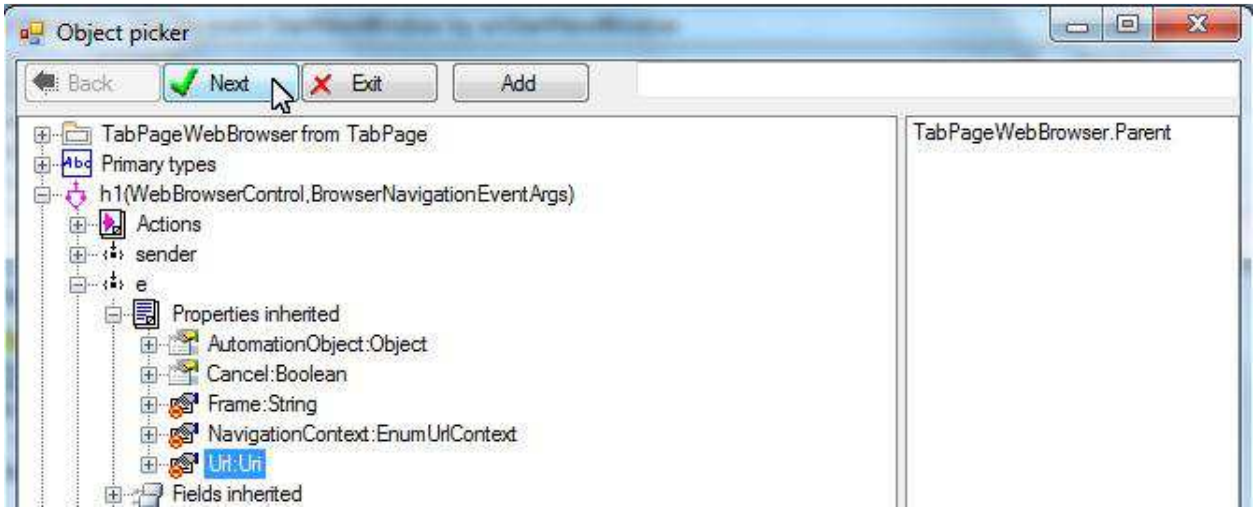


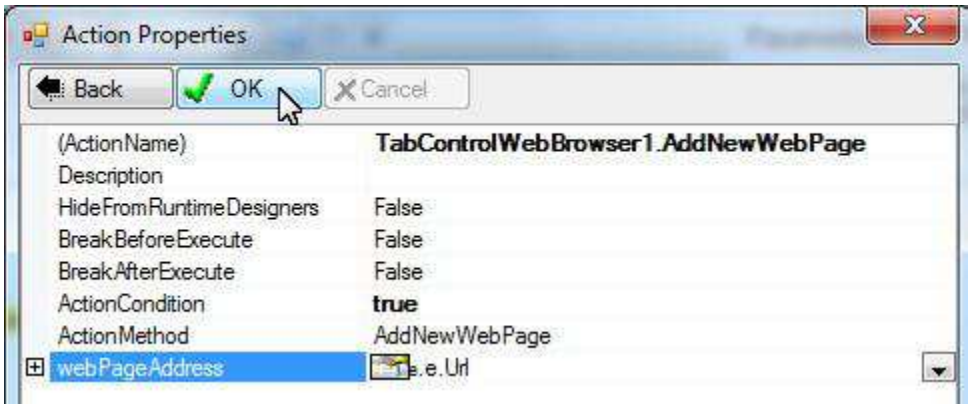
Create an AddNewWebPage action:



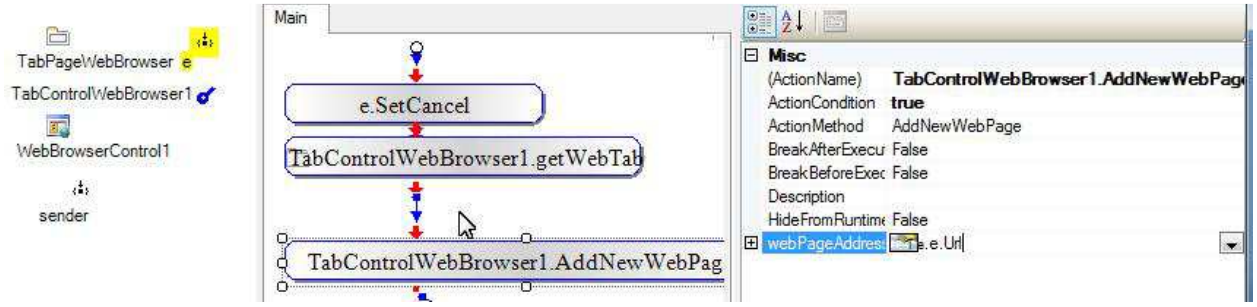
Pass the URL for the popup window to the action:







Link it to the last action:



That is all for the method:

Provide Close button

We need to allow the user to remove a tab page. One convenient UI design is to add a “Close” button beside the tab page caption, like IE does:

One problem is that a tab control provided by the Microsoft .Net Framework does not allow adding button control to it.

One solution is to take over the drawing of the tab caption and draw a button on the right side of the tab caption. We also handle the MouseDown event of the tab control and check whether the mouse is

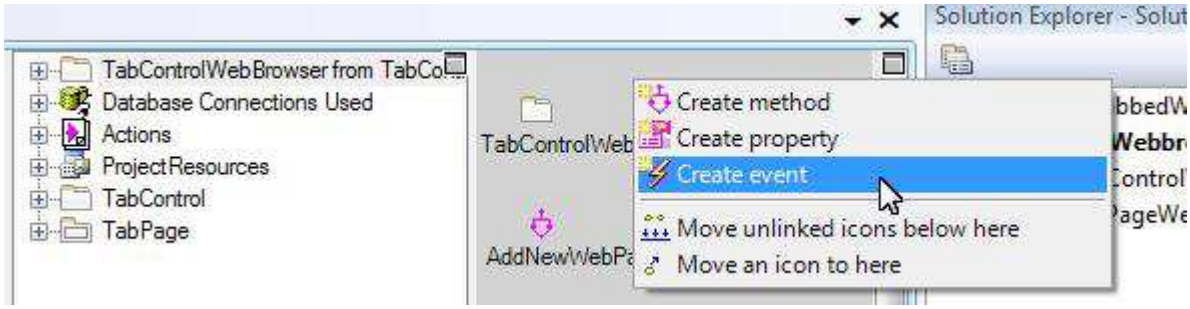
in the area where we draw the “close” button, and remove the tab page if the mouse pointer is inside the “close” button area.

In summary:

1. Create a BeforeRemoveTabPage event so that it can be programmed to allow the user to cancel removing tab page.
2. Take over tab caption drawing
3. Draw tab caption and a “close” button beside the caption
4. Handle mouse down event to check if the mouse pointer is within the area of the “close” button
5. Fire BeforeRemoveTabPage event if the mouse pointer is inside the area of the “close” button.
6. Remove the tab page if the user does not cancel it.

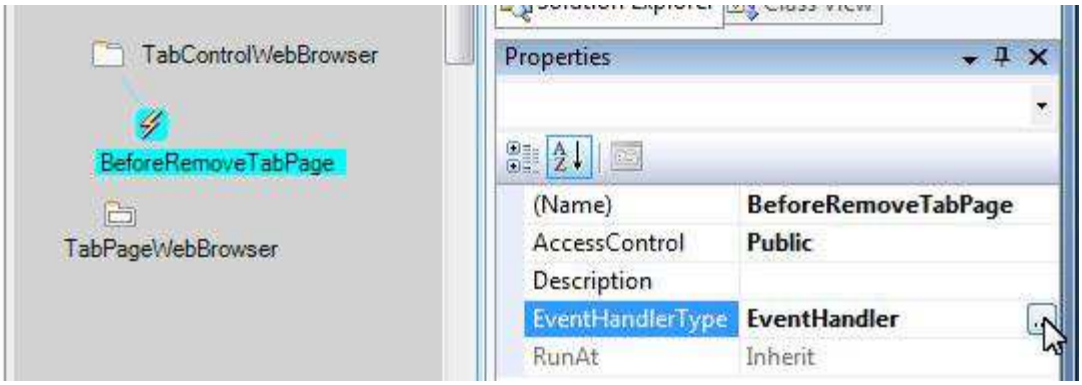
Create BeforeRemoveTabPage event

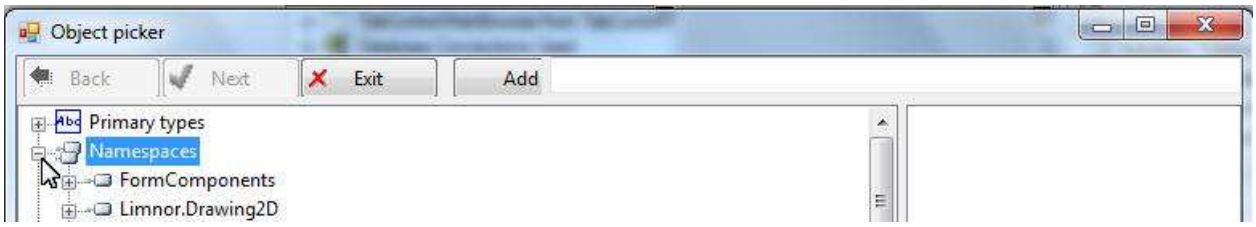
Create a new event on TabControlWebBrowser:

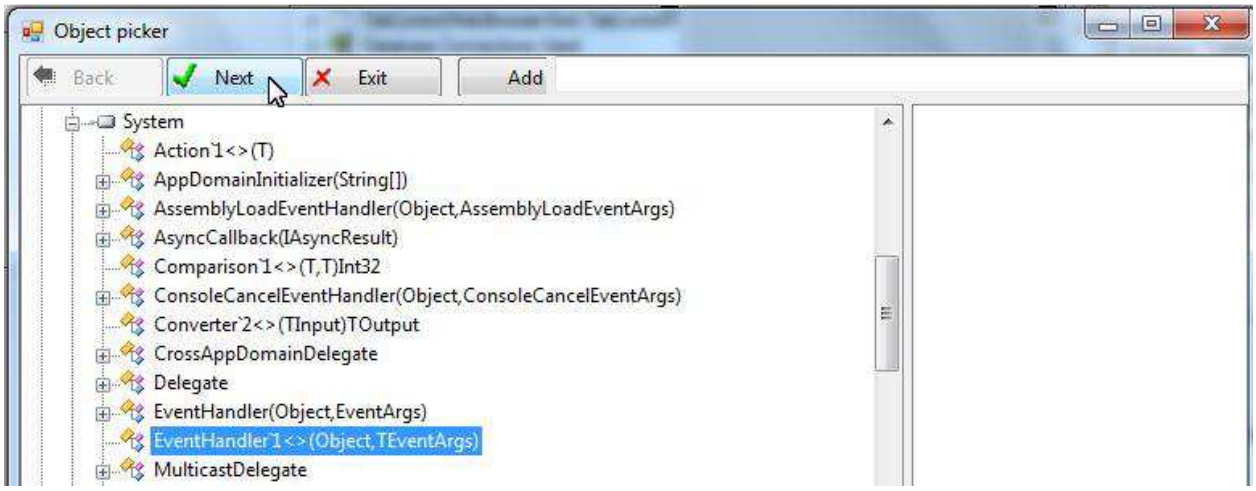


A new event is created. Rename it to BeforeRemoveTabPage:

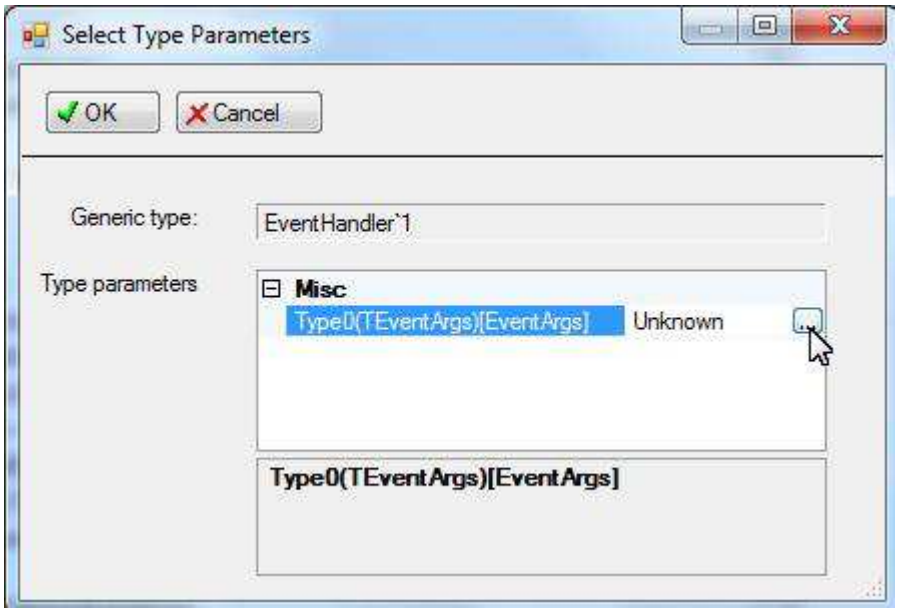
Change the event handler type to provide “Cancel” ability to the user:

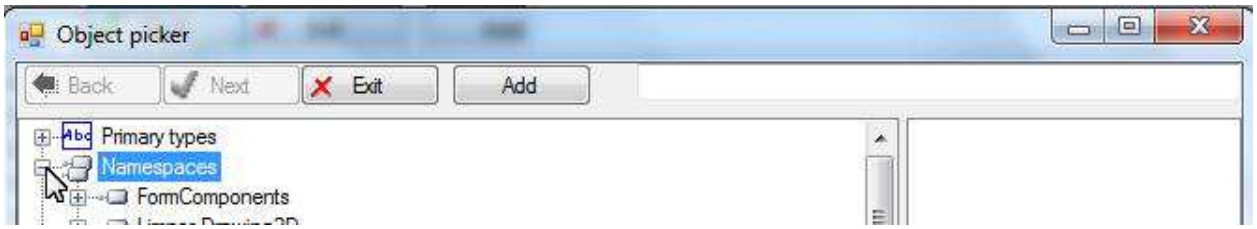


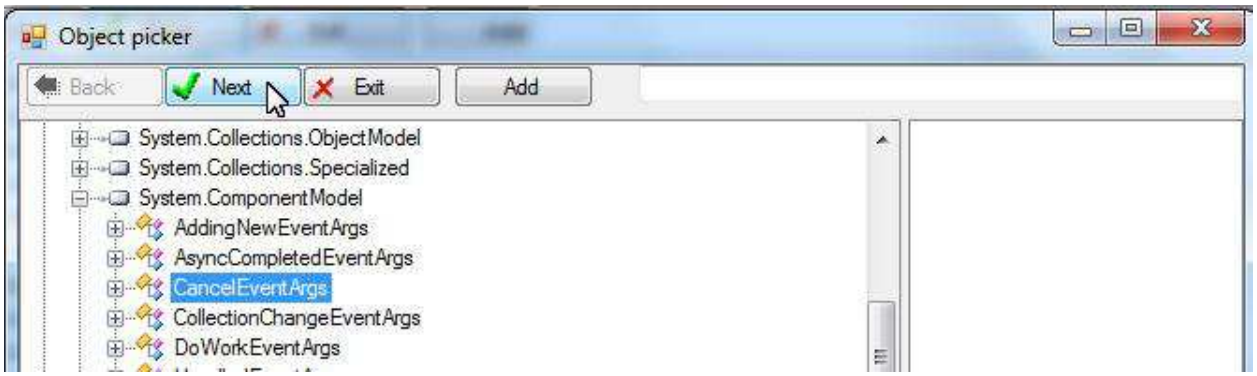


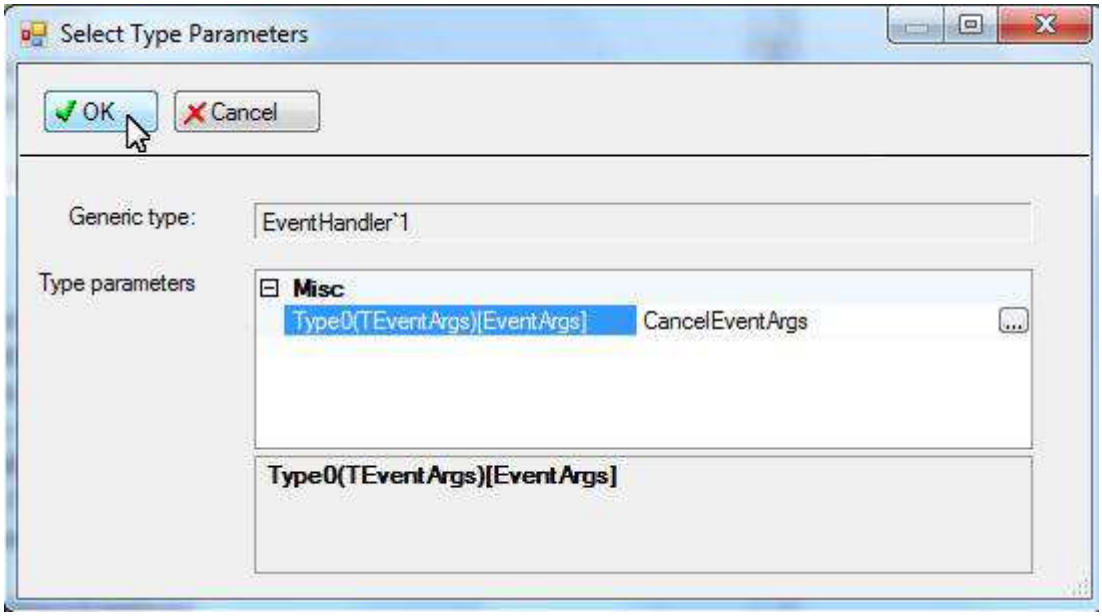


Select CancelEventArgs to provide Cancel ability:


Select Type Parameters
OK
Cancel
Generic type:
EventHandler`1
Type parameters
Misc
Type0(TEventArgs)[EventArgs]
Unknown
Type0(TEventArgs)[EventArgs]

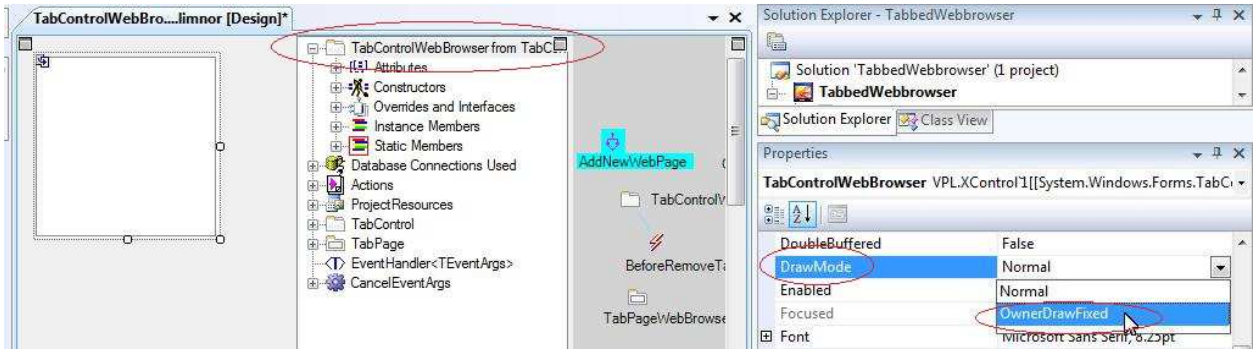

Object picker
Back
Next
Exit
Add
Primary types
Namespaces
FormComponents


Object picker
Back
Next
Exit
Add
System.Collections.ObjectModel
System.Collections.Specialized
System.ComponentModel
AddingNewEventArgs
AsyncCompletedEventArgs
CancelEventArgs
CollectionChangeEventArgs
DoWorkEventArgs



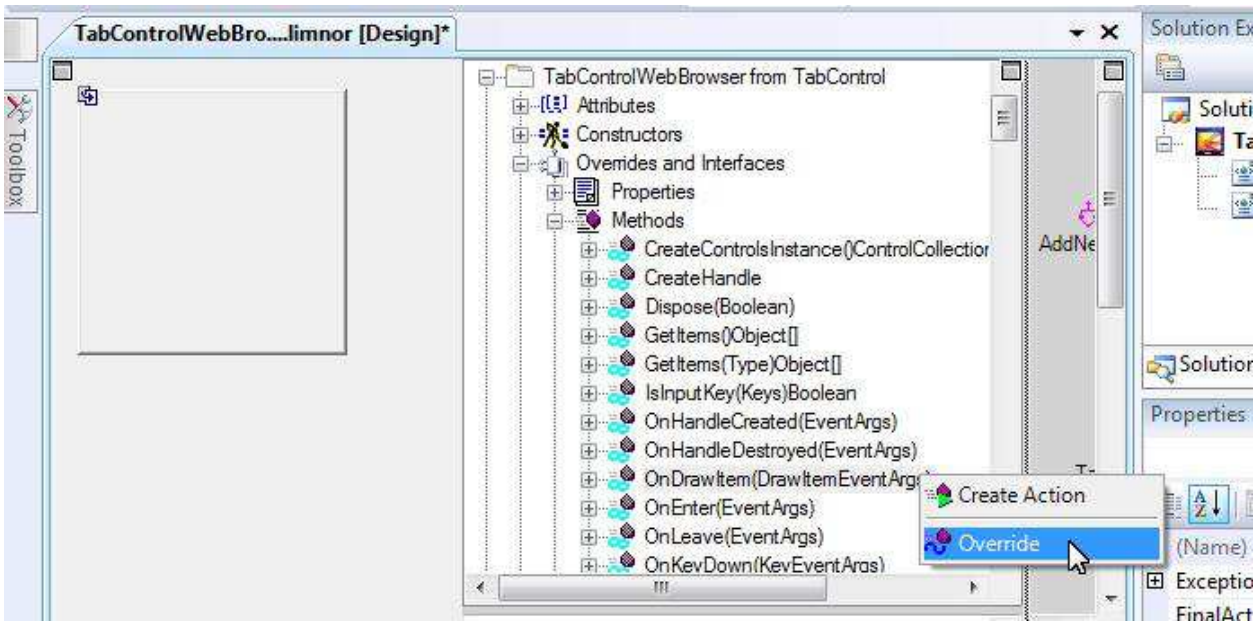
Take over tab caption drawing

Select the TabControlWebBrowser, set DrawMode to OwnerDrawFixed take over the tab caption drawing:

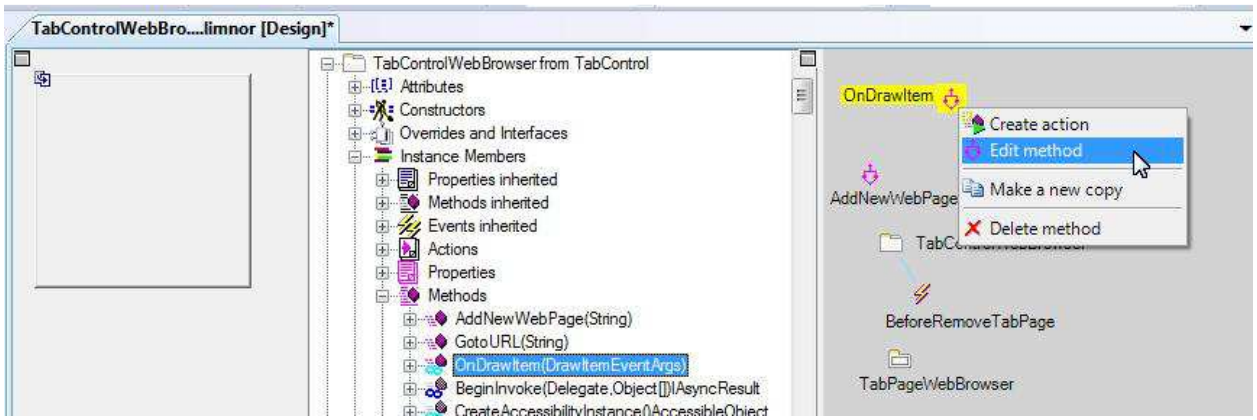


Draw tab caption

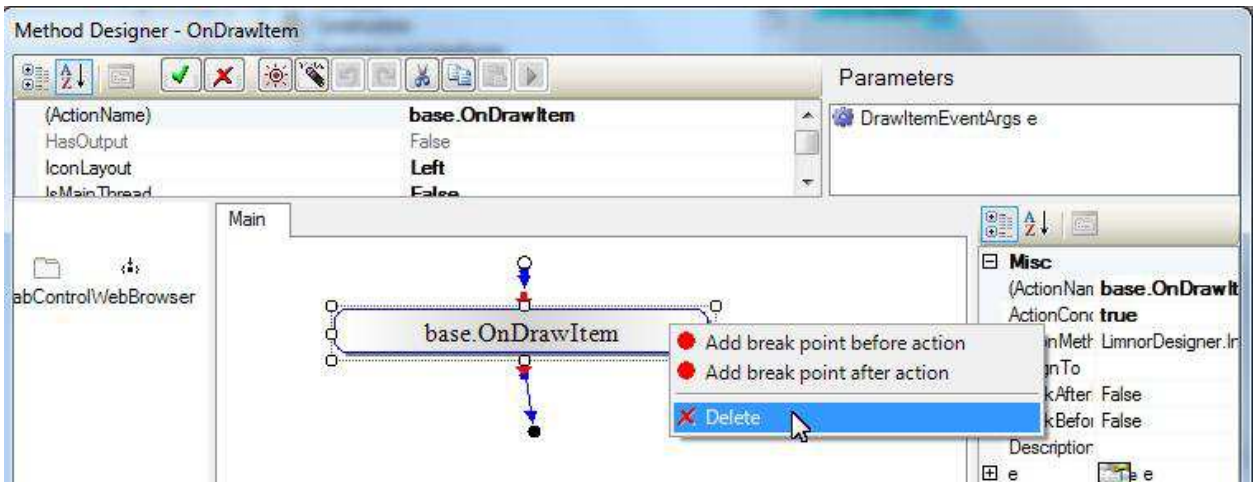
Drawings should be done by overriding `OnDrawItem` method:



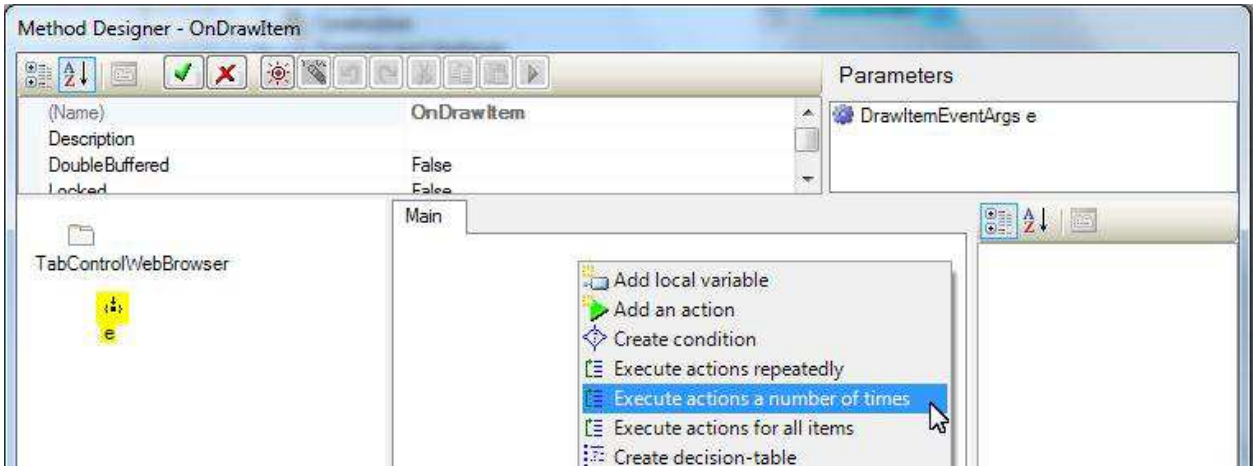
A new method is created. We may edit it to add drawing actions to it:



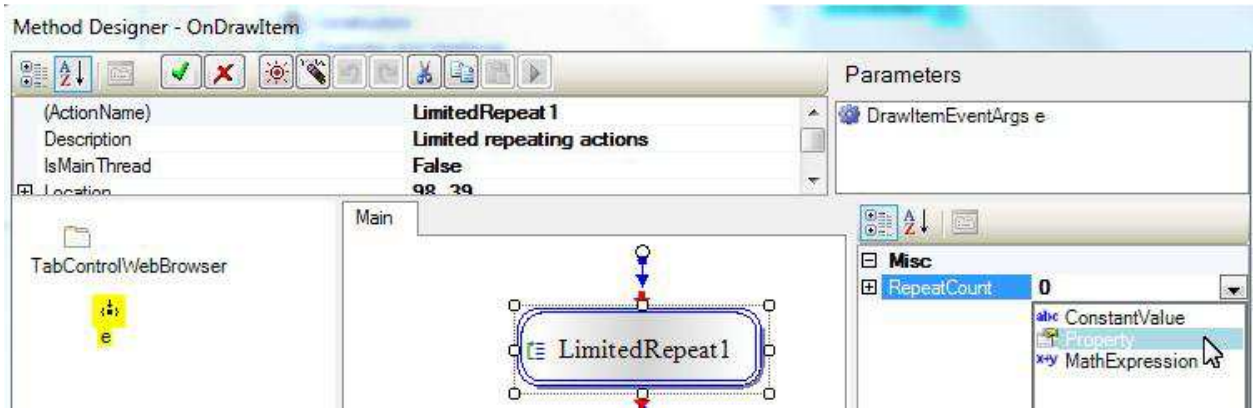
Remove the calling of the base method:

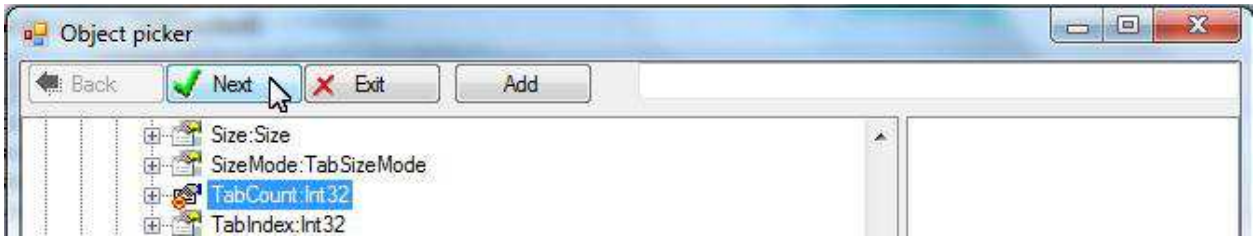


We need to draw tab captions for all tab pages. Add an “Execute actions a number of times” to apply drawing actions to all tab pages:

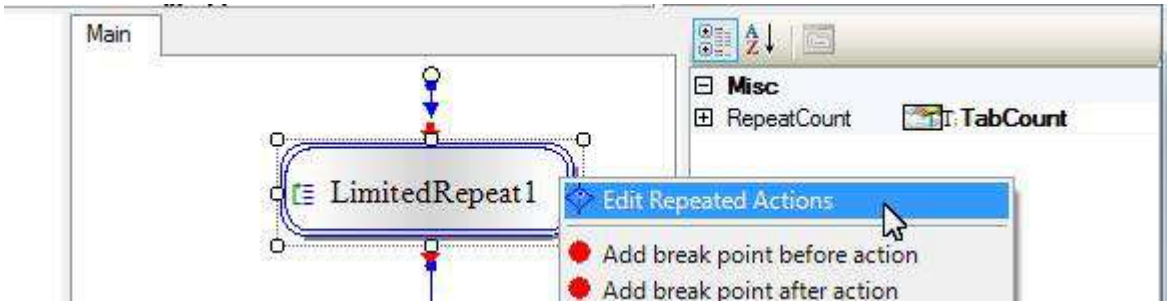


Set RepeatCount to the number of tab pages:

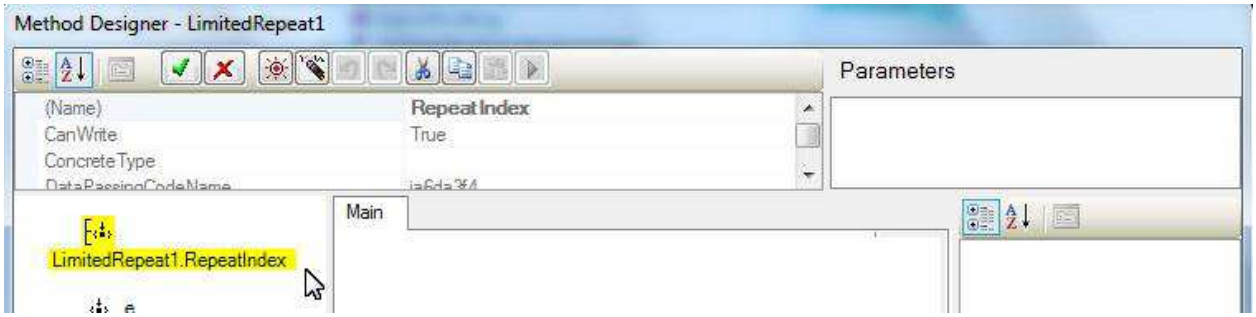




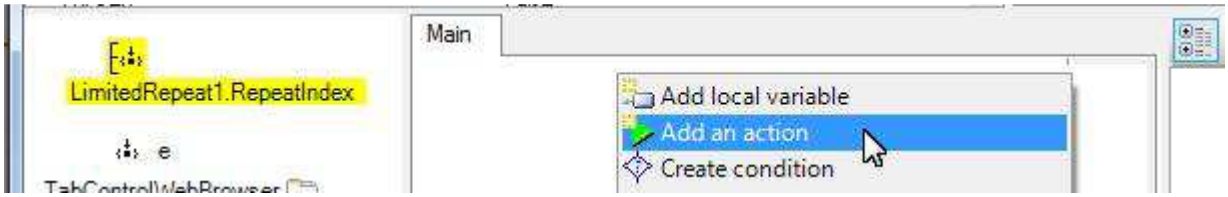
Edit the action to add actions:

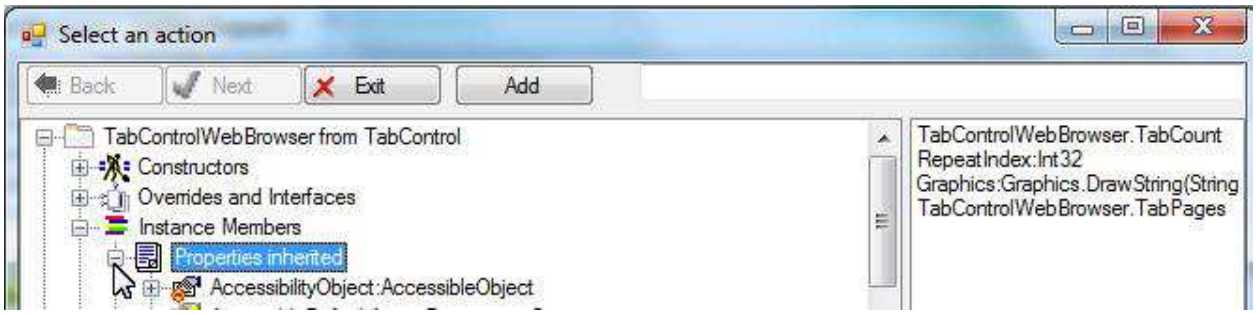


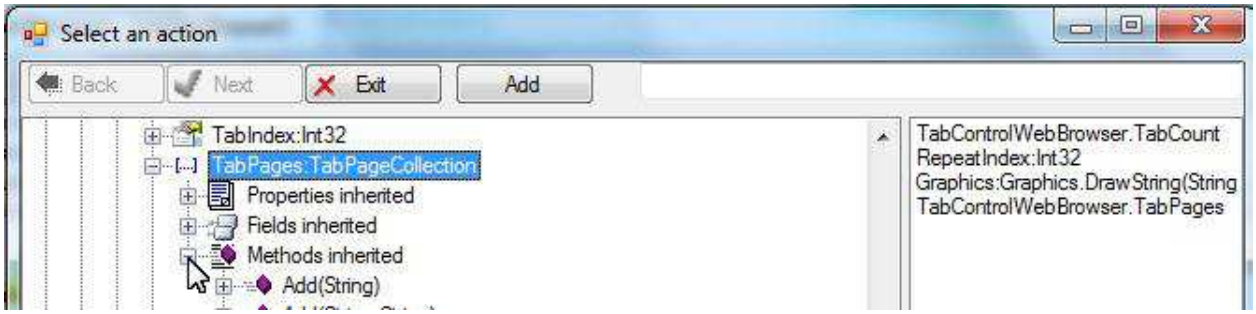
A method editor appears. RepeatIndex represents the tab page index:



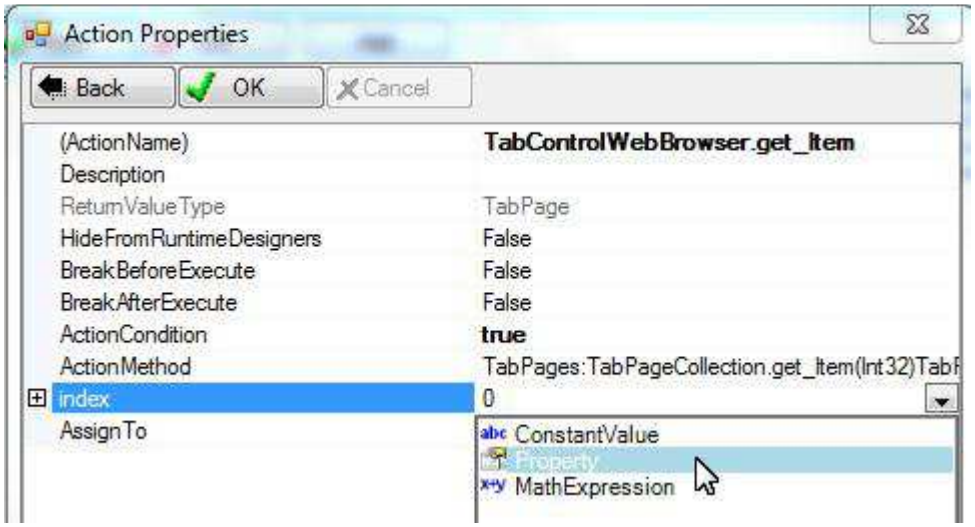
We may use RepeatIndex to the get the corresponding tab page:

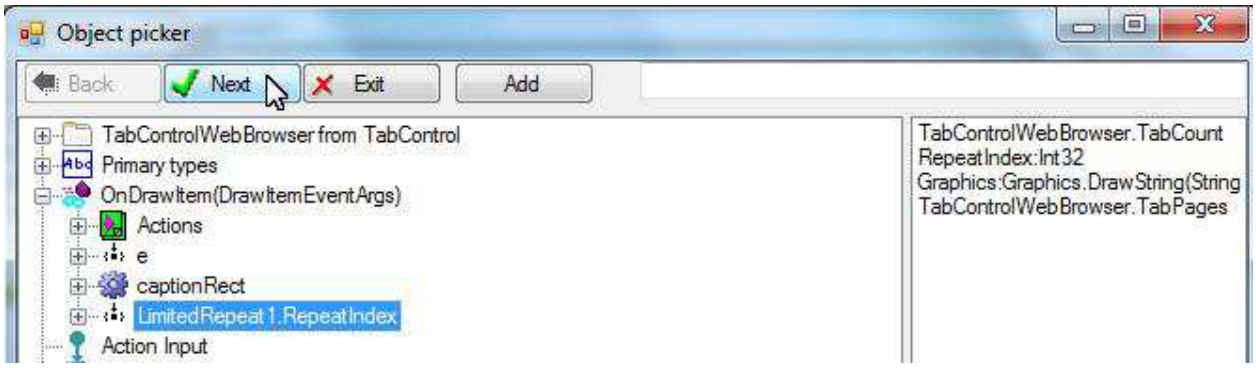




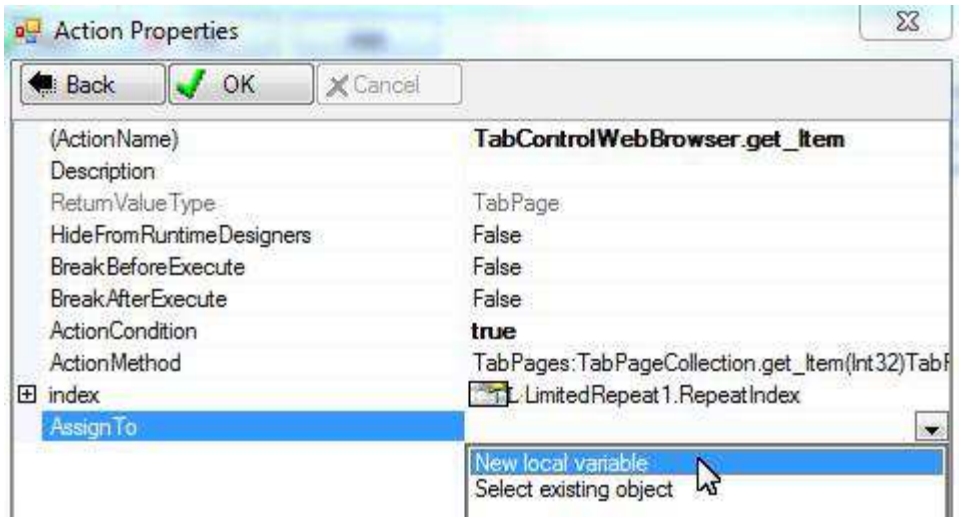


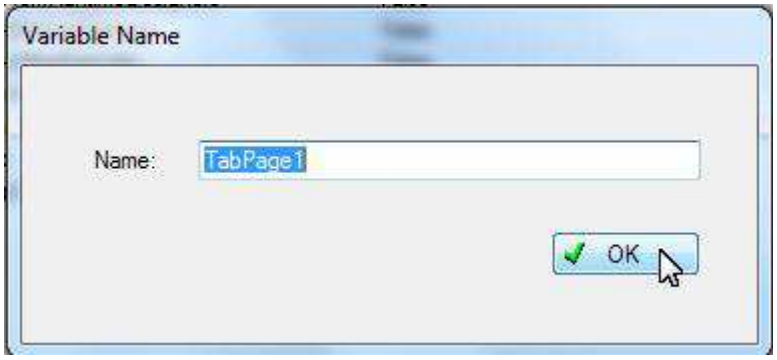
Pass RepeatIndex to the action:

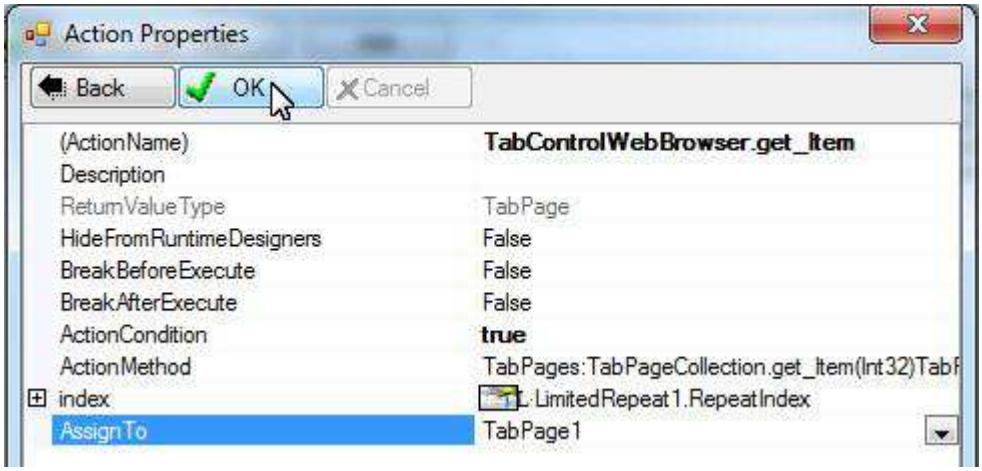




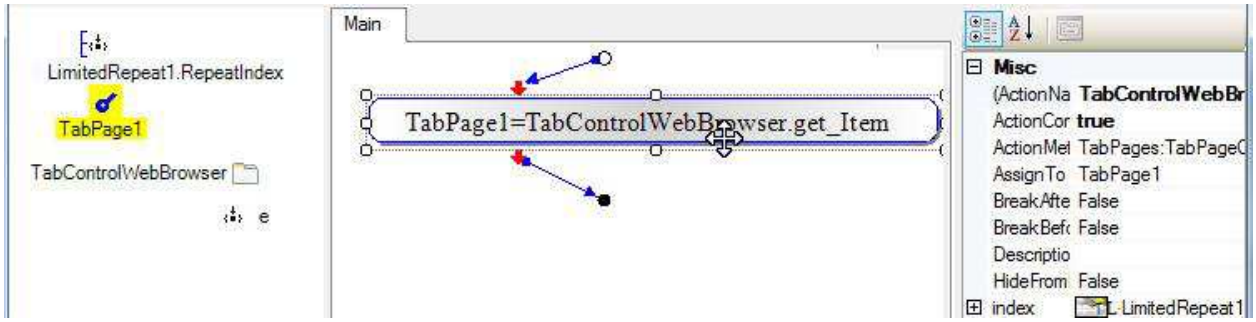
Use a new variable to hold the tab page:



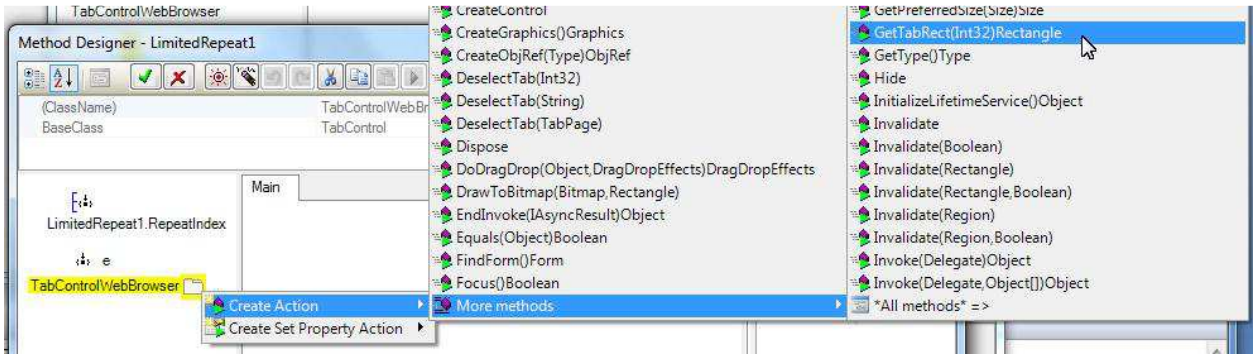




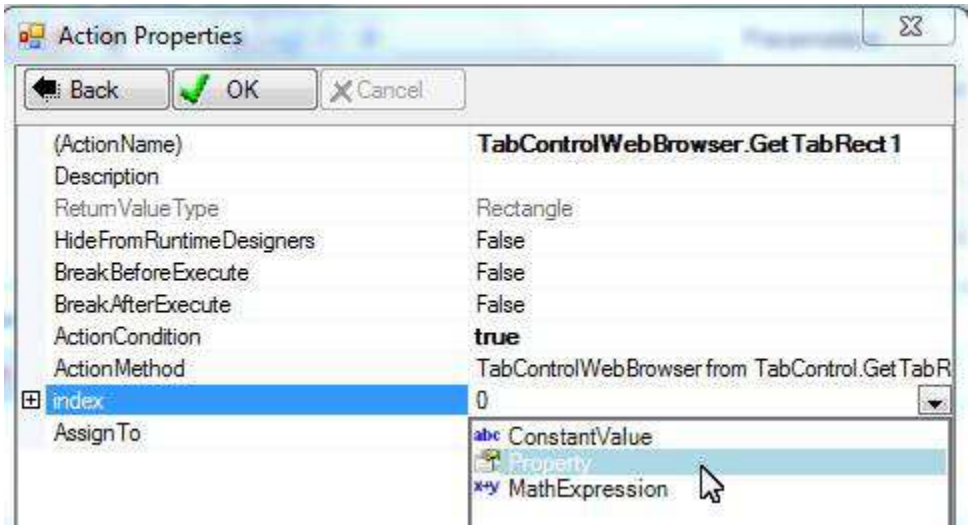
The action and variable appear:

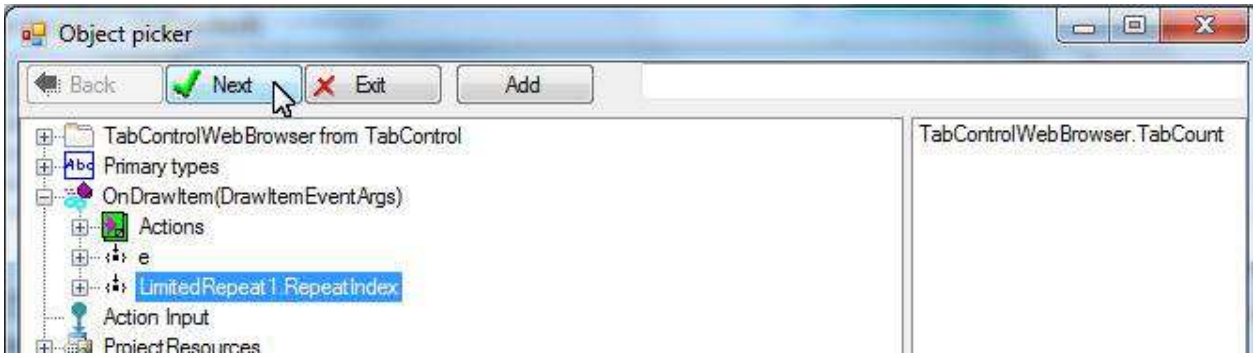


Get the rectangle for the tab caption:

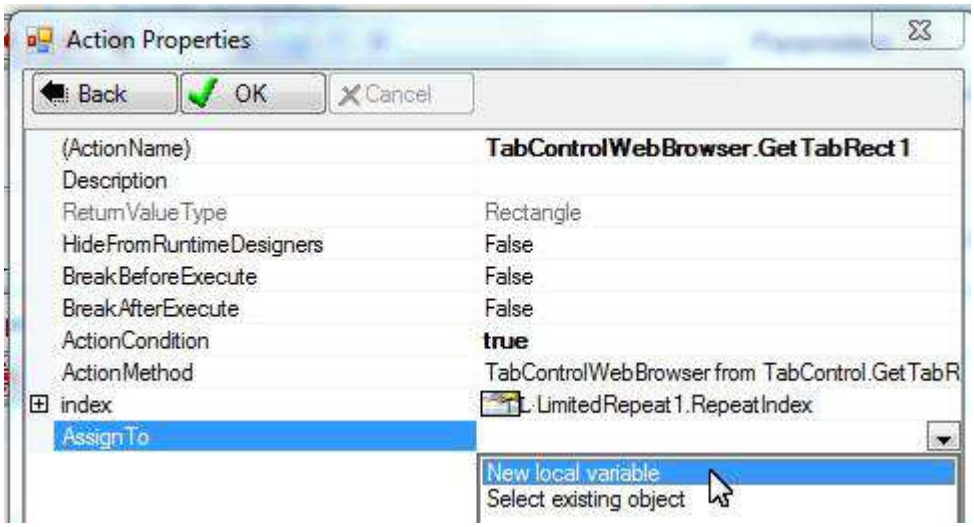


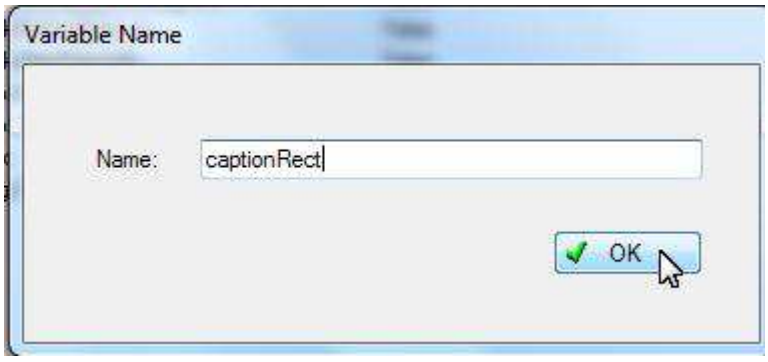
Pass the RepeatIndex to the action to indicate for which tab page we want to get the caption rectangle:

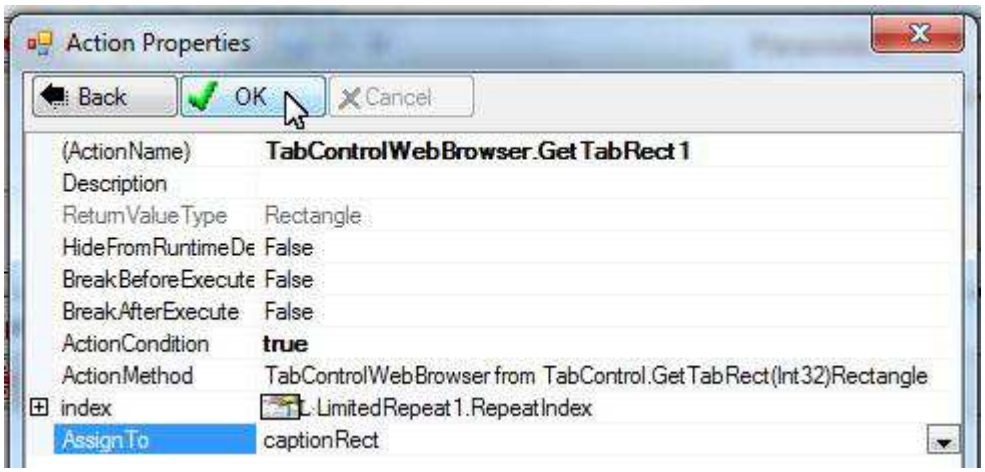




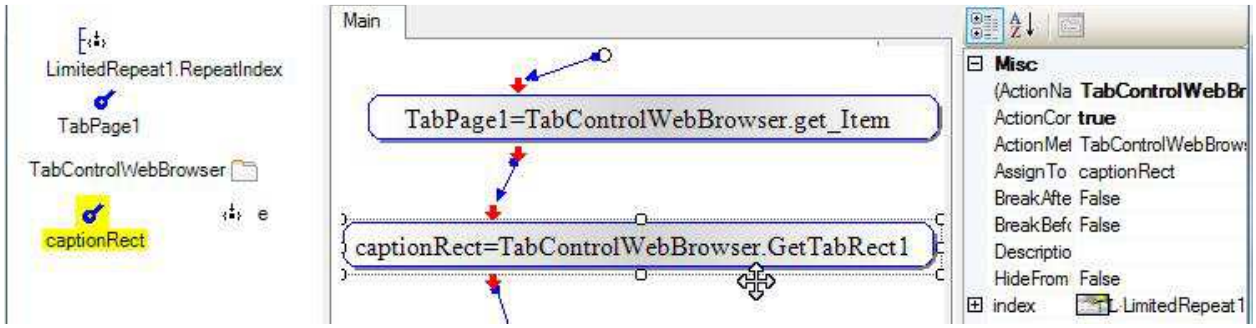
Use a new variable to get the result:



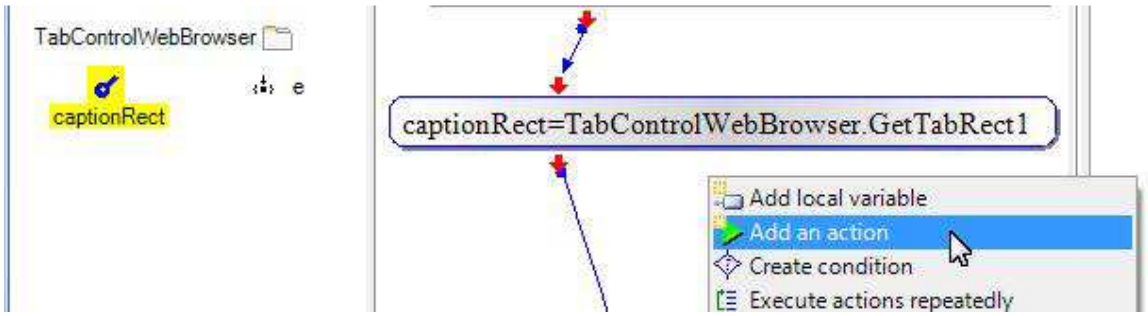


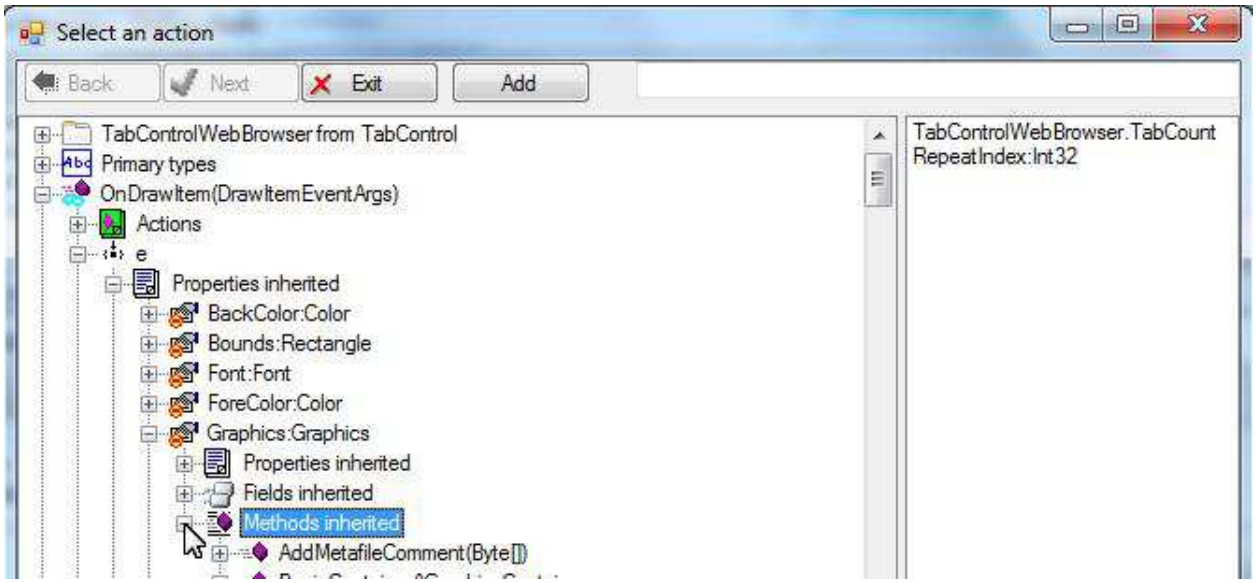


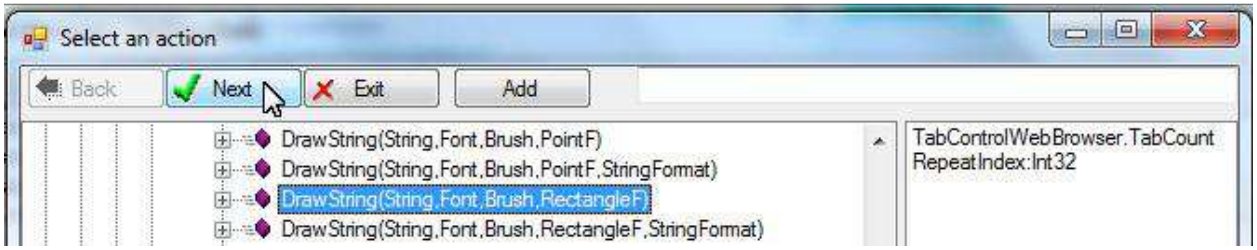
The action appears. Link it to the last action:



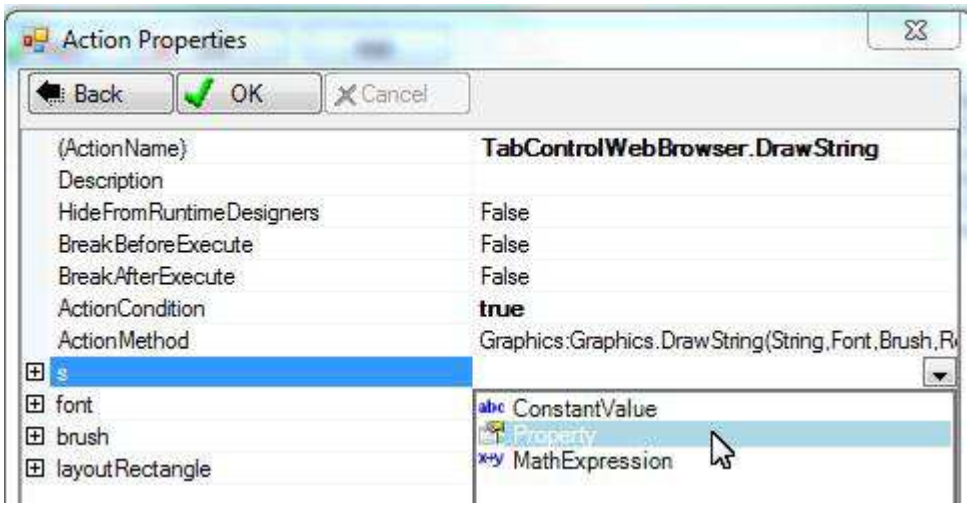
We may use an action to draw tab page caption on the rectangle:

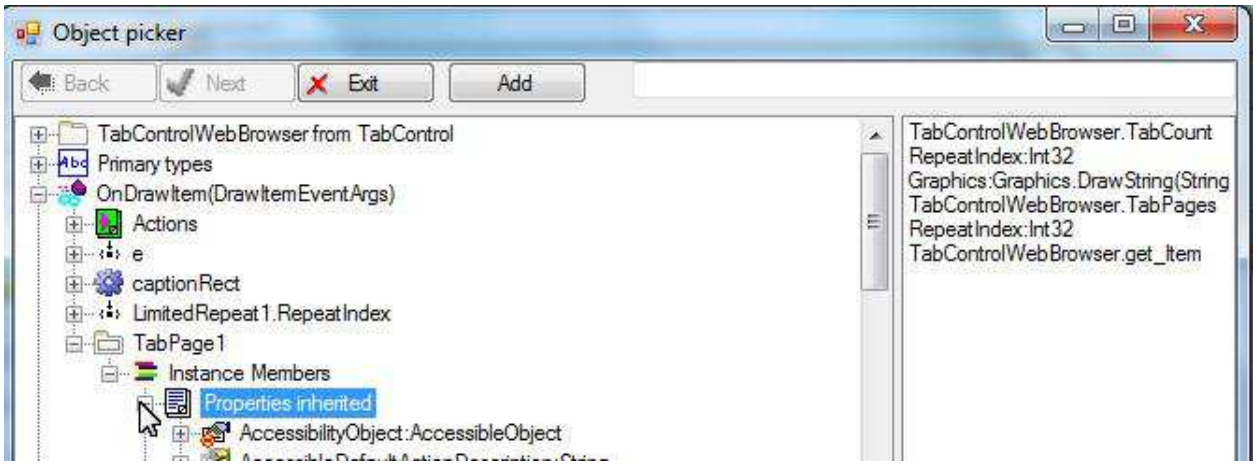


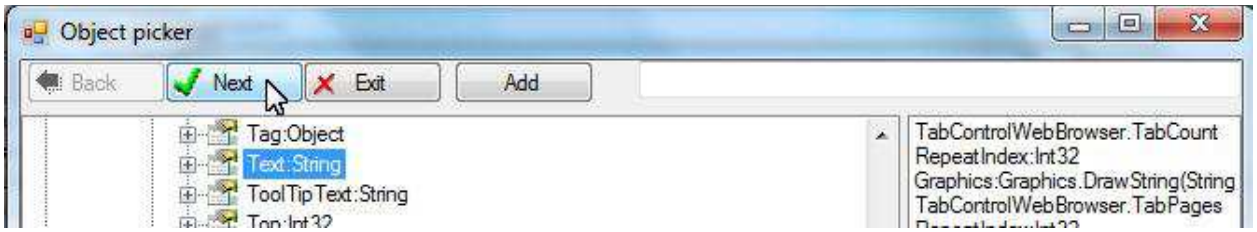




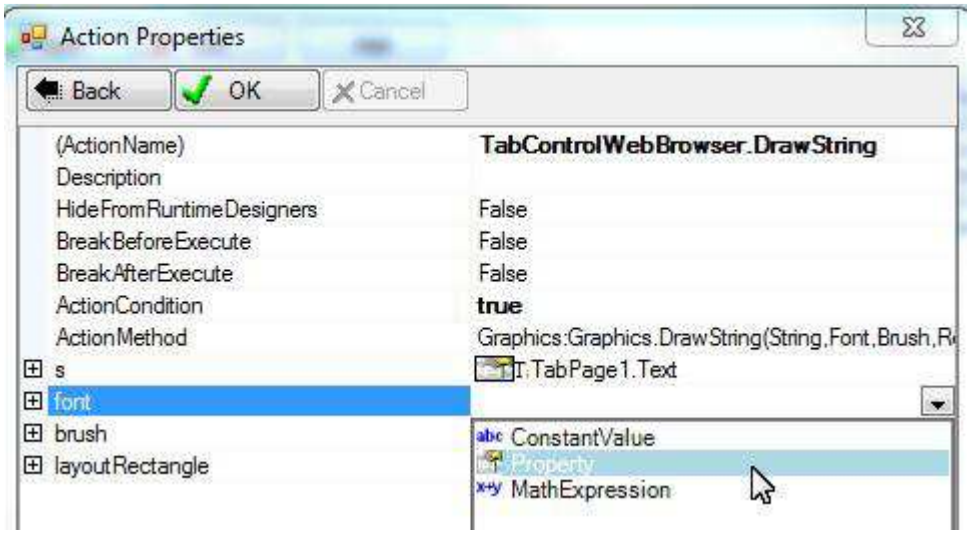
"s" is the text to draw. Select Text property of the tab page for it:

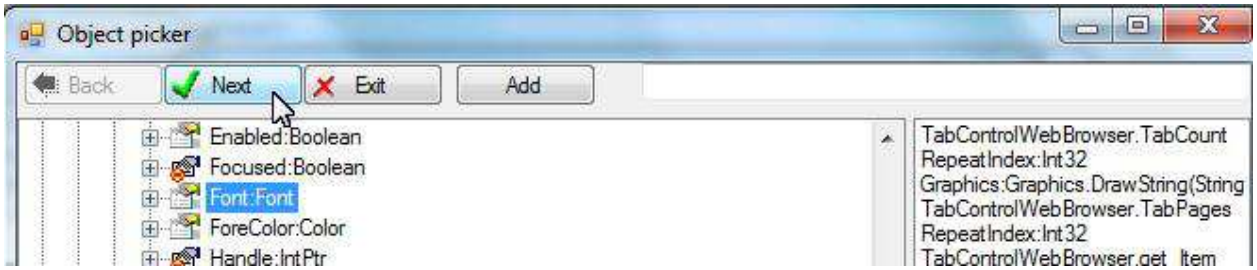




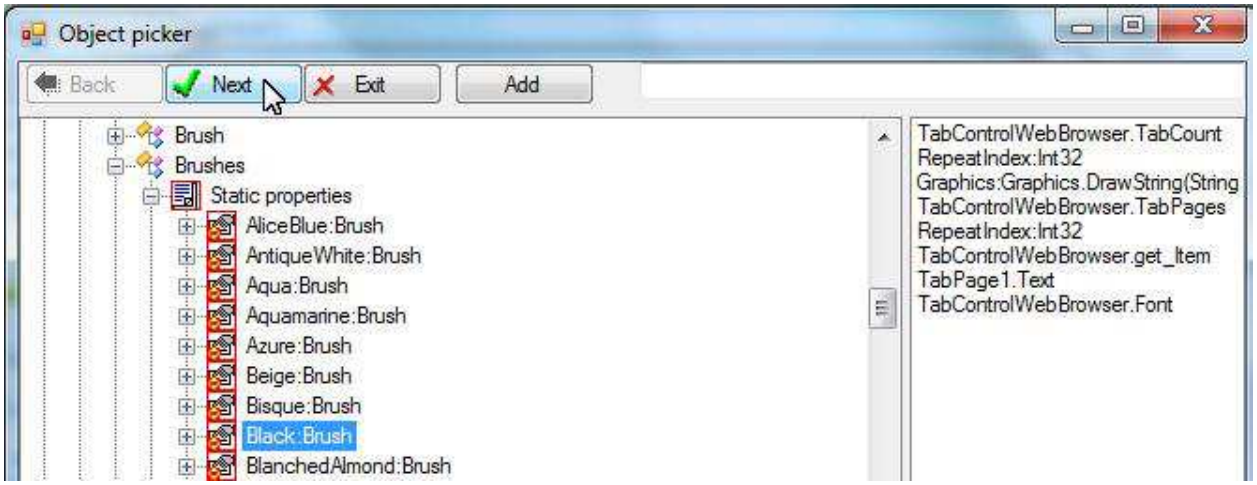


Use the TabControlWebBrowser's Font property for "font":

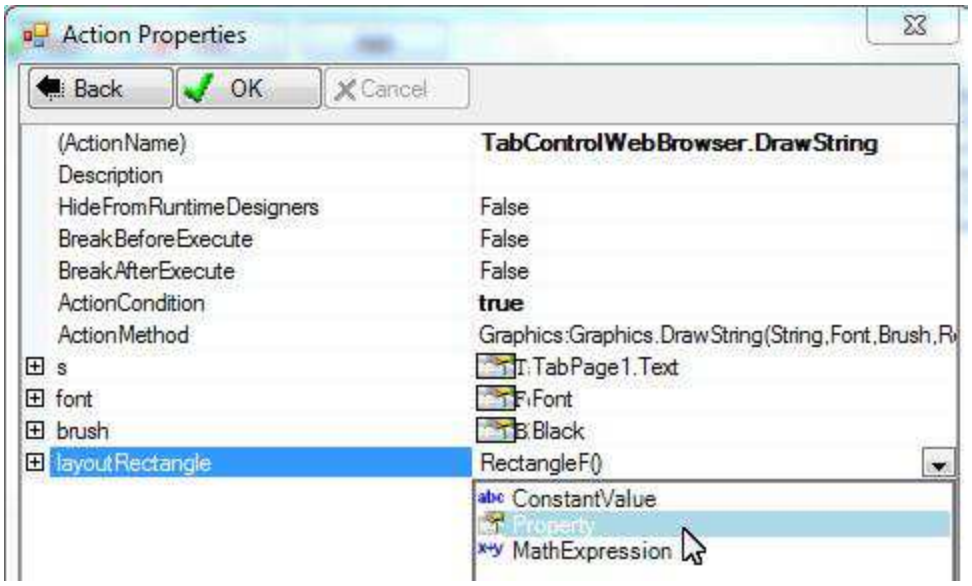


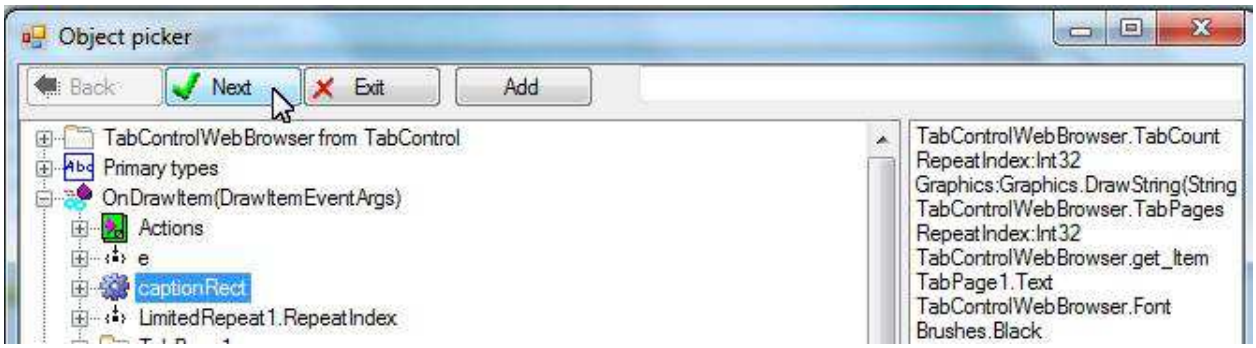


Use black brush to draw text:

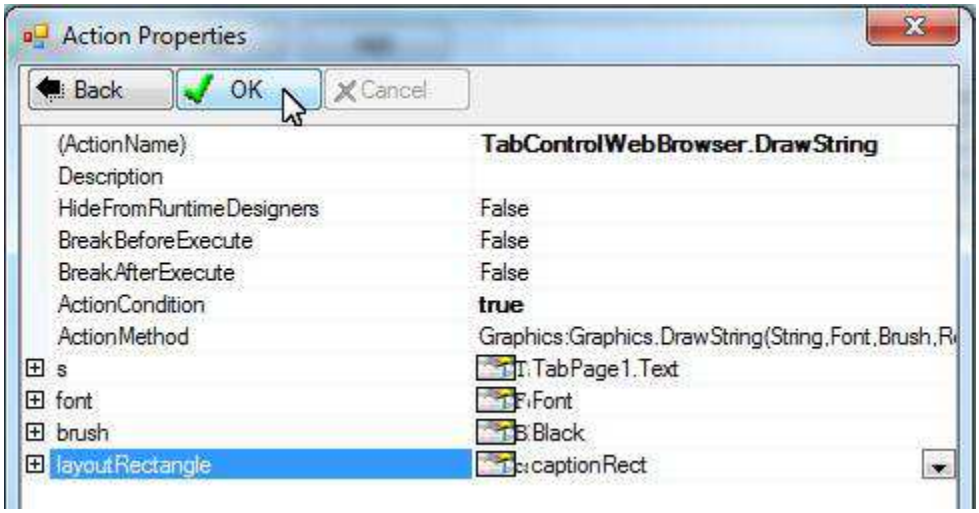


Use the caption rectangle for the drawing rectangle:

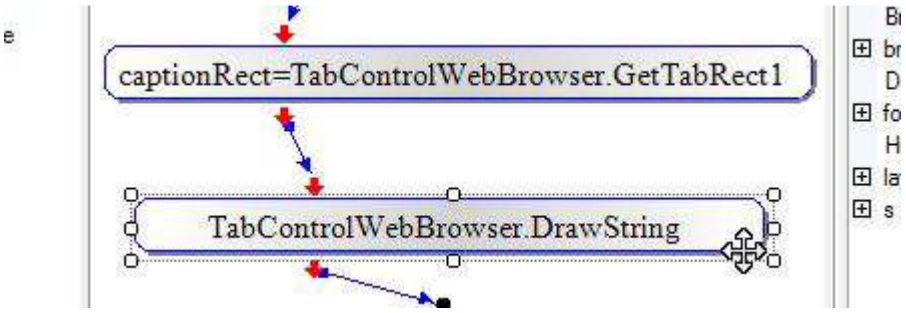




Click OK:

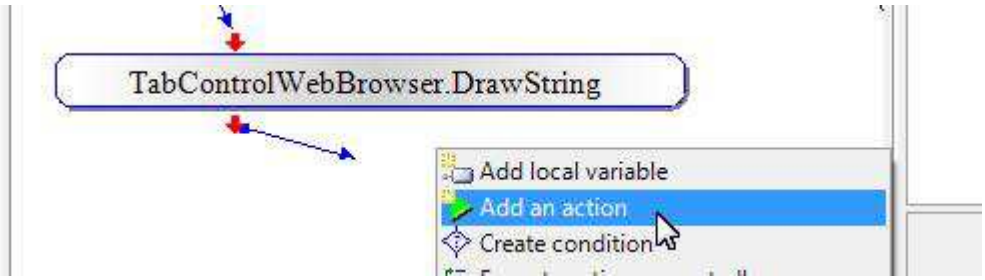


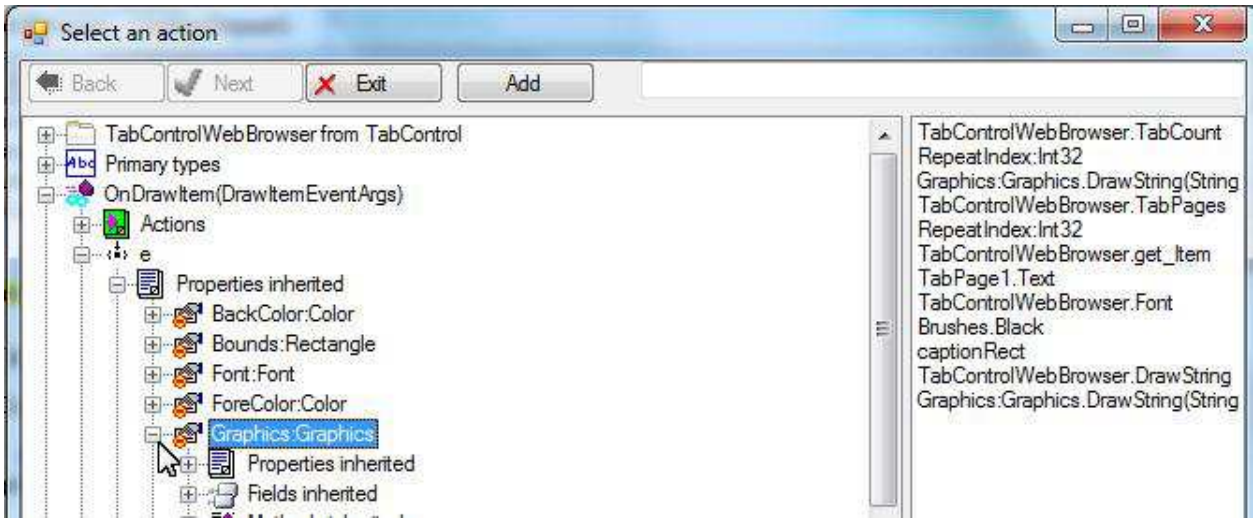
Link it to the last action:

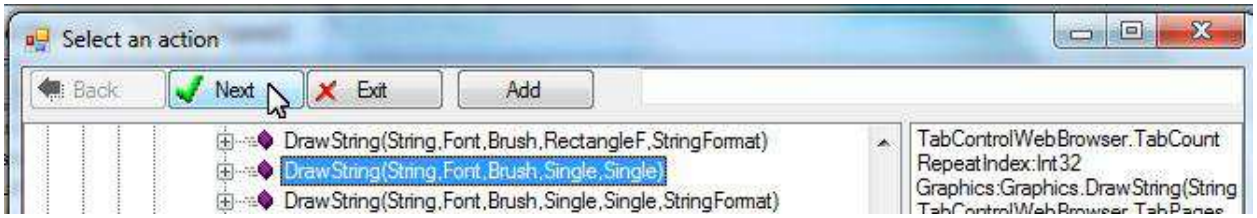


Draw "close" button

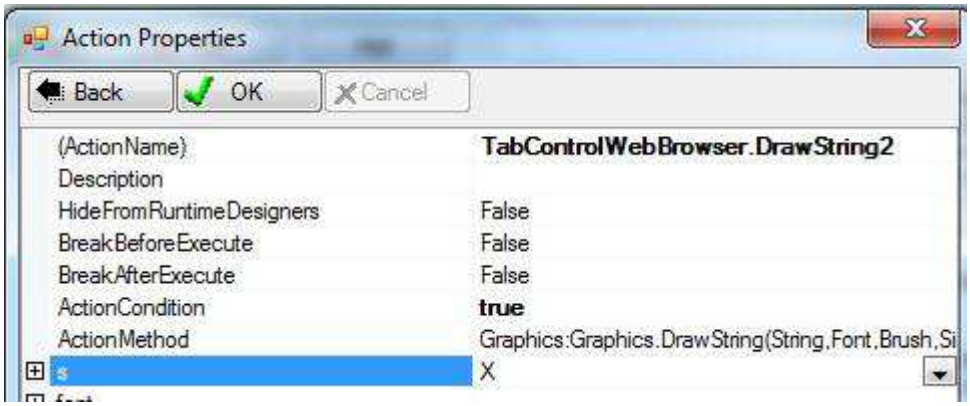
We draw a red "x" on the right end of the caption rectangle to represent a "close" button.



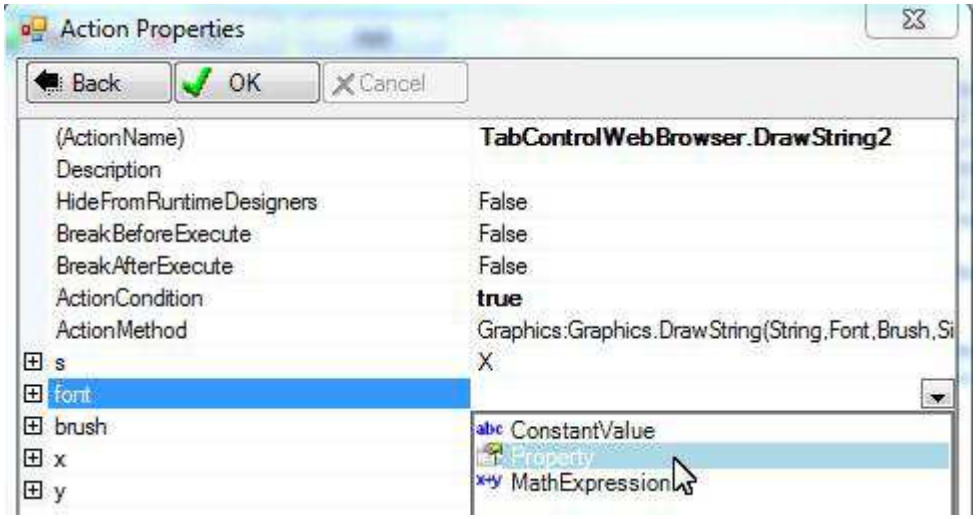


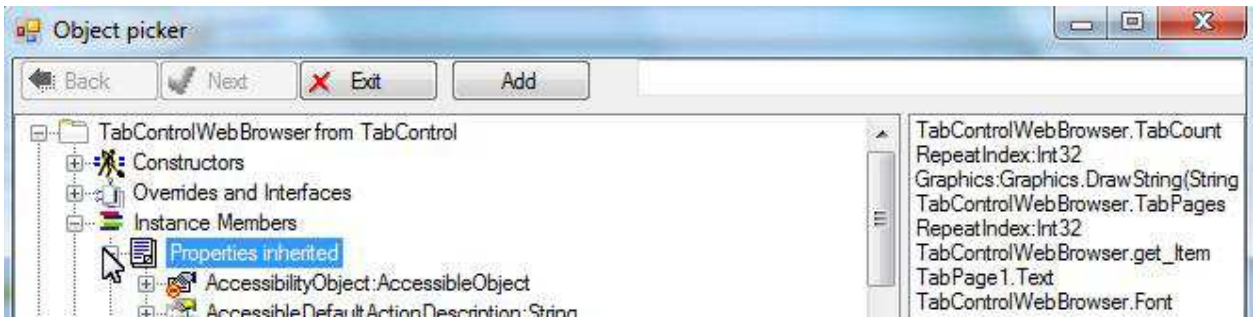


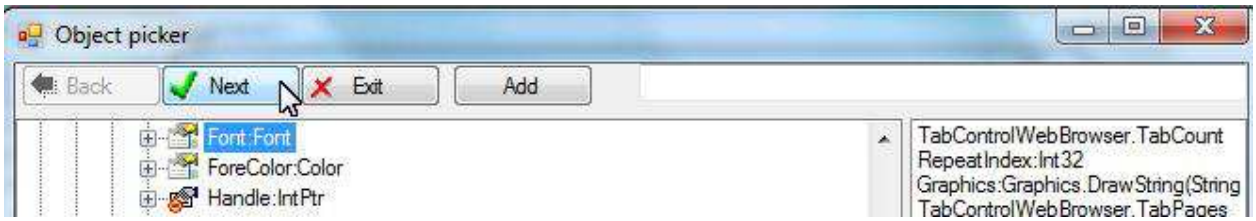
Draw an “X” to represent a “close” button:



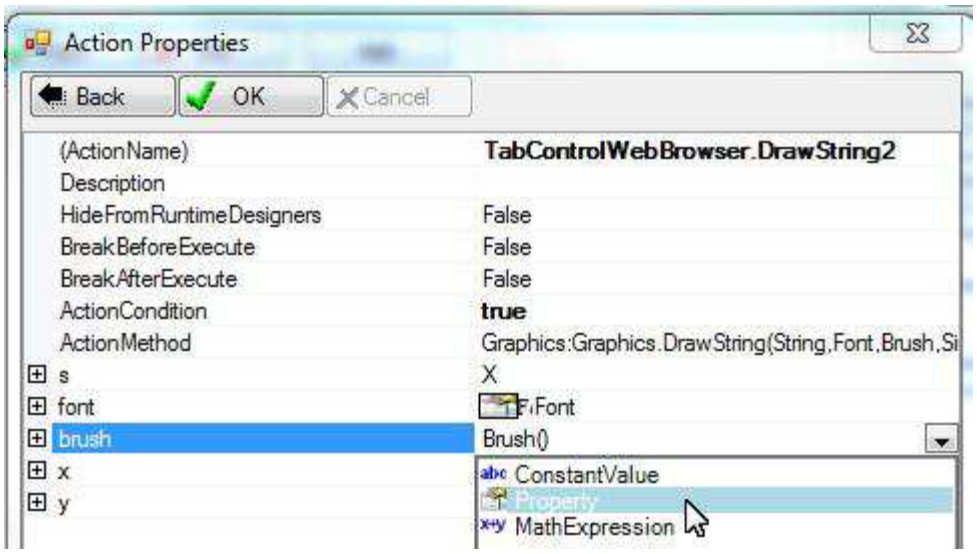
Use TabControlWebBrowser's "Font" for "font":

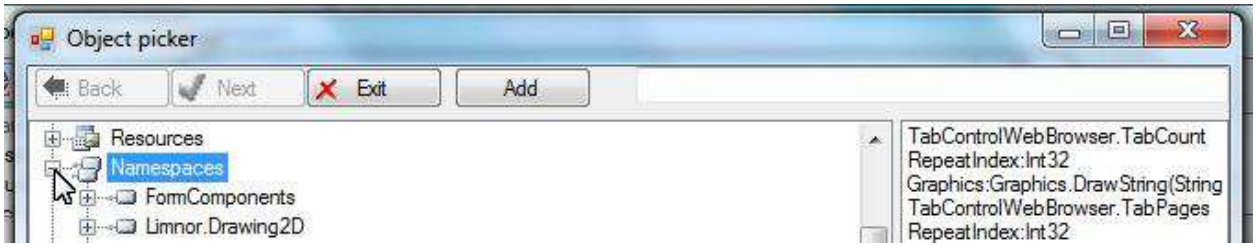


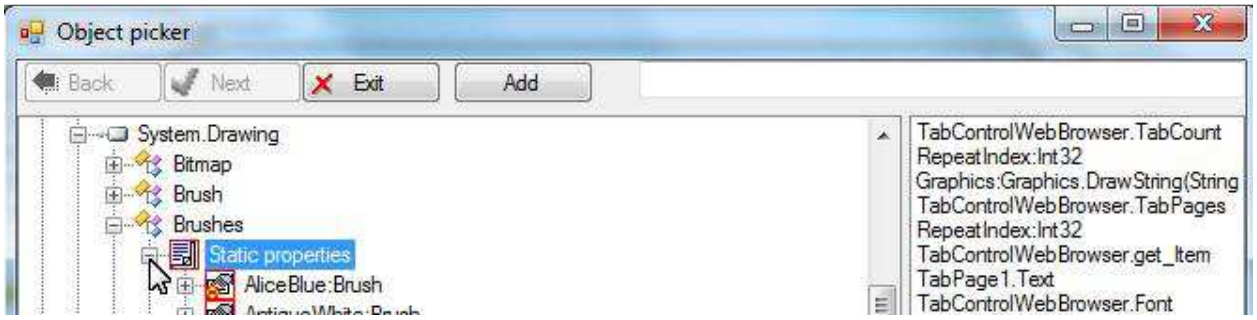


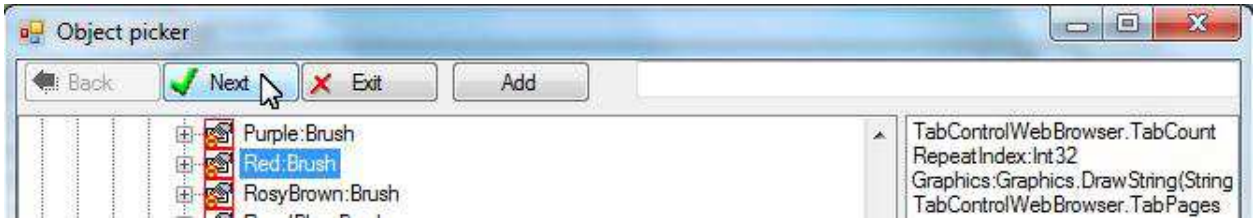


Use red brush:



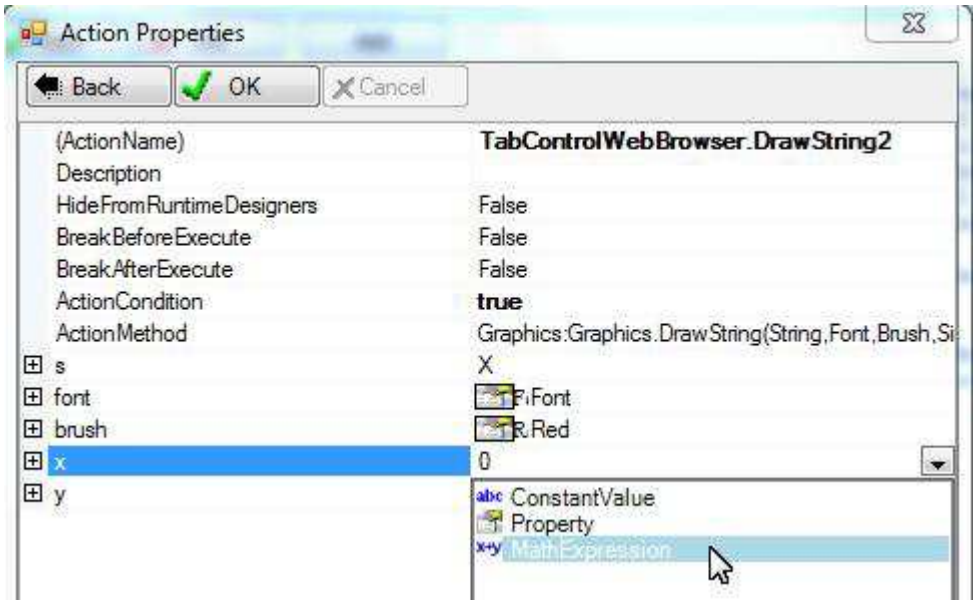


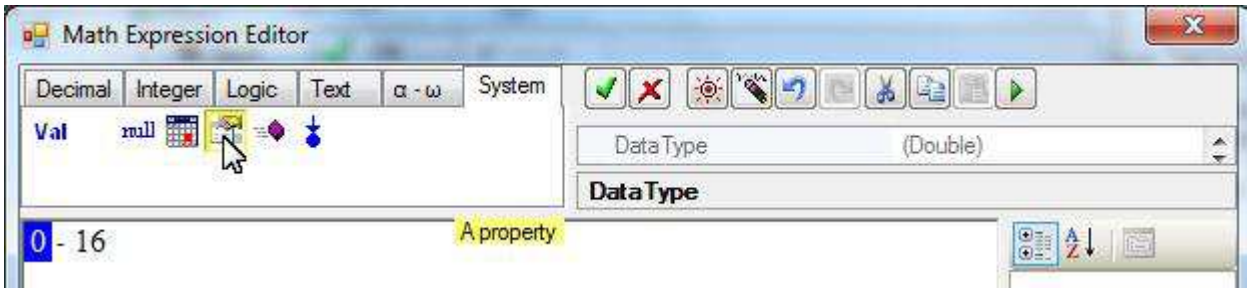


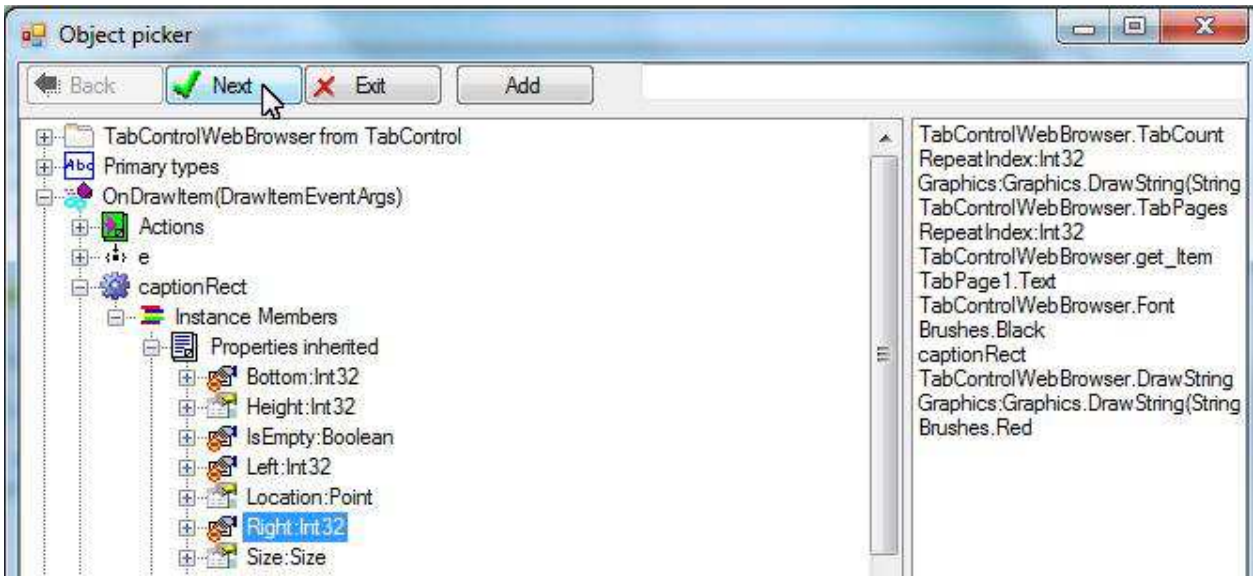


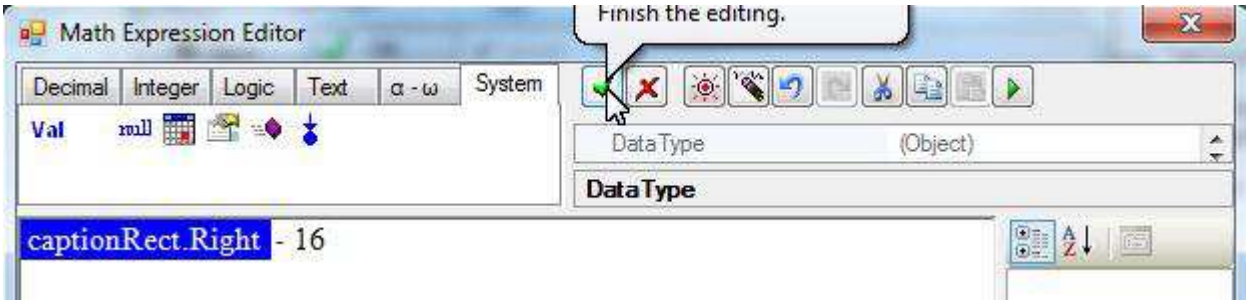
To draw “X” at the right end of the caption rectangle, set x to the following expression:

{Right side of the rectangle} – 16

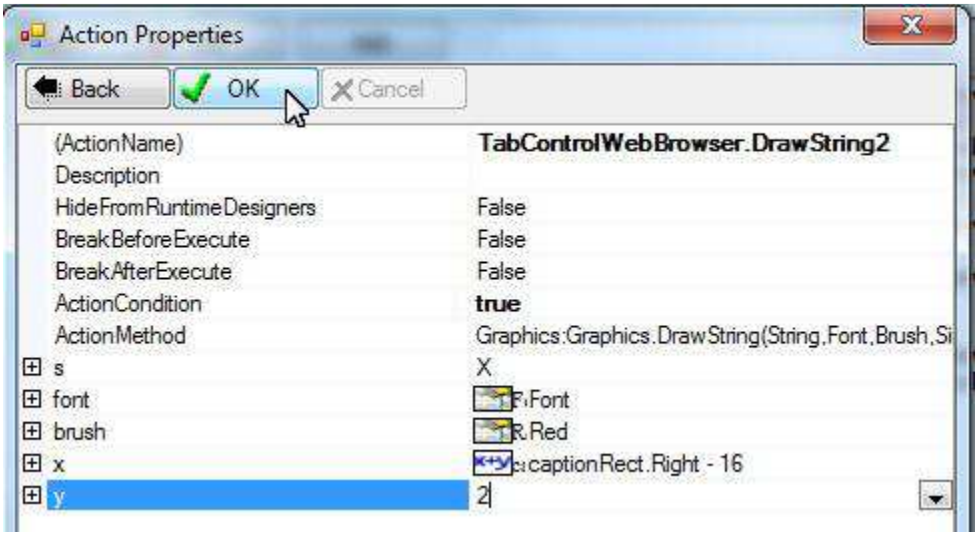

Action Properties
Back
OK
Cancel
(ActionName)
TabControlWebBrowser.DrawString2
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
Graphics:Graphics.DrawString(String,Font,Brush,Si
s
X
font
Font
brush
Red
x
0
y
ConstantValue
Property
Math Expression


Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
Data Type
(Double)
Data Type
0 - 16
A property


Object picker
Back
Next
Exit
Add
TabControlWebBrowser from TabControl
Primary types
OnDrawItem(DrawItemEventArgs)
Actions
e
captionRect
Instance Members
Properties inherited
Bottom:Int32
Height:Int32
IsEmpty:Boolean
Left:Int32
Location:Point
Right:Int32
Size:Size
TabControlWebBrowser.TabCount
RepeatIndex:Int32
Graphics:Graphics.DrawString(String
TabControlWebBrowser.TabPages
RepeatIndex:Int32
TabControlWebBrowser.get_Item
TabPage1.Text
TabControlWebBrowser.Font
Brushes.Black
captionRect
TabControlWebBrowser.DrawString
Graphics:Graphics.DrawString(String
Brushes.Red

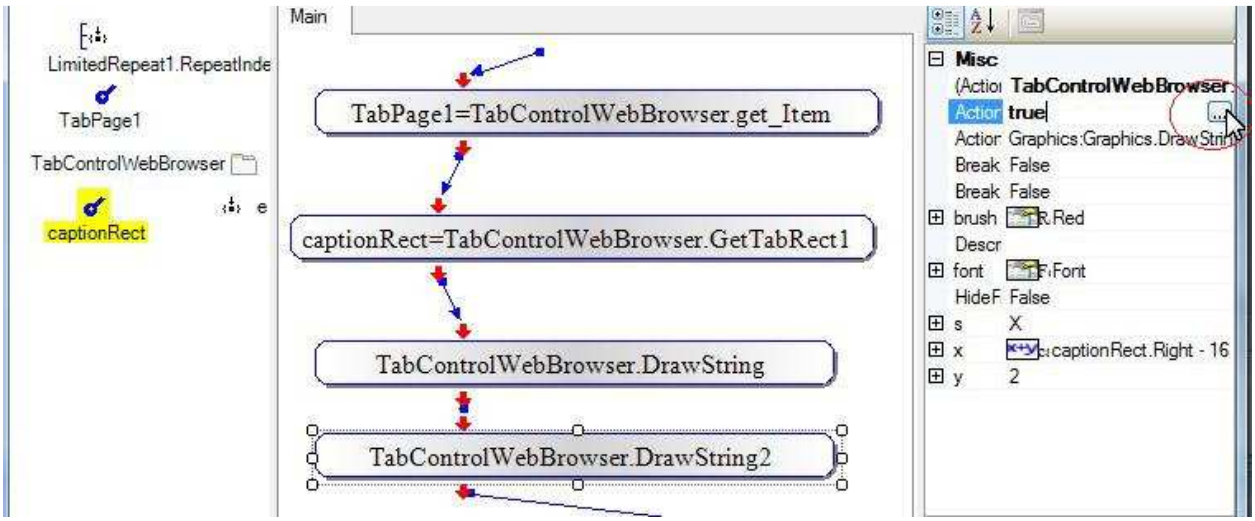


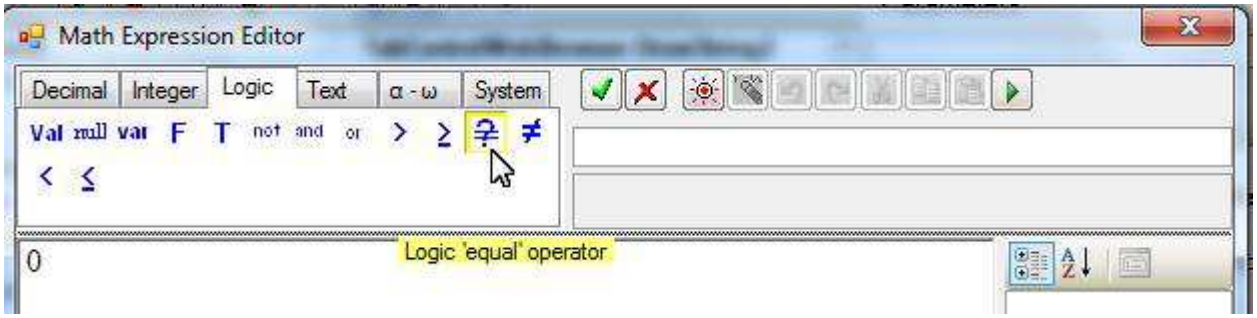
Set “y” to 2. Click OK:

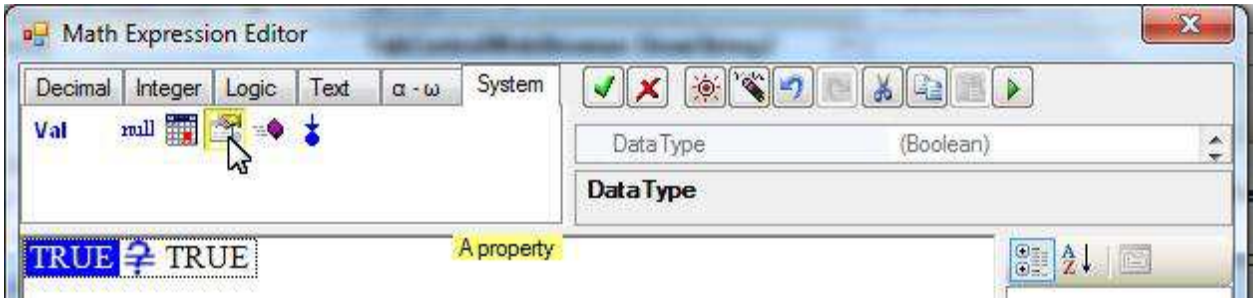


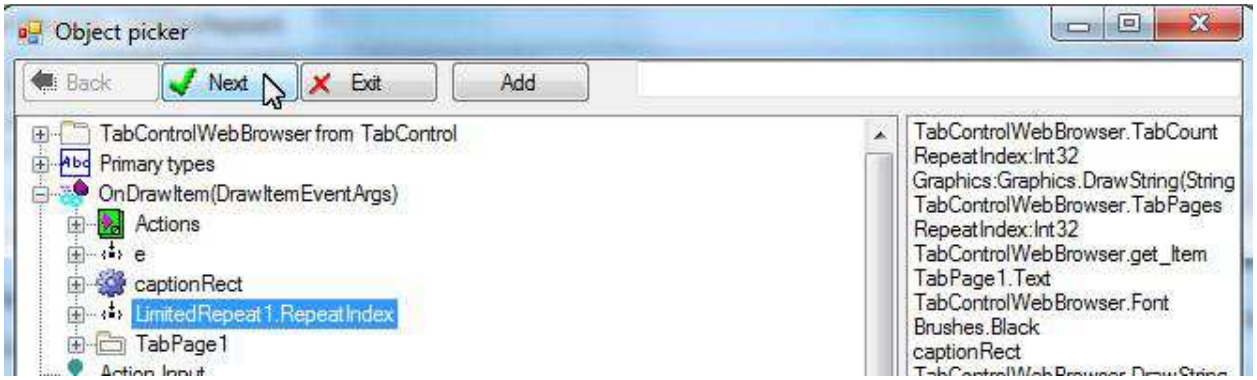
Link it to the last action.

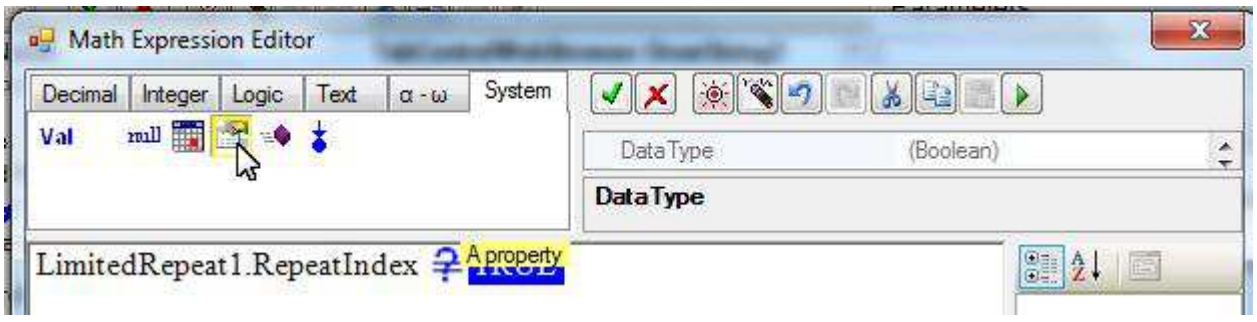
Suppose we want to only draw a red “X” on the selected tab page. We may set the ActionCondition to check whether the RepeatIndex equals the SelectedIndex property of the tab control:

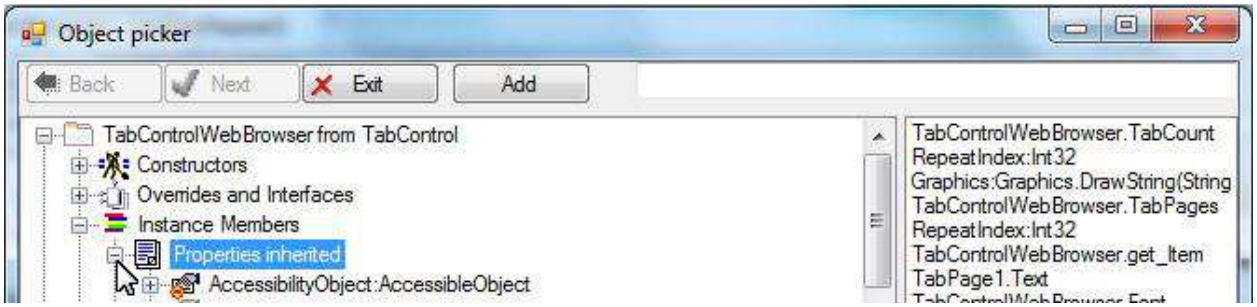



Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
var
F
T
not
and
or
0
Logic 'equal' operator

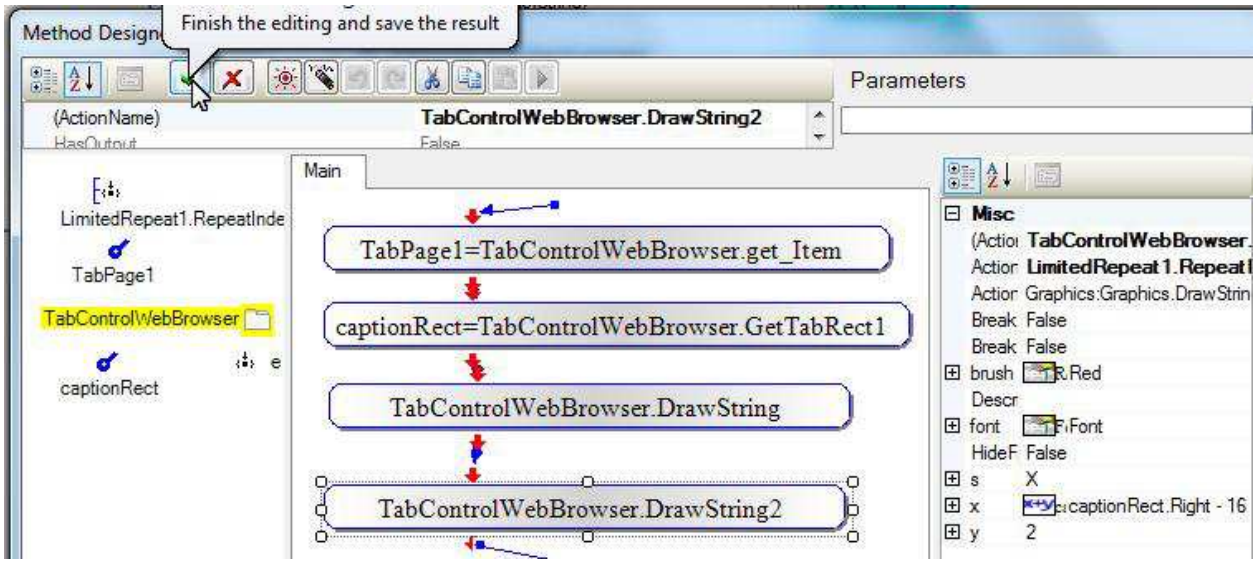

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(Boolean)
DataType
TRUE
TRUE
A property

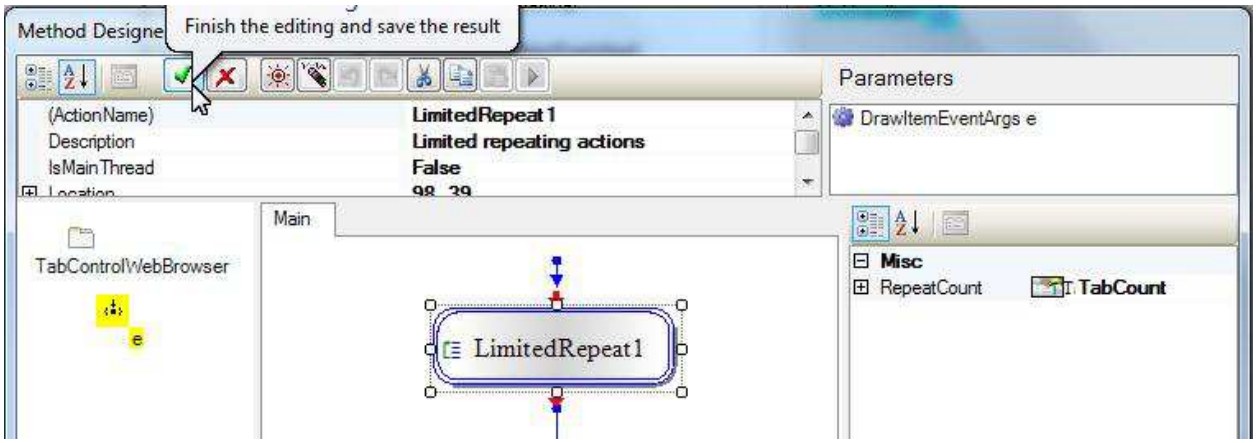

Object picker
Back
Next
Exit
Add
TabControlWebBrowser from TabControl
Primary types
OnDrawItem(DrawItemEventArgs)
Actions
e
captionRect
LimitedRepeat1.RepeatIndex
TabPage1
TabControlWebBrowser.TabCount
RepeatIndex:Int32
Graphics:Graphics.DrawString(String
TabControlWebBrowser.TabPages
RepeatIndex:Int32
TabControlWebBrowser.get_Item
TabPage1.Text
TabControlWebBrowser.Font
Brushes.Black
captionRect


Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(Boolean)
DataType
LimitedRepeat1.RepeatIndex
A property


Object picker
Back
Next
Exit
Add
TabControlWebBrowser from TabControl
Constructors
Overrides and Interfaces
Instance Members
Properties inherited
AccessibilityObject:AccessibleObject
TabControlWebBrowser.TabCount
RepeatIndex:Int32
Graphics:Graphics.DrawString(String
TabControlWebBrowser.TabPages
RepeatIndex:Int32
TabControlWebBrowser.get_Item
TabPage1.Text

That is all the drawings:

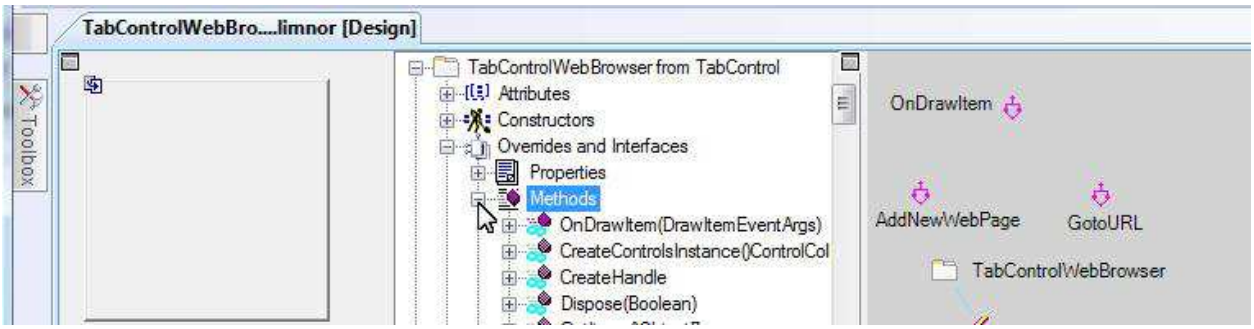


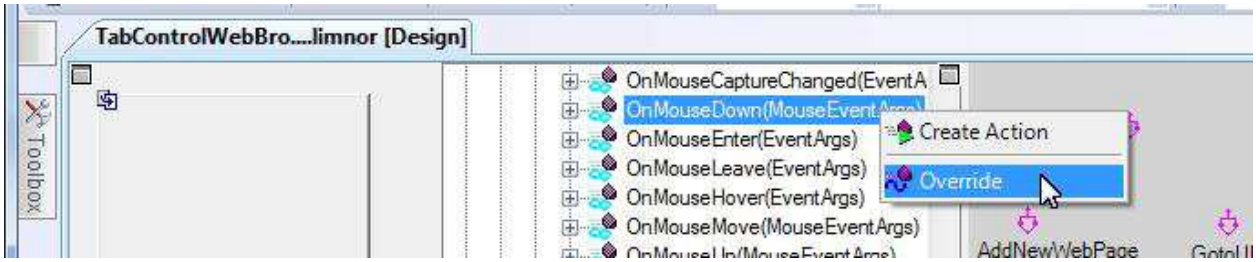


Handle Mouse Down

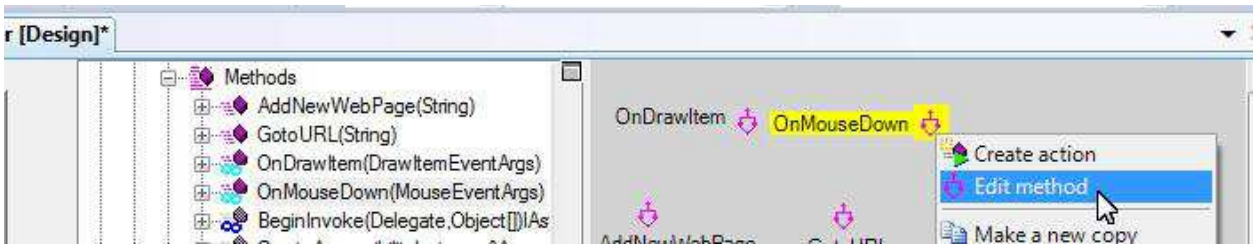
Now a red X will appear on the tab caption. At the mouse down event we want to check whether the mouse is on the red X.

We may override the method OnMouseDown to do it:

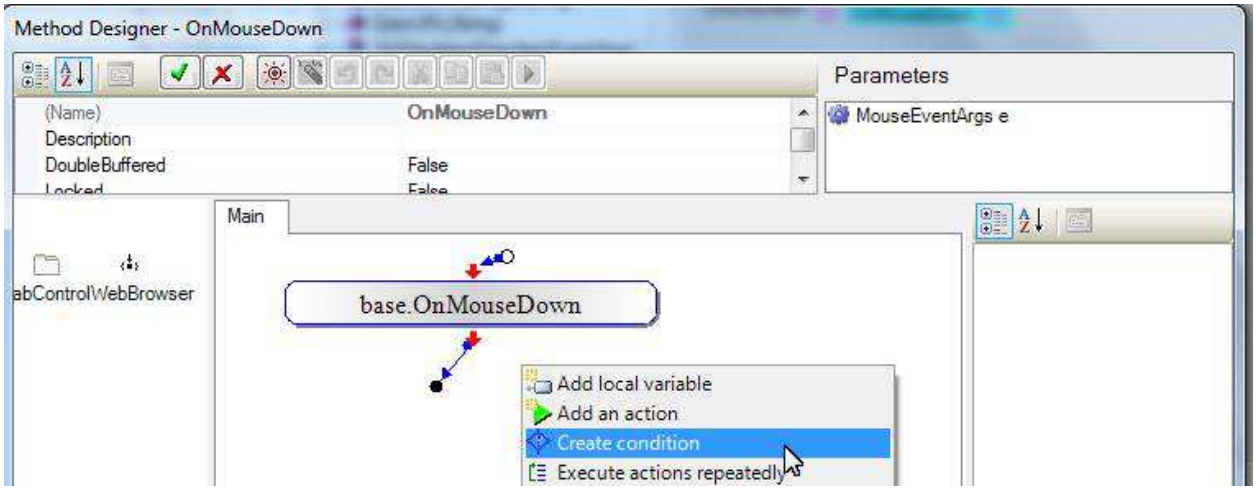




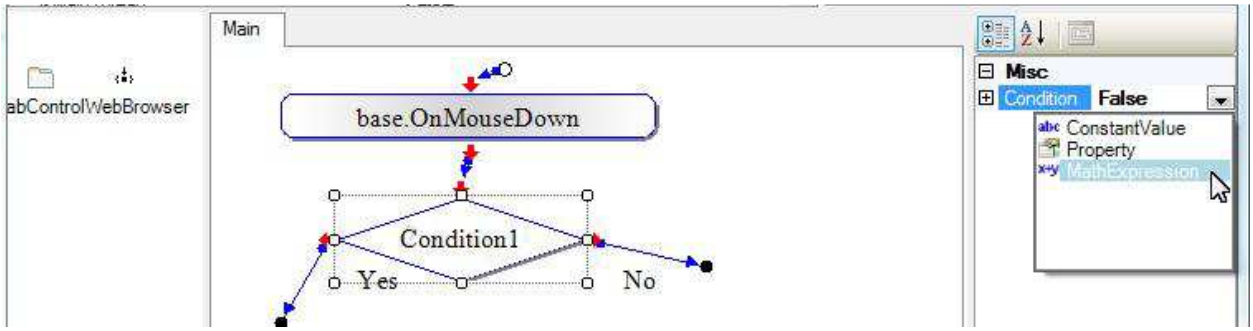
Edit the method to add actions to it:



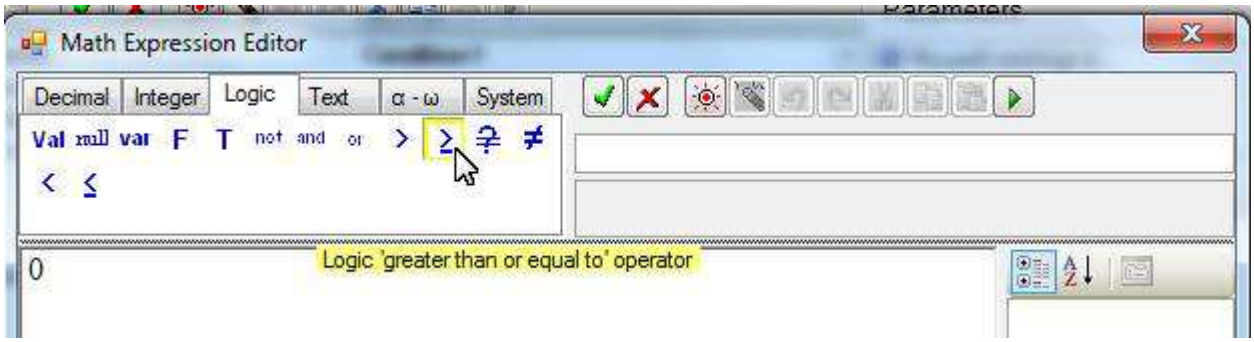
First, add a Condition action to make sure that there is a selected tab page:

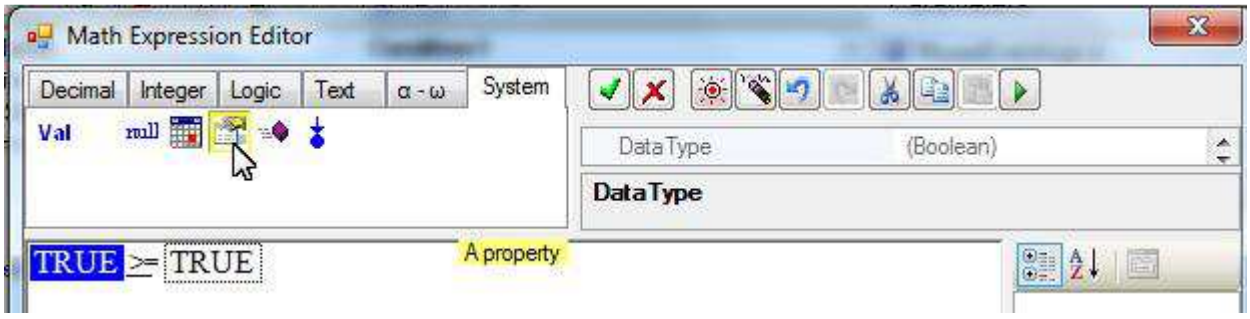


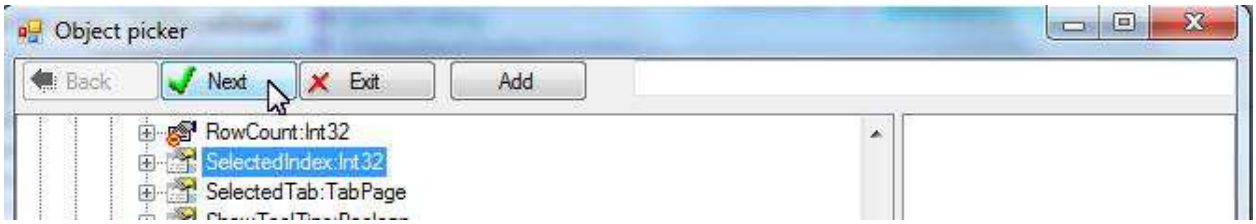
Set the condition:

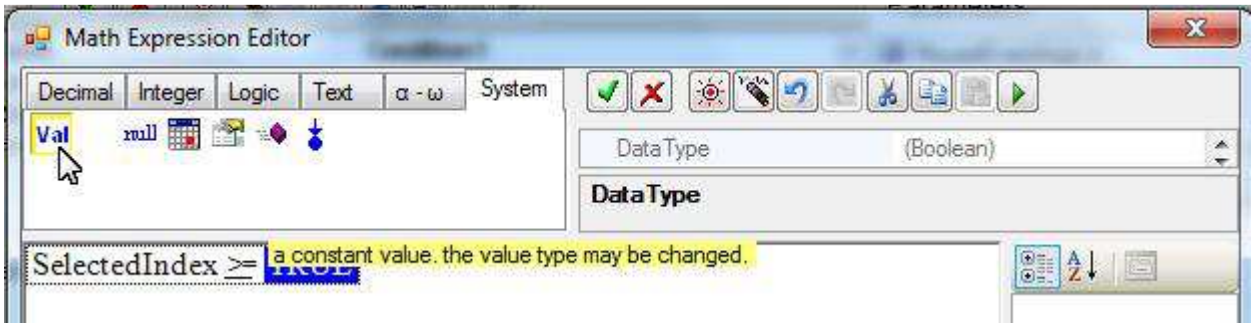


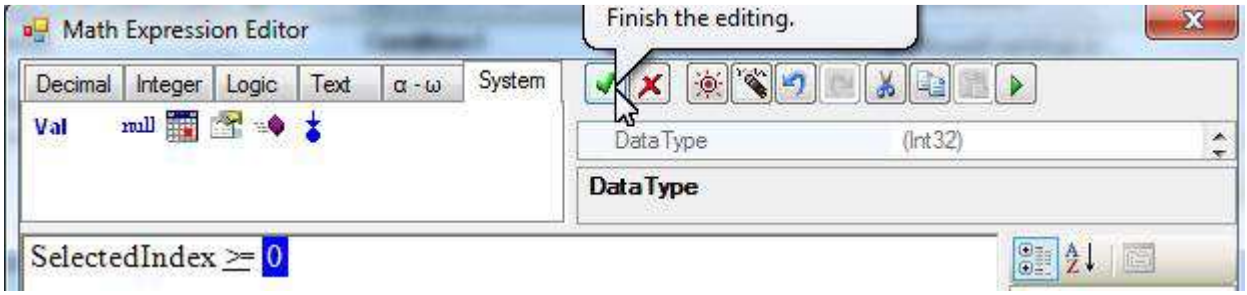
We want to check whether SelectedIndex property is larger than or equals to 0:



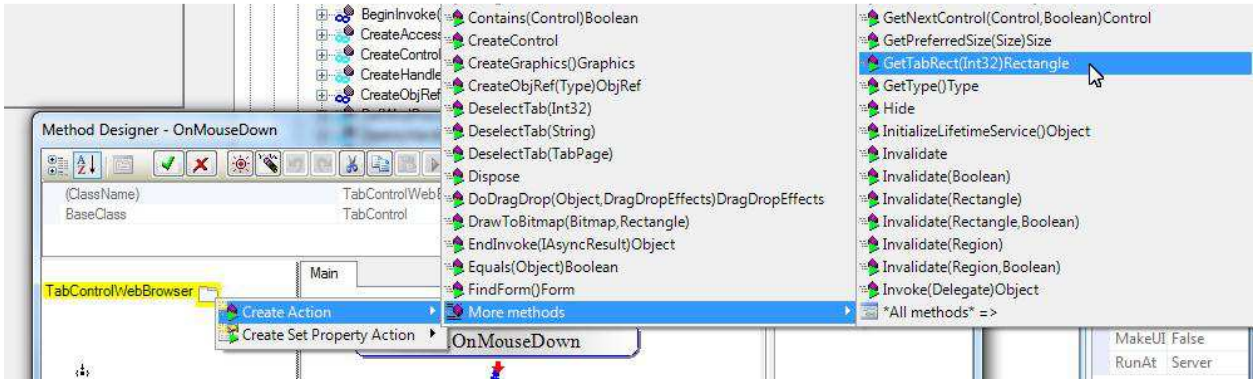




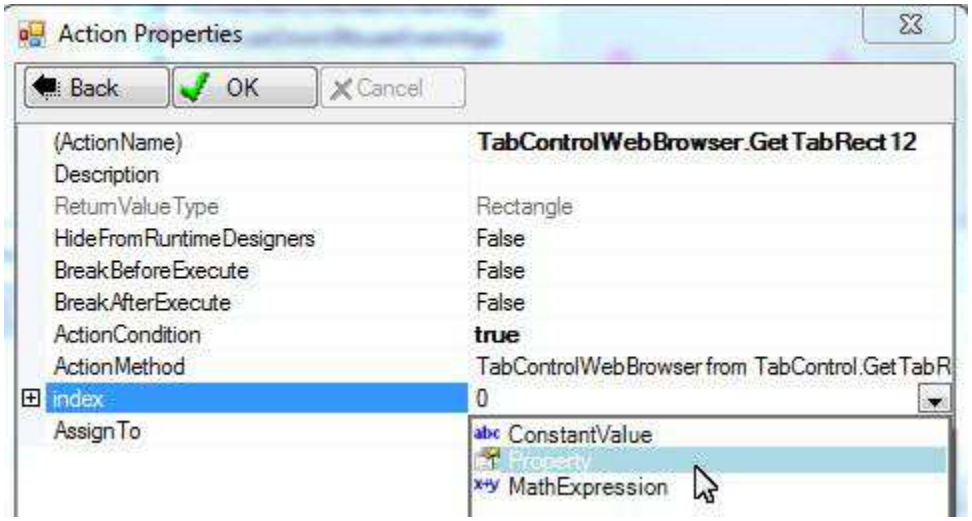


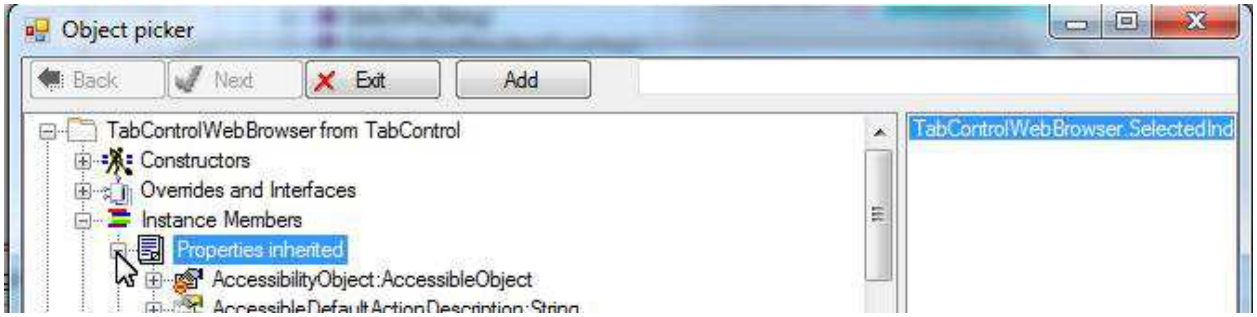


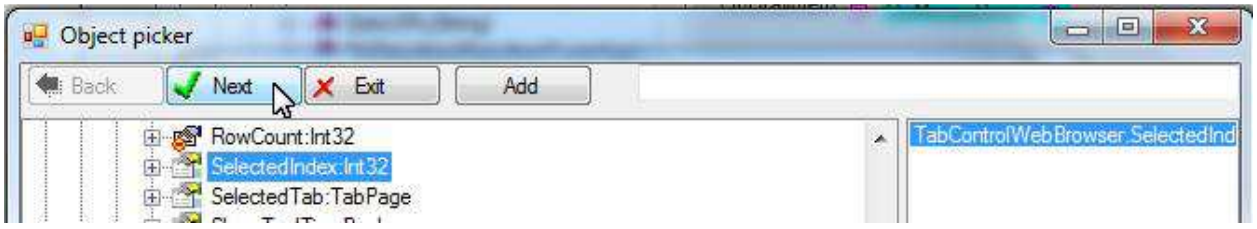
Get caption rectangle so that we may know where the red X is located:



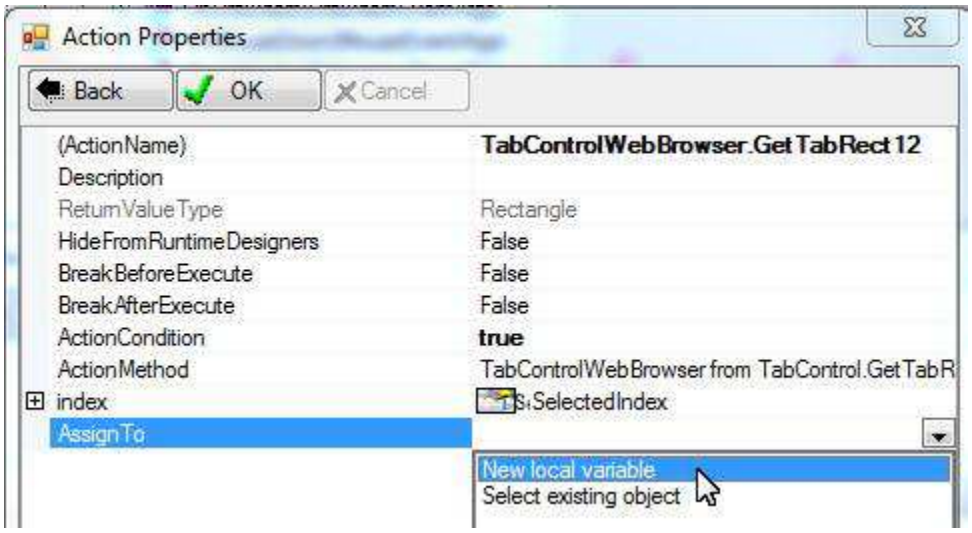
Pass SelectedIndex to the action:



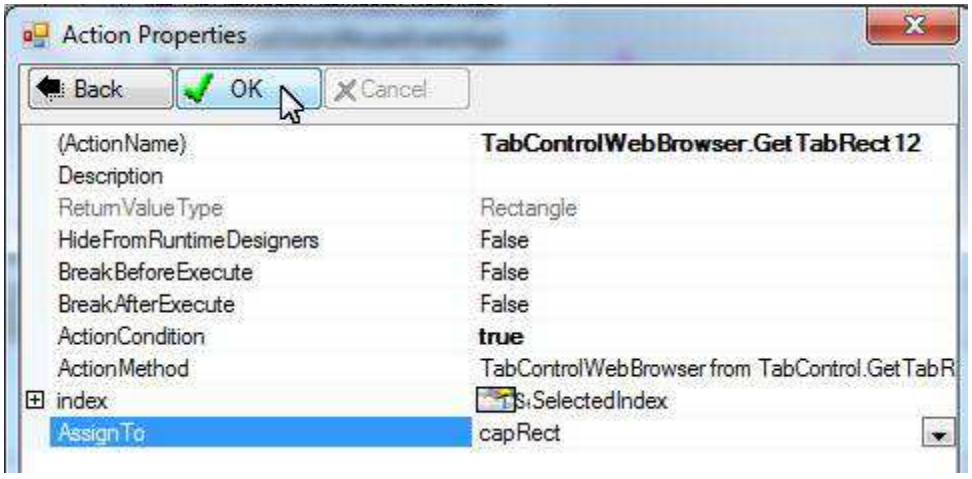




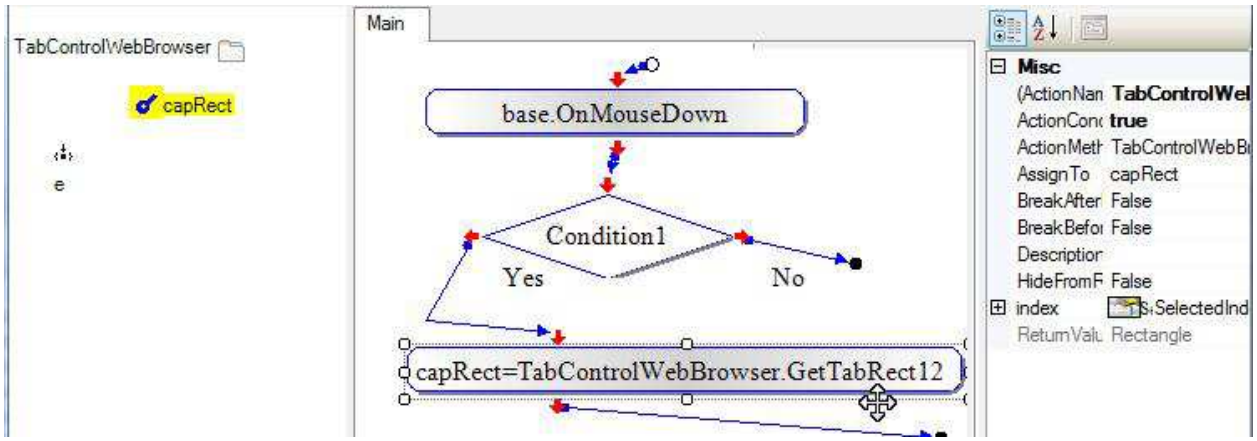
Use a variable to receive the rectangle:



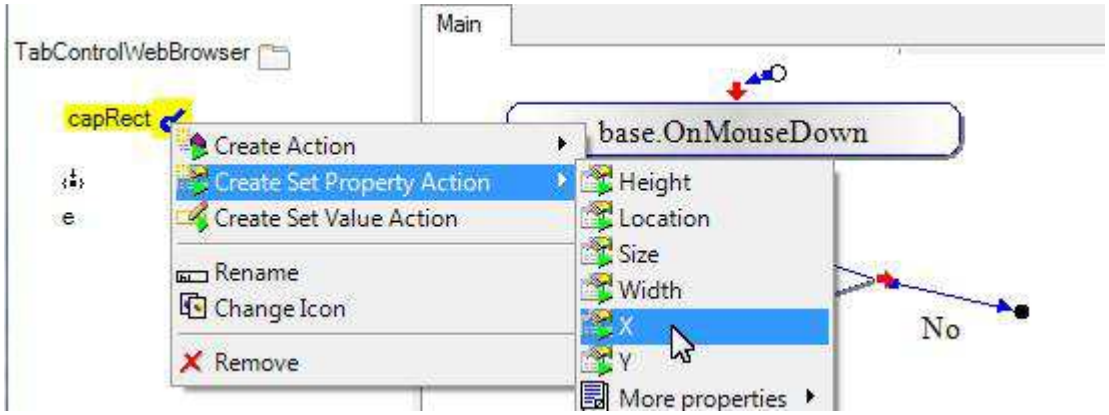
Click OK:

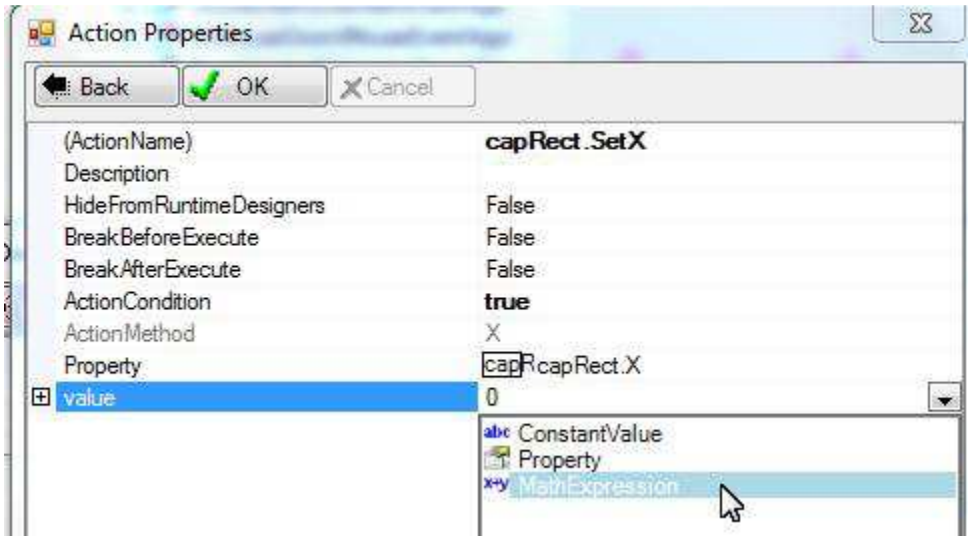


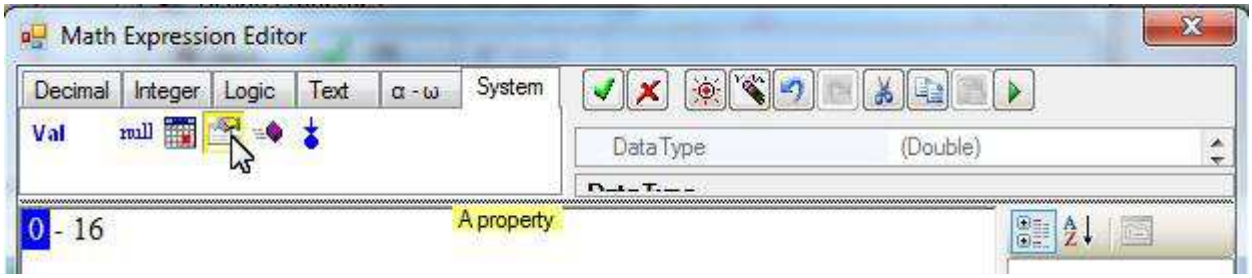
Link it to the “Yes” port:

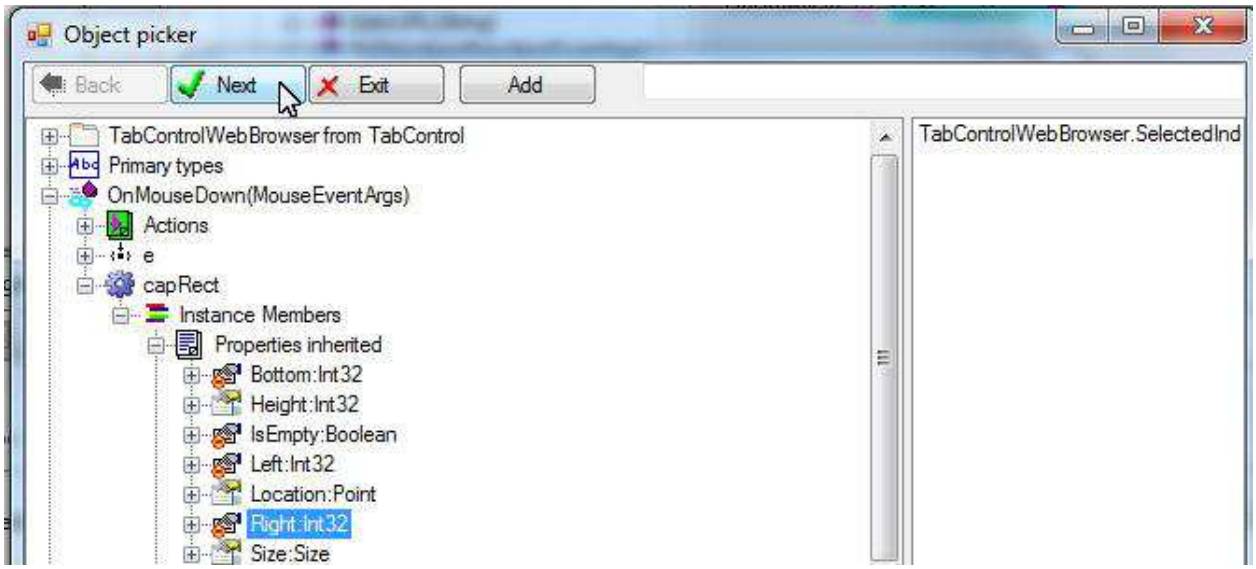


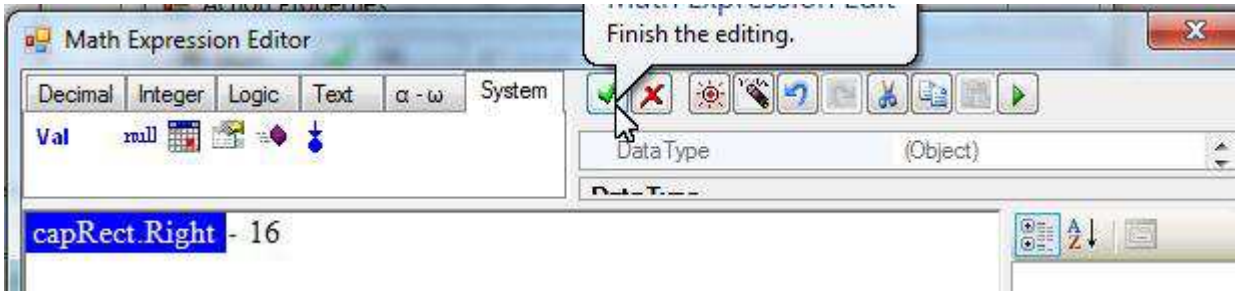
From rectangle capRect, we may find out where the red X is. We know that we only use 16 as the width when we draw red X. So, we move the Left side of capRect to its right side minus 16:

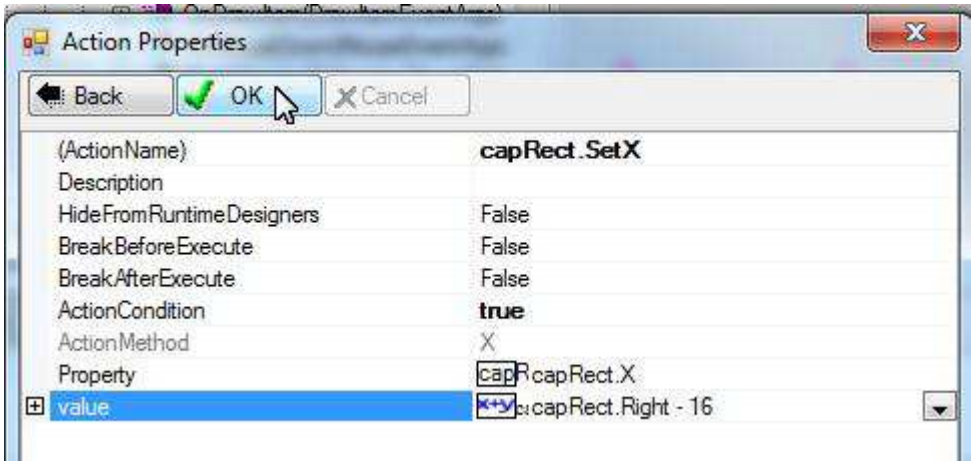




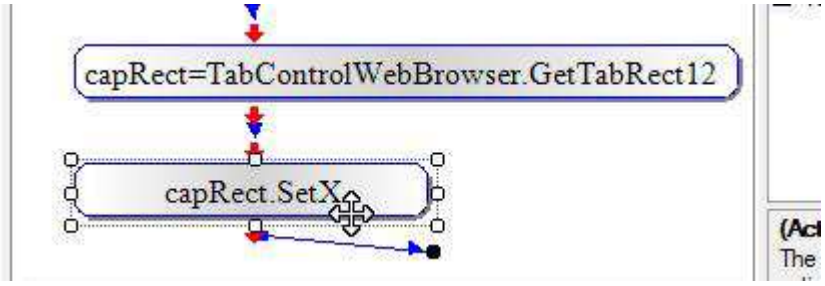

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(Double)
0 - 16
A property


Object picker
Back
Next
Exit
Add
TabControlWebBrowser from TabControl
Primary types
OnMouseDown(MouseEventArgs)
Actions
e
capRect
Instance Members
Properties inherited
Bottom:Int32
Height:Int32
IsEmpty:Boolean
Left:Int32
Location:Point
Right:Int32
Size:Size
TabControlWebBrowser.SelectedInd

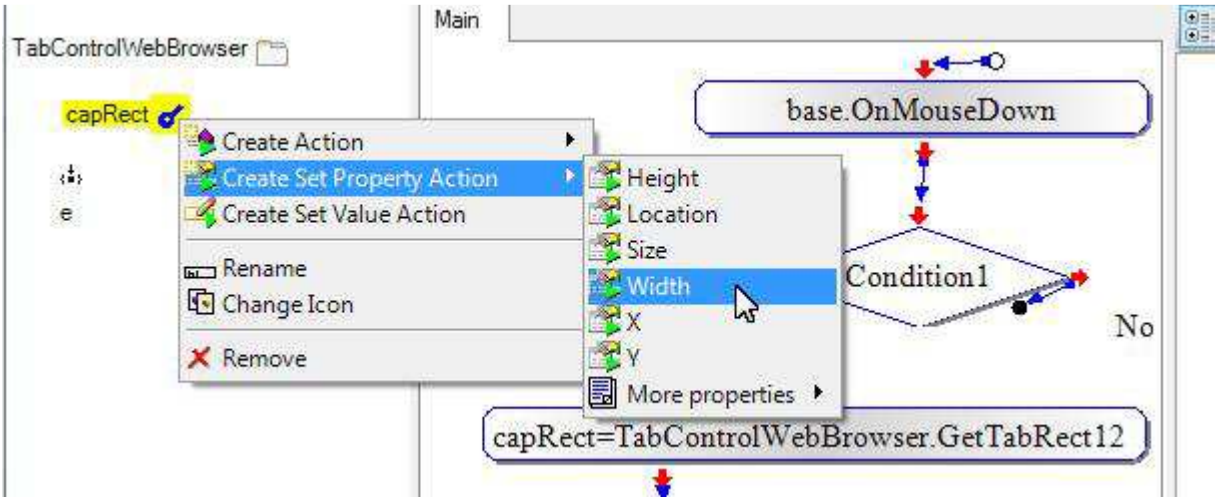

Math Expression Editor
Finish the editing.
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(Object)
capRect.Right - 16

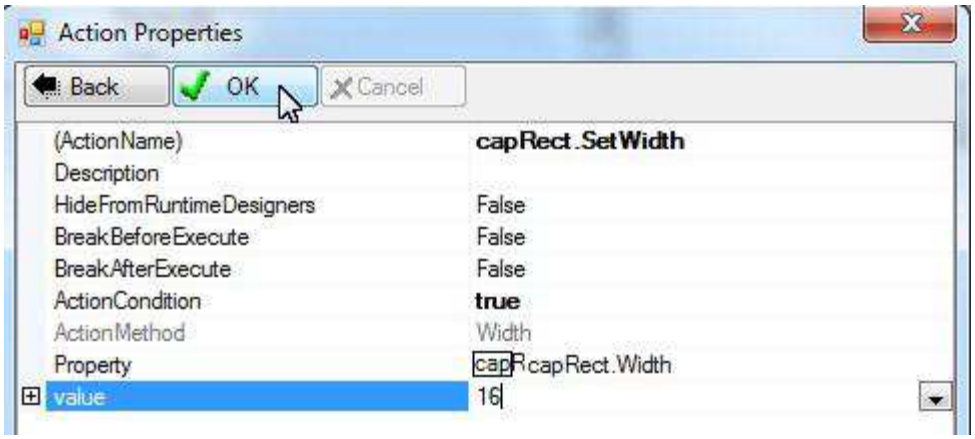

Action Properties
Back
OK
Cancel
(ActionName)
capRect.SetX
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
X
Property
capRect.X
value
capRect.Right - 16

Link it to the last action:

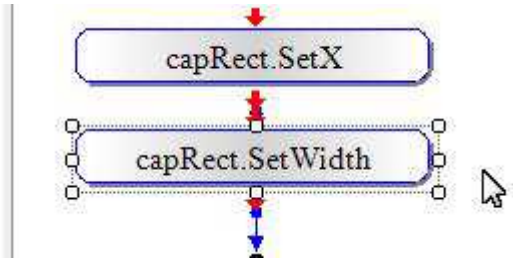


Change the Width of capRect to 16:

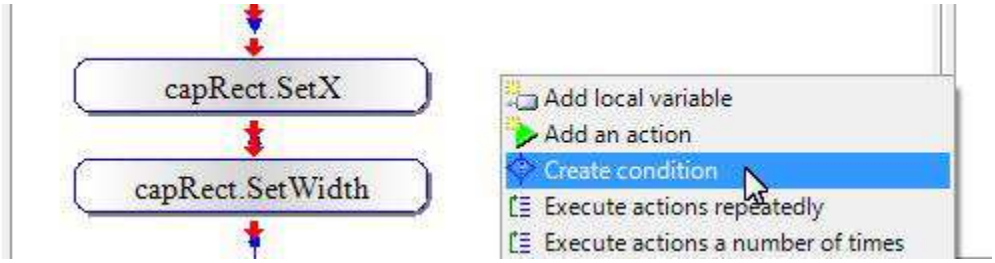




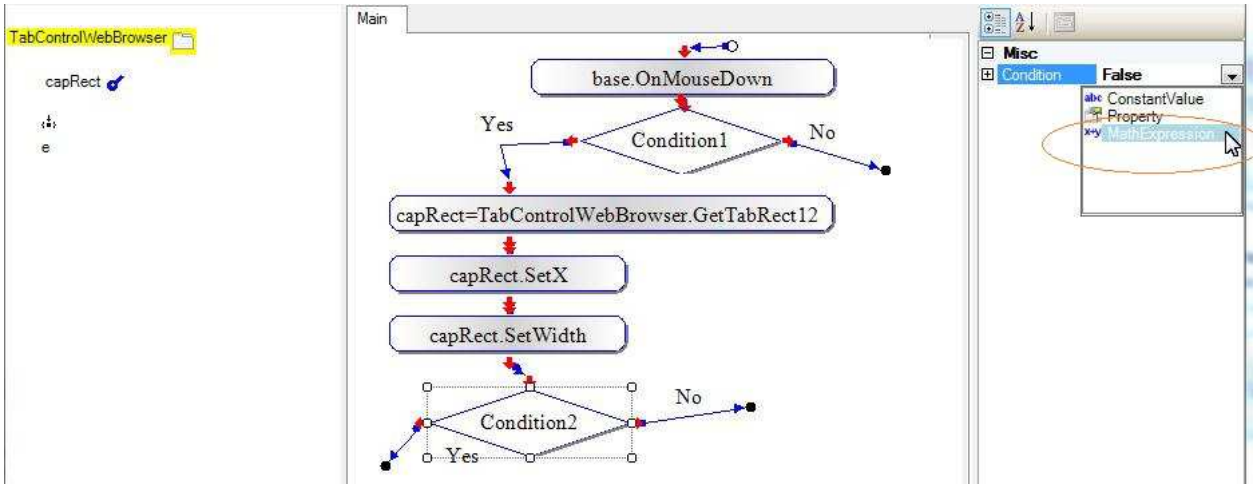
Link it to the last action:

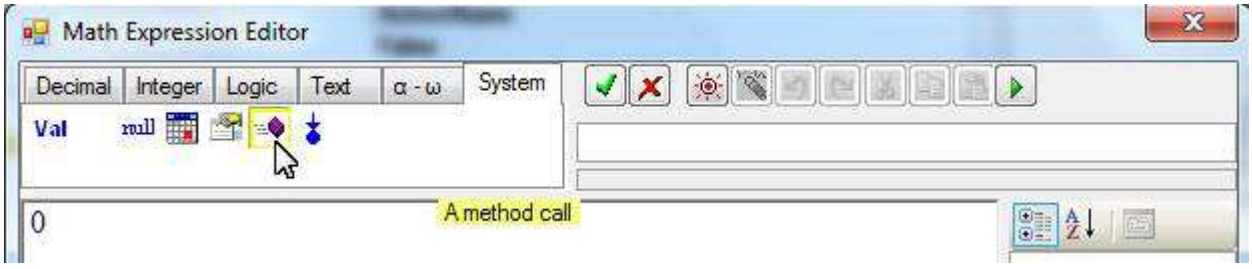


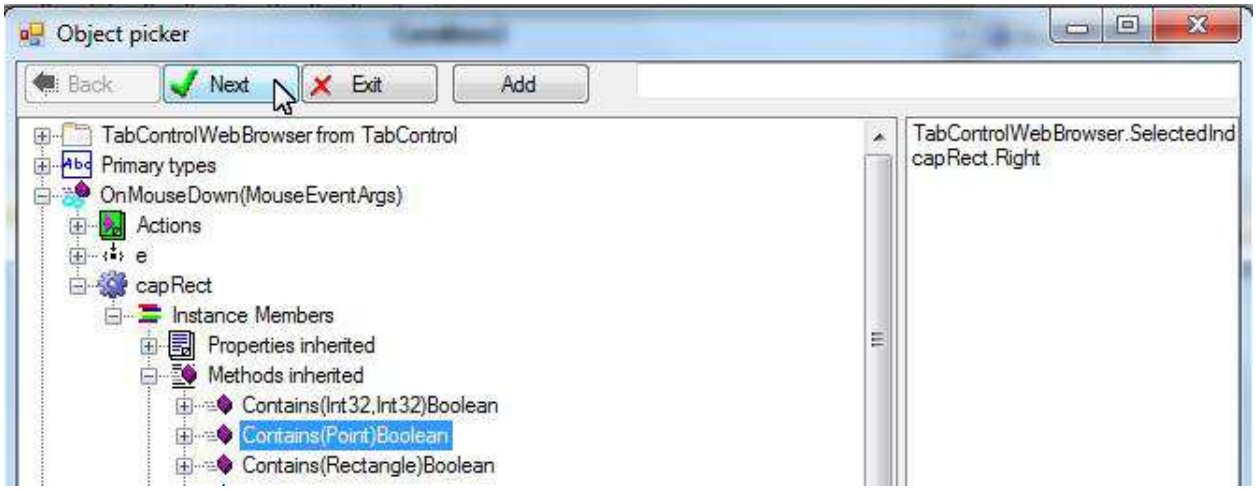
Now capRect is a rectangle covering the red X. We may add a Condition action to check whether the mouse pointer is within the rectangle:

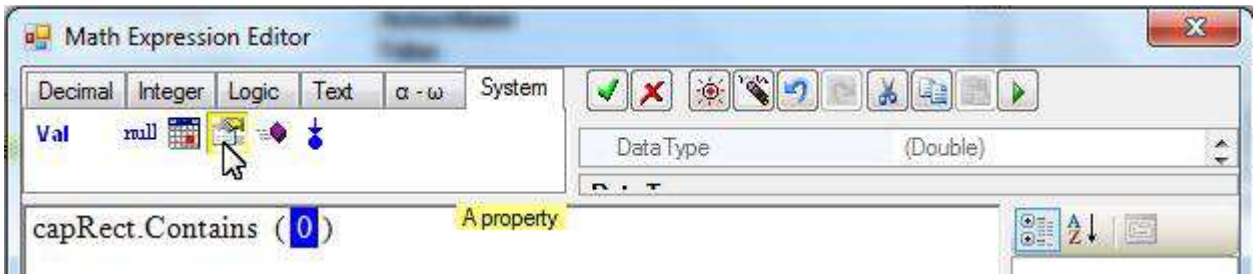


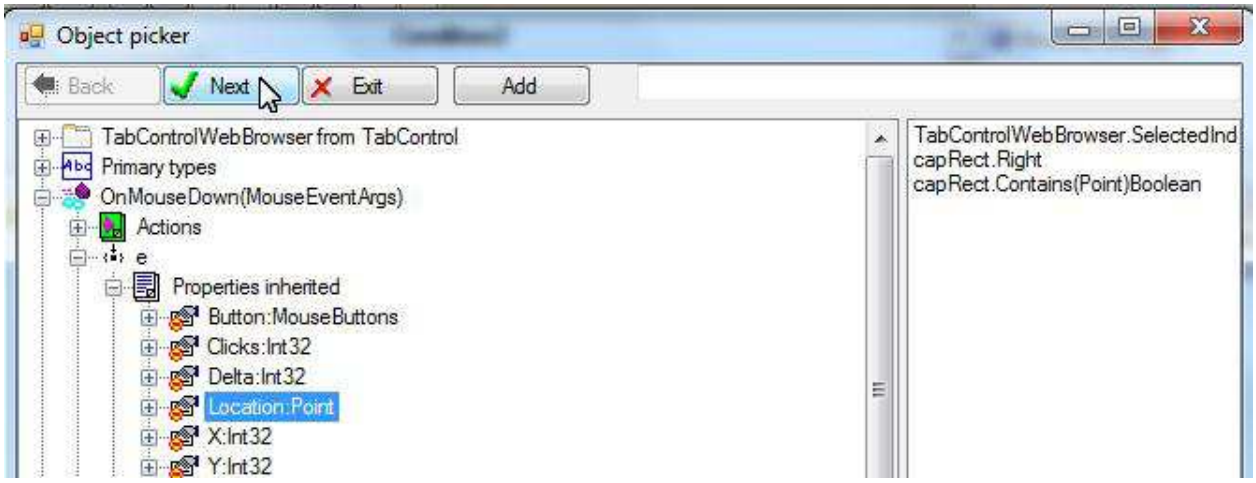
Link it to the last action. Set its Condition to check whether the mouse pointer is inside the rectangle:

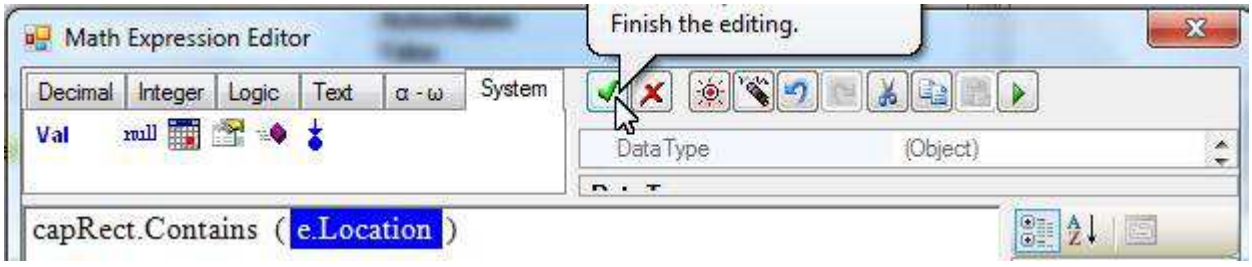






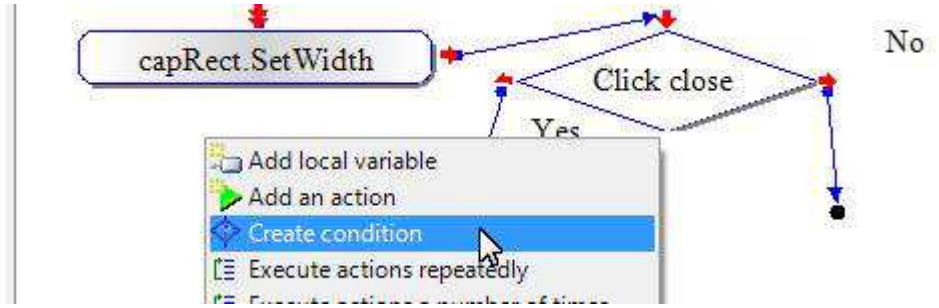




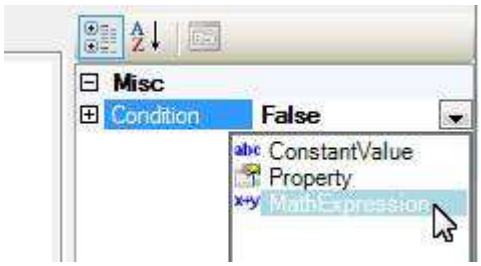


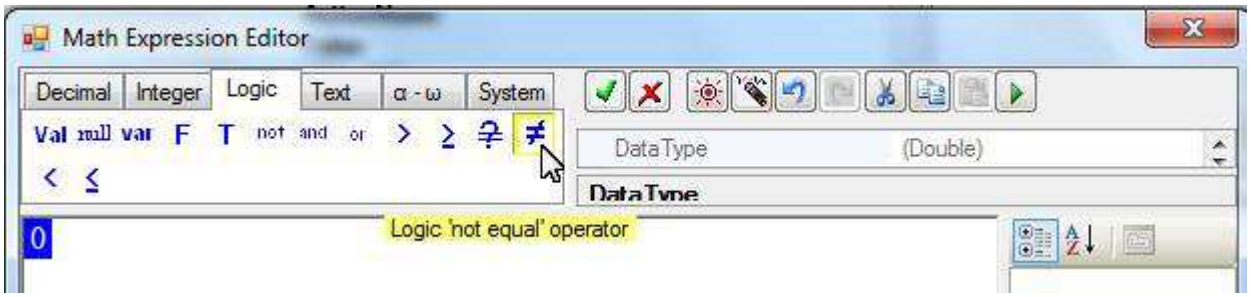
If the condition is True then we may remove the tab page. Before doing that, we may fire event BeforeRemoveTabPage to give the user a chance to cancel the operation.

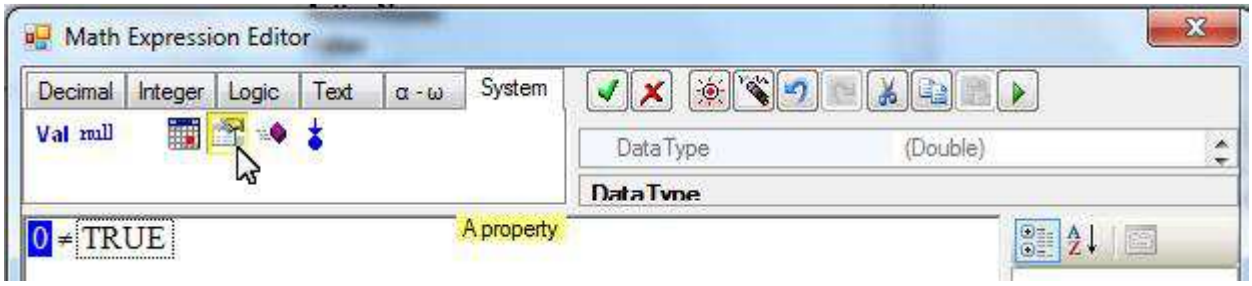
We fire the event only if BeforeRemoveTabPage is not null. Add a Condition to check it:

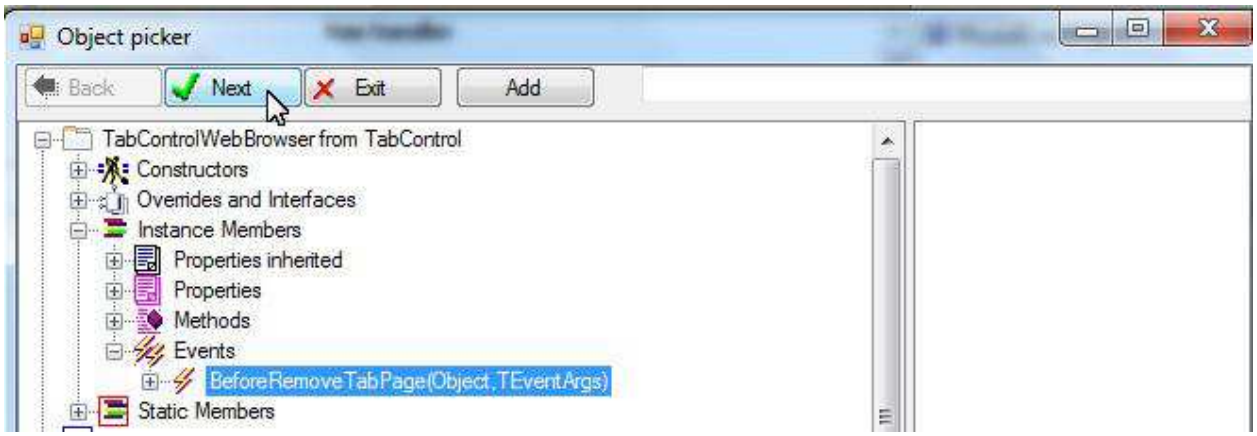


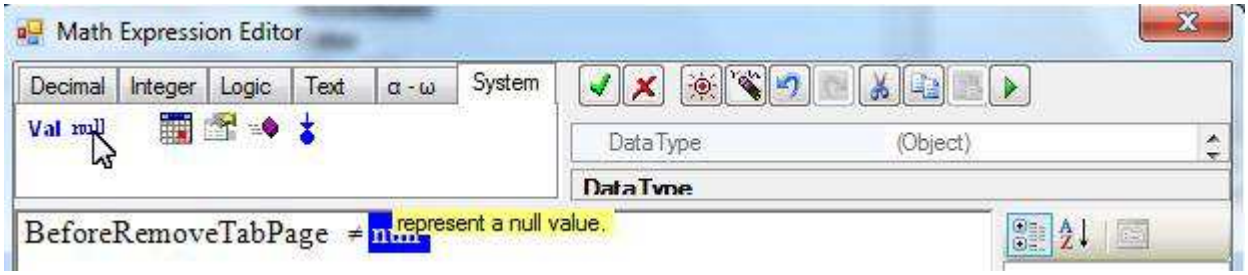
Set its Condition to check BeforeRemoveTabPage:



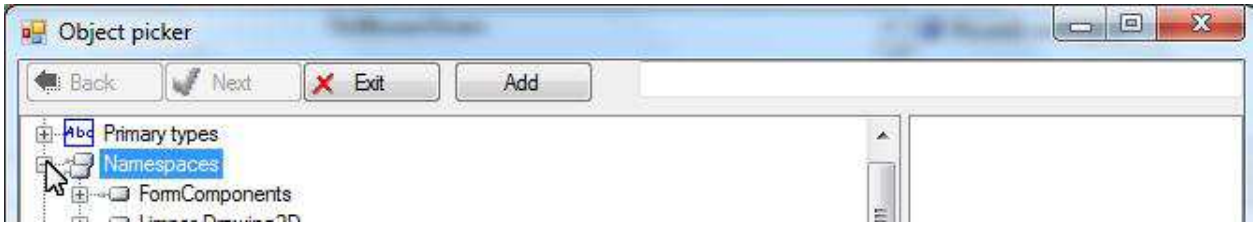


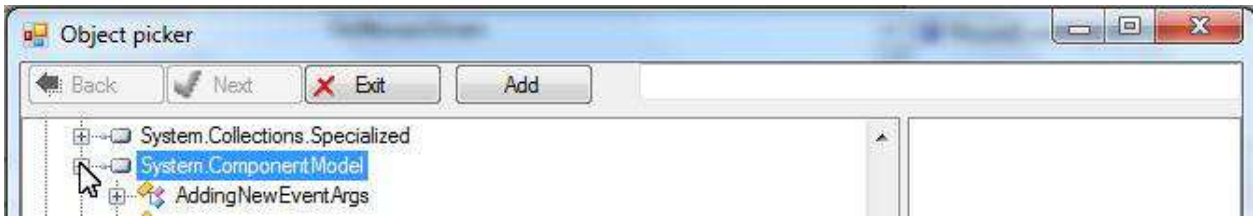


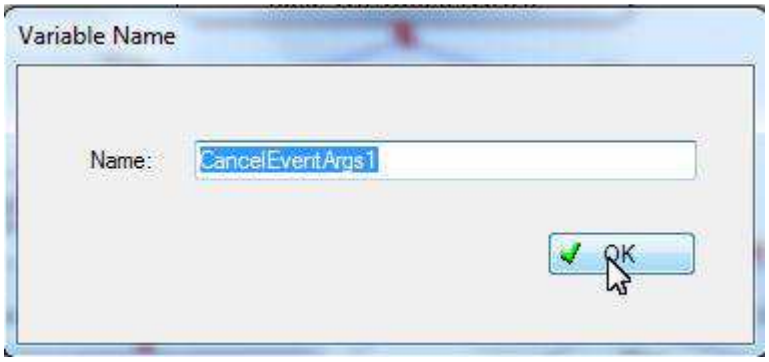


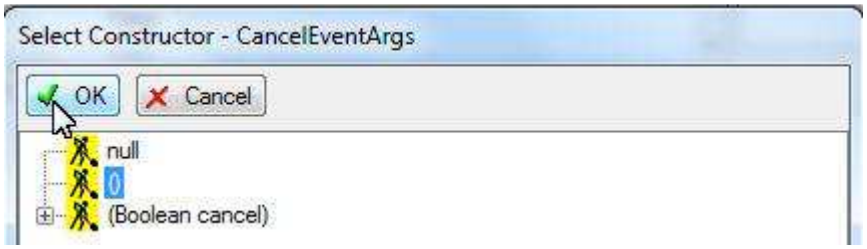


Create a CancelEventArgs variable to receive user cancelation signal:

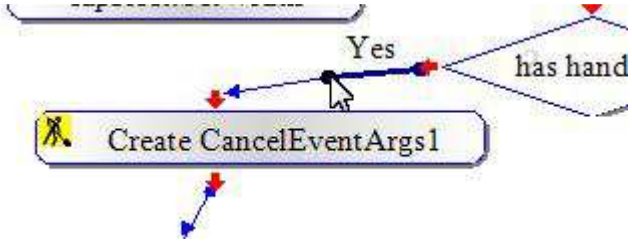




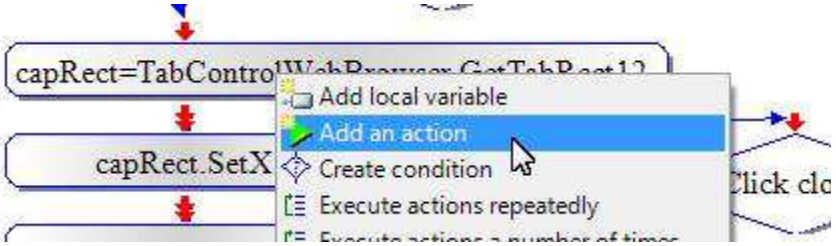


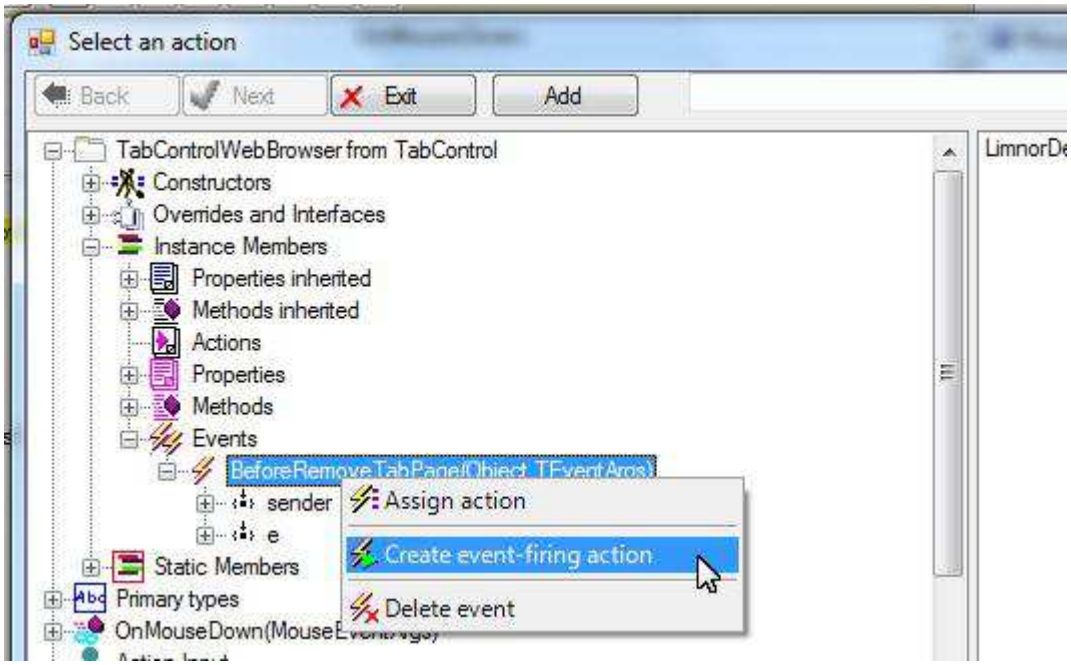


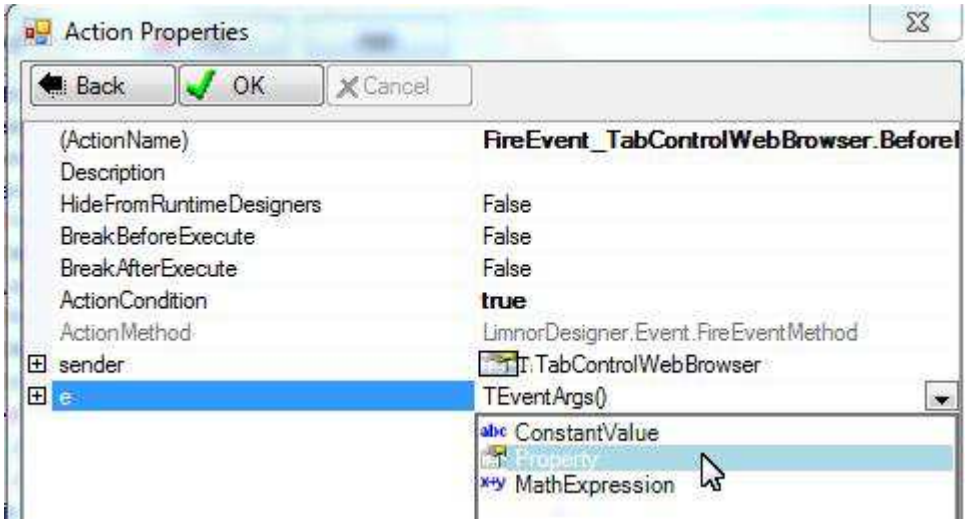
Link it to Yes:

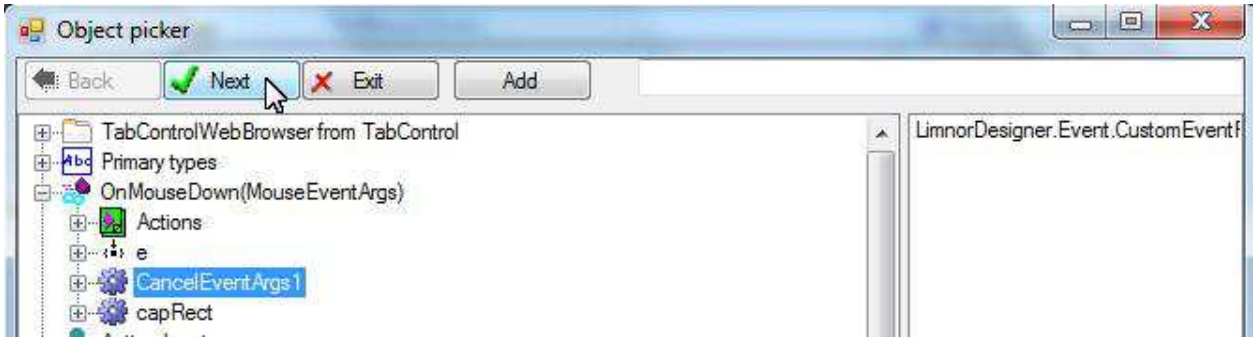


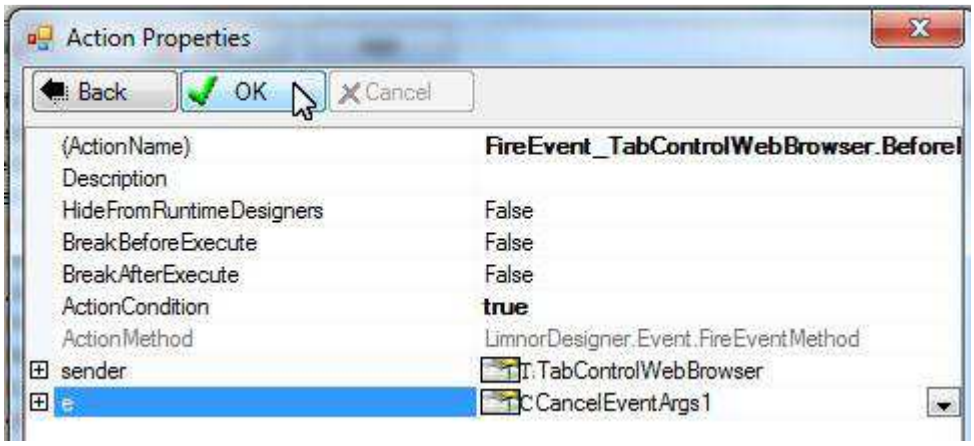
Create an action for fire event:

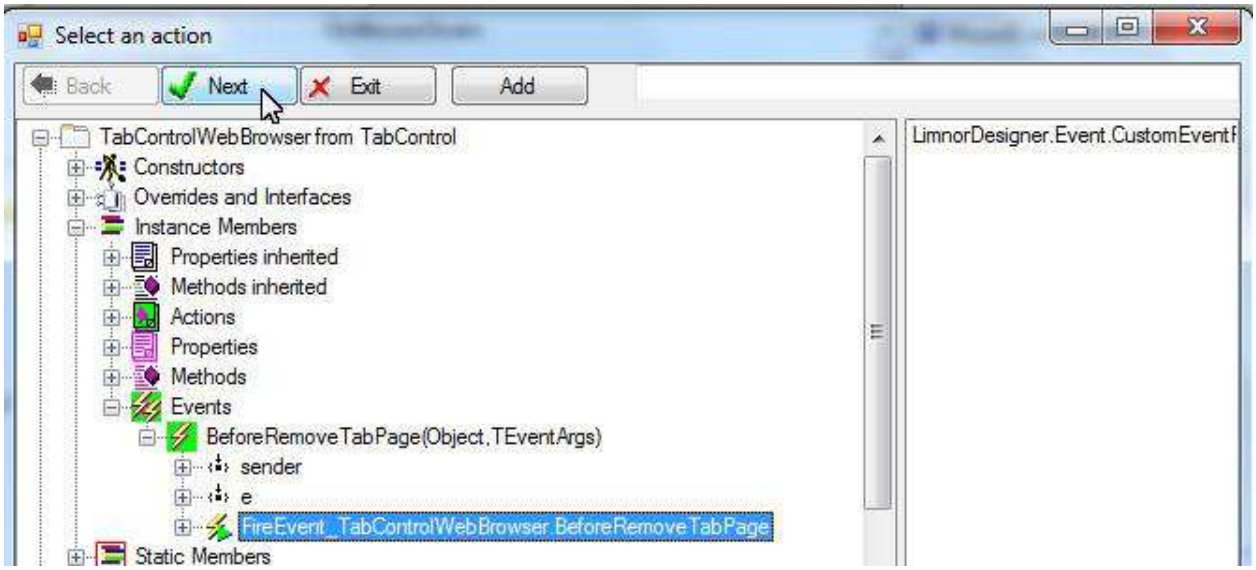

capRect=TabContro
capRect.SetX
Add local variable
Add an action
Create condition
Execute actions repeatedly


Select an action
Back
Next
Exit
Add
TabControlWebBrowser from TabControl
Constructors
Overrides and Interfaces
Instance Members
Properties inherited
Methods inherited
Actions
Properties
Methods
Events
sender
e
Static Members
Primary types
Assign action
Create event-firing action
Delete event

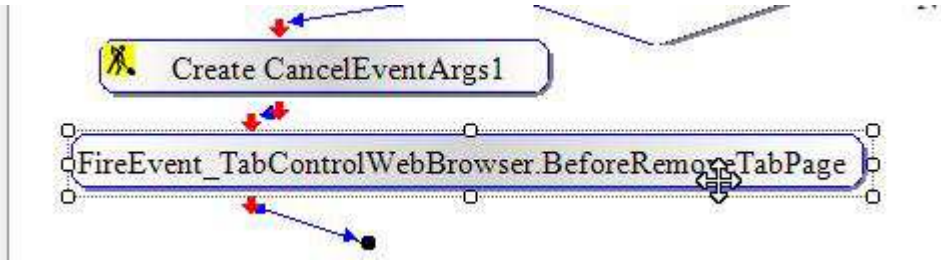

Action Properties
Back
OK
Cancel
(ActionName)
Description
HideFromRuntimeDesigners
BreakBeforeExecute
BreakAfterExecute
ActionCondition
ActionMethod
sender
e
False
False
False
true
LimnorDesigner.Event.FireEventMethod
TabControlWebBrowser
TEventArgs()
ConstantValue
Property
MathExpression



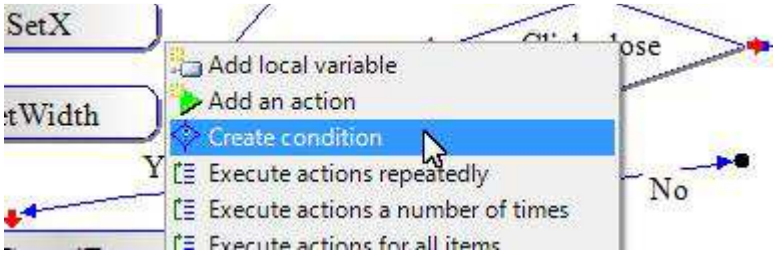




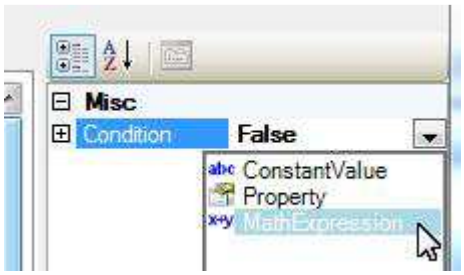
Link it to the last action:

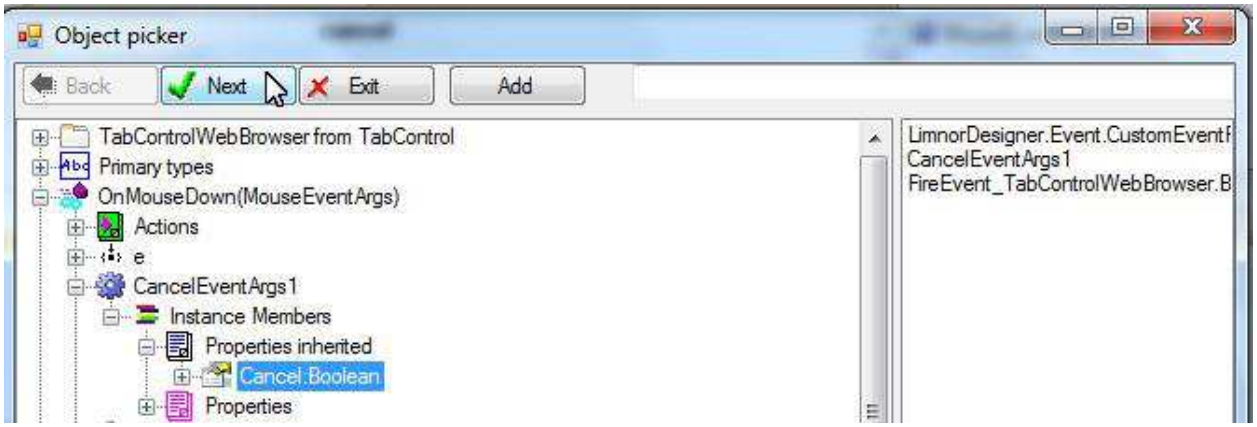


Add a Condition to check the cancelation:

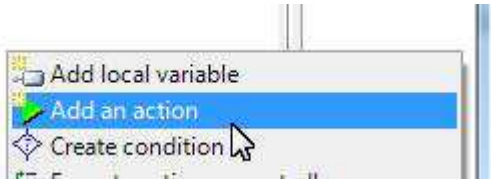


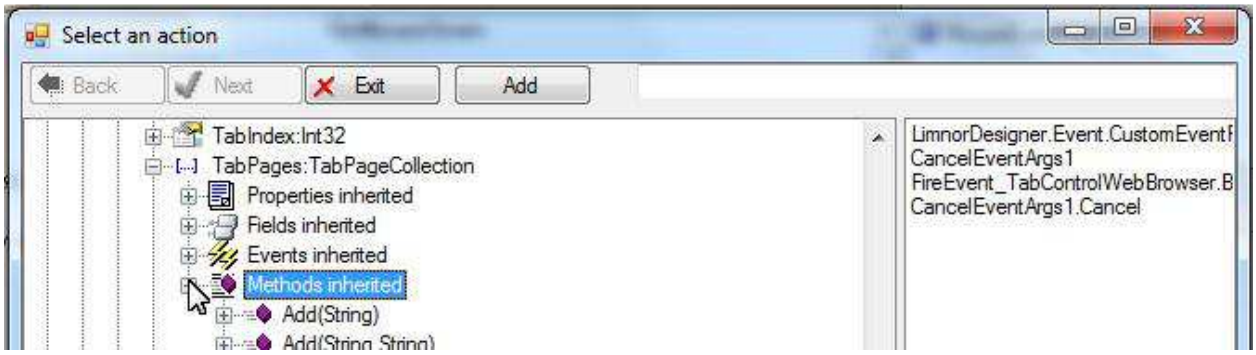
Set the Condition to the Cancel property of the event parameter:

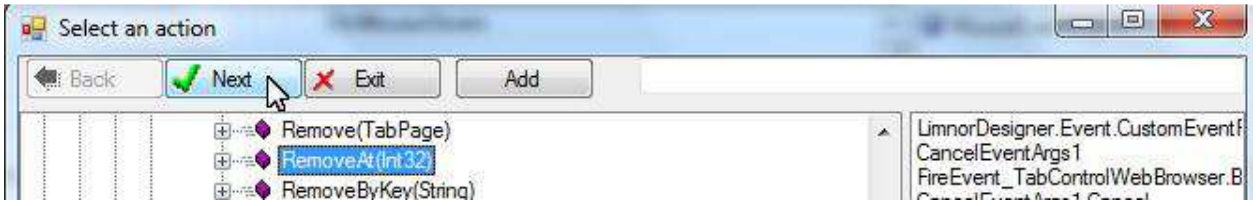




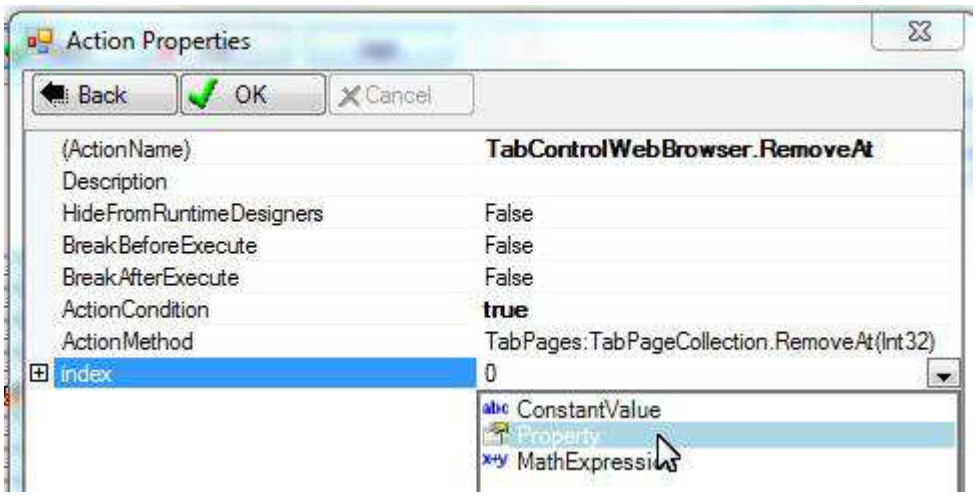
Create an action to remove the tab page:

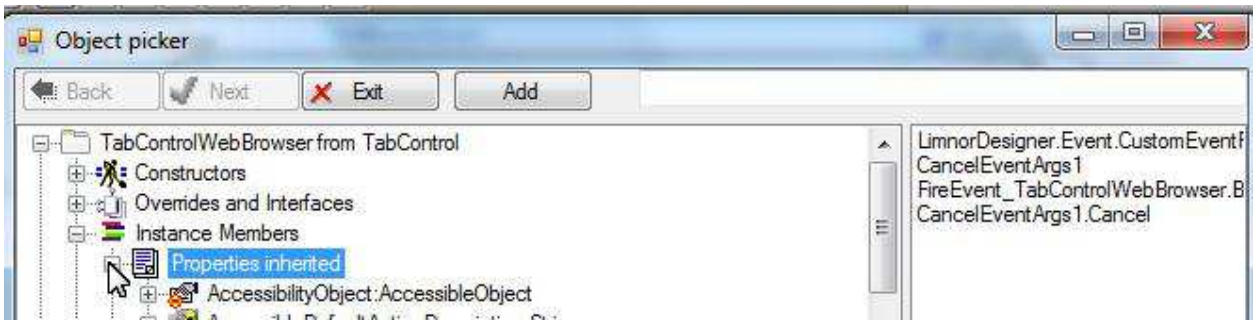


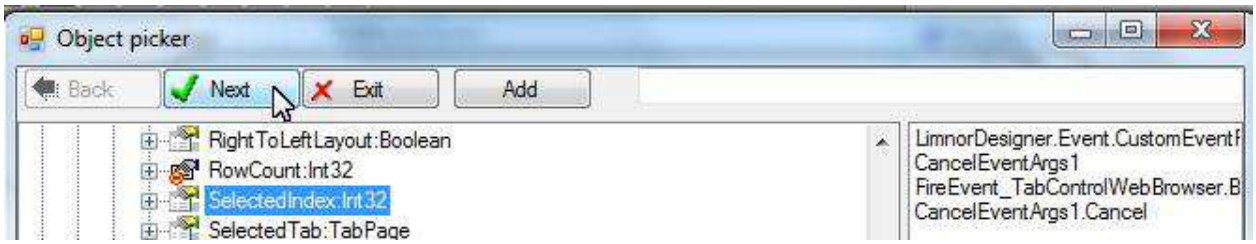


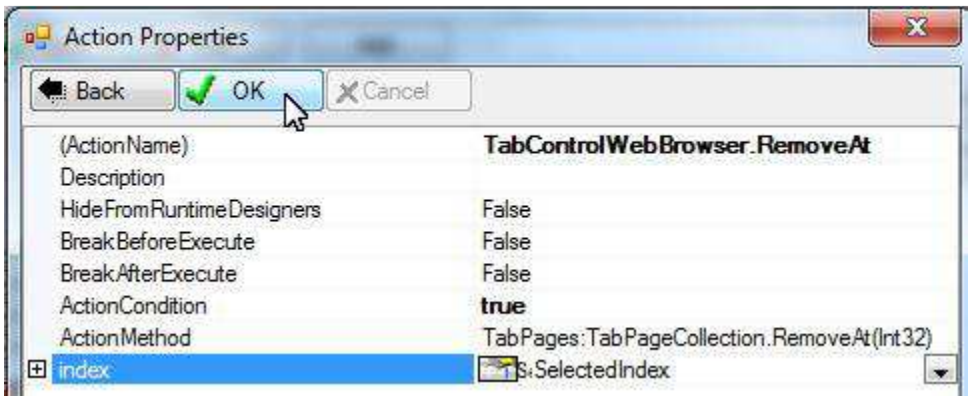


Pass SelectedIndex property to the action:

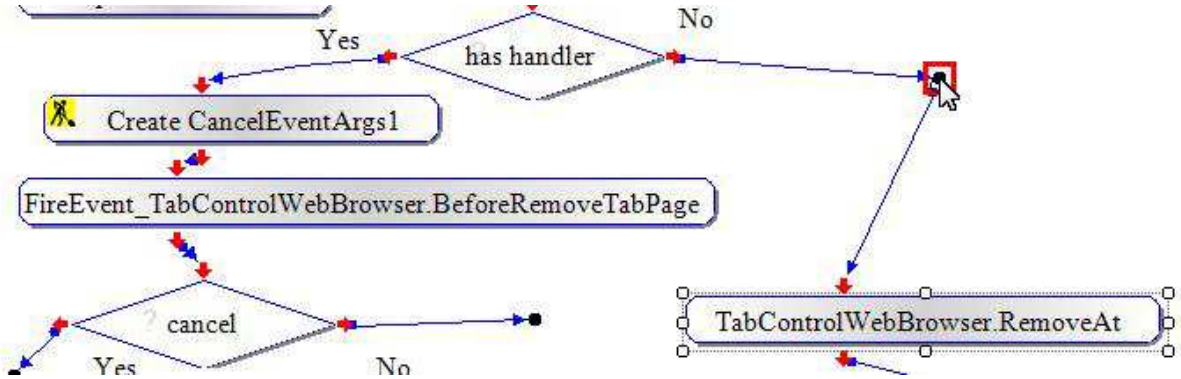




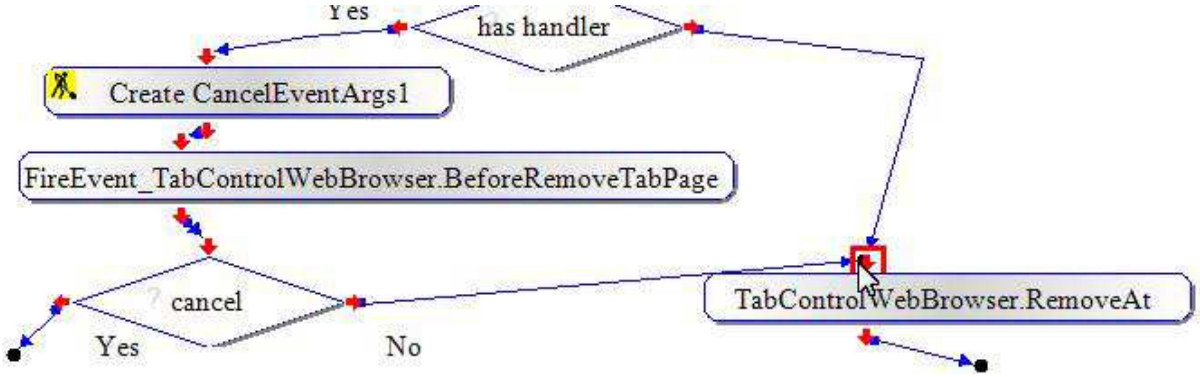




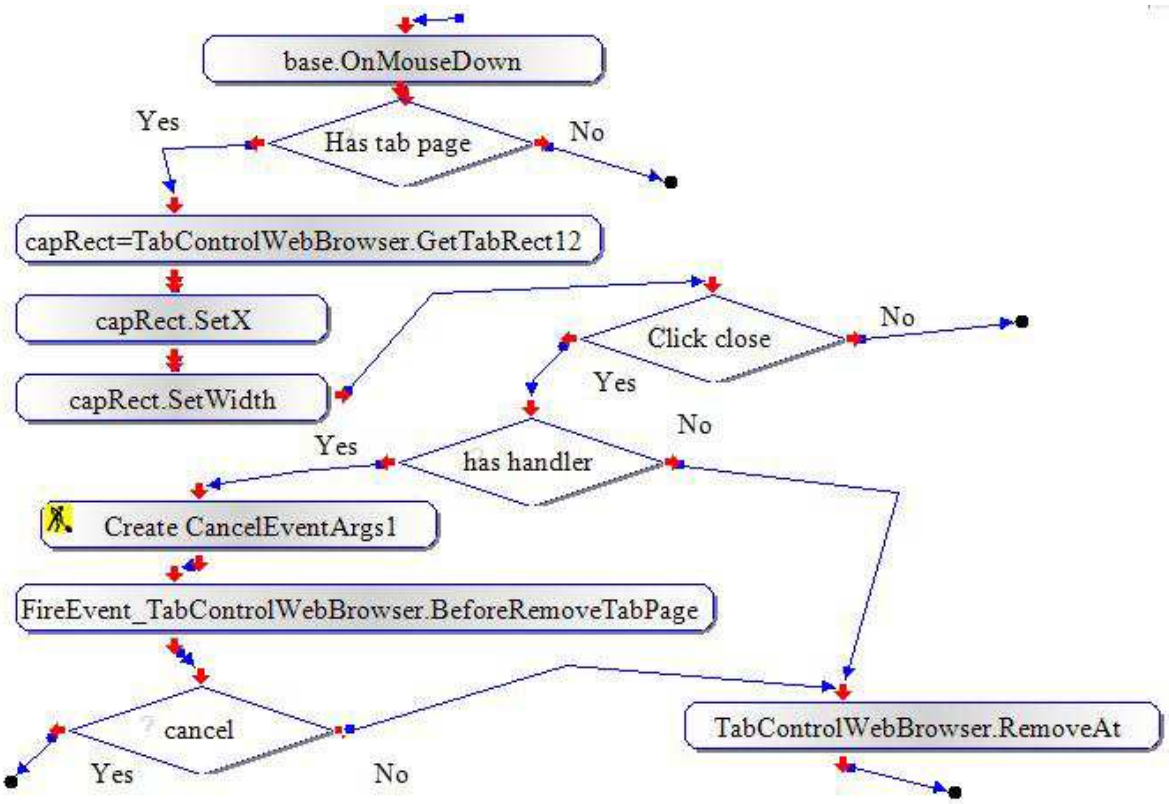
Link the action to “No” port of handler checking:



Also link it to “No” port of cancelation checking:



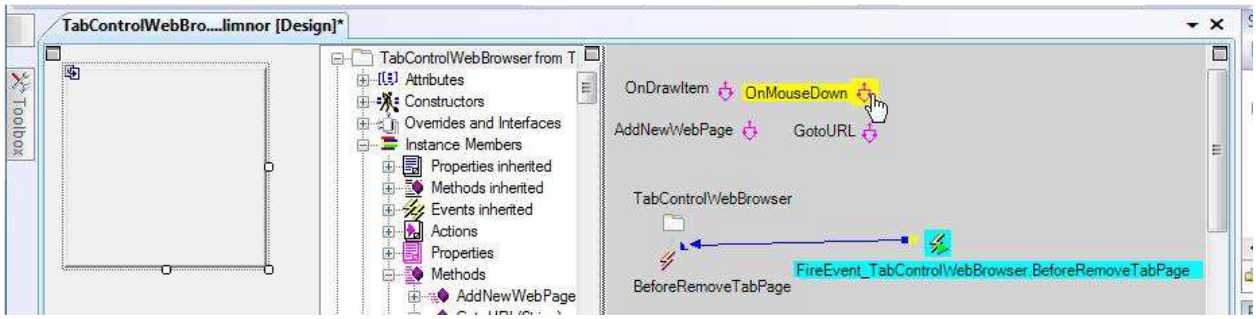
That is all for the method:



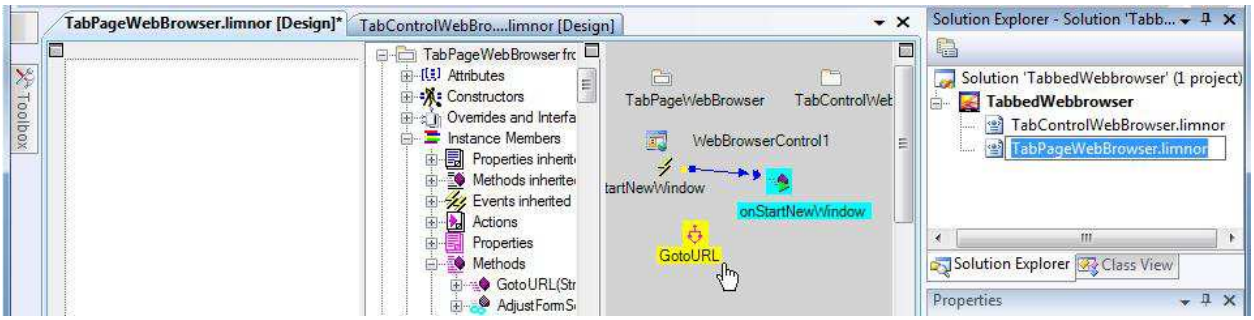
Generate DLL

Summary of what we programmed:

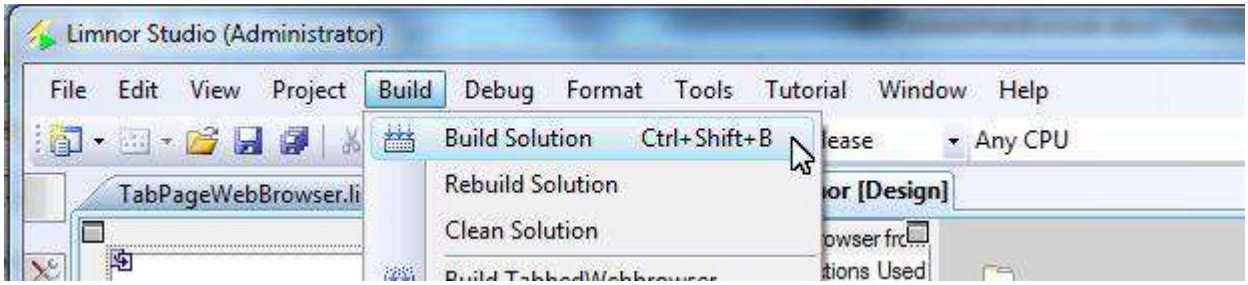
1. Created a class TabControlWebBrowser derived from TabControl. Added two methods, AddNewWebPage and GotoURL, to it for web browsing. Added OnDrawItem and OnMouseDown to provide “close” button for removing tab page.



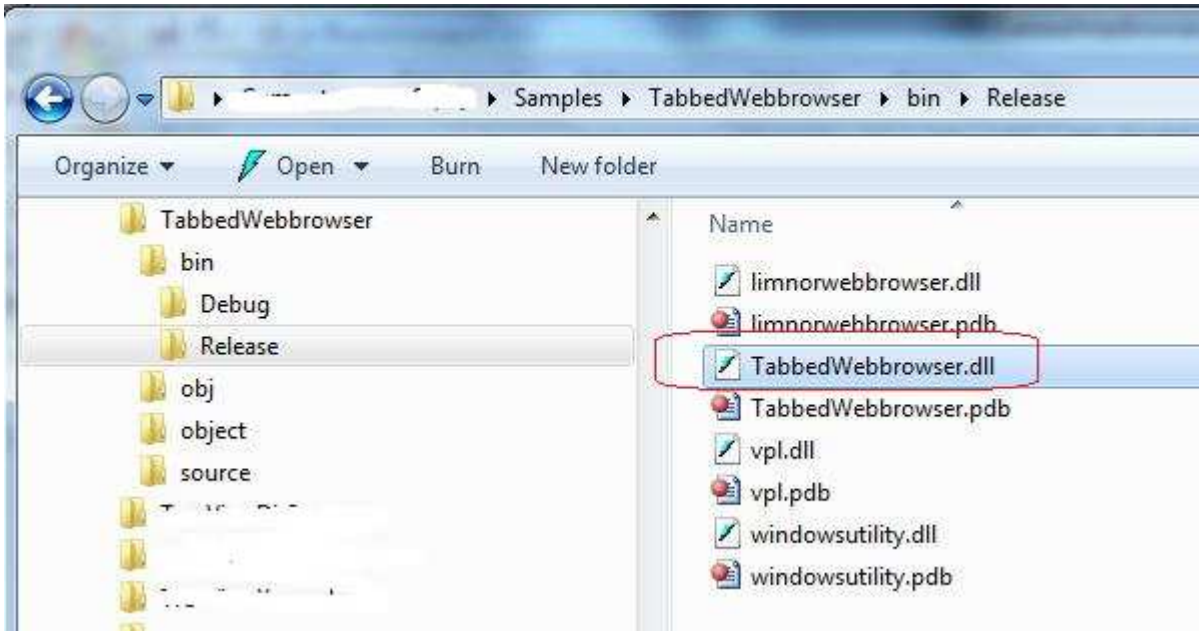
2. Created a class TabPageWebBrowser derived from TabPage. Added a method, GotoURL, to it and added an event handler, onStartNewWindow, to event StartNewWindow.



Compile the project:



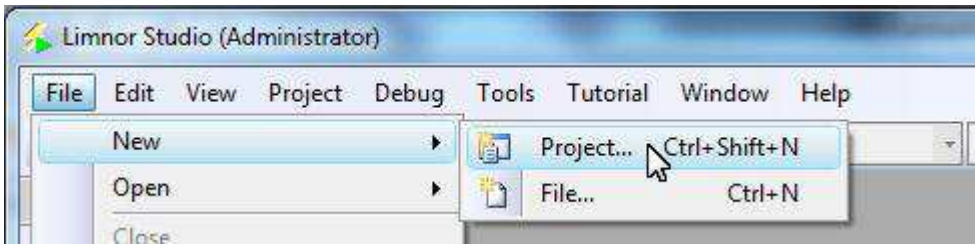
TabbedWebbroser.dll is created:



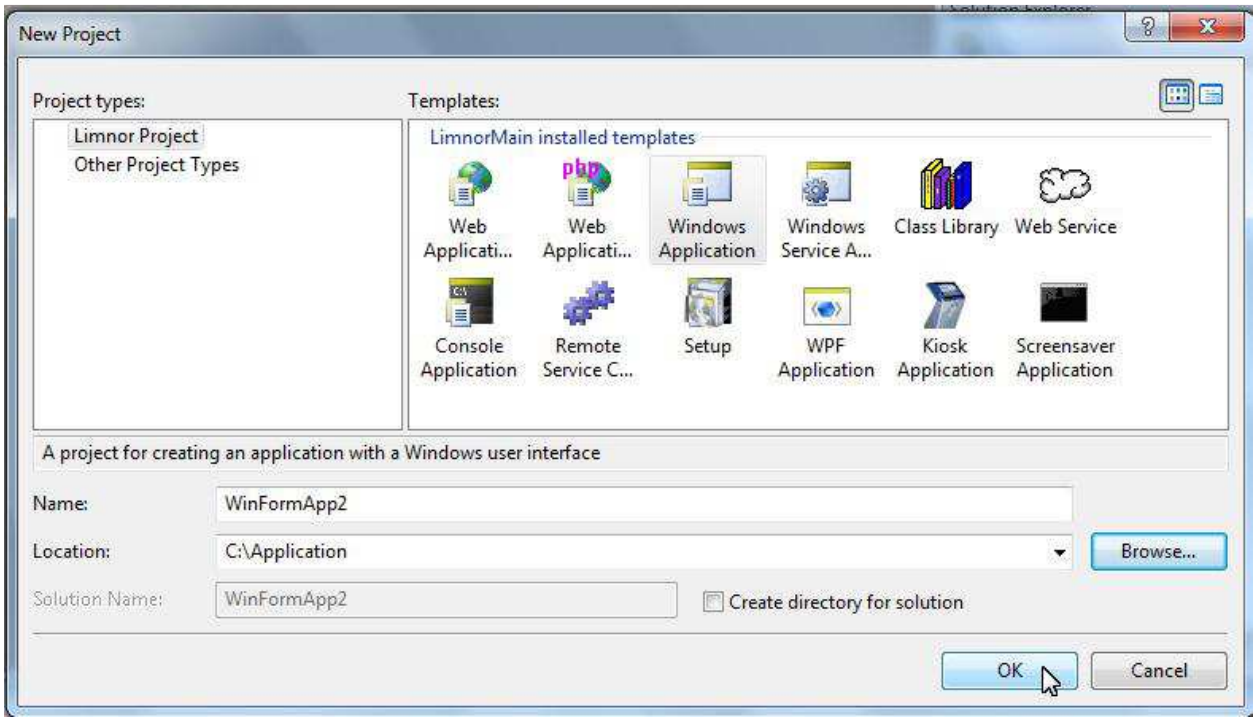
Use of TabControlWebBrowser

Create Windows Application

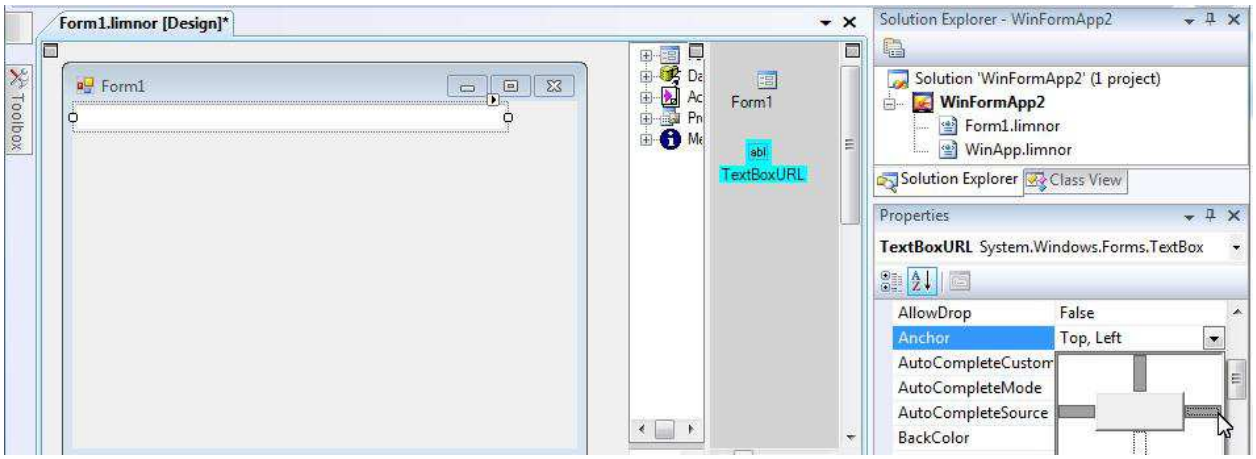
Create a new Windows Form application project:



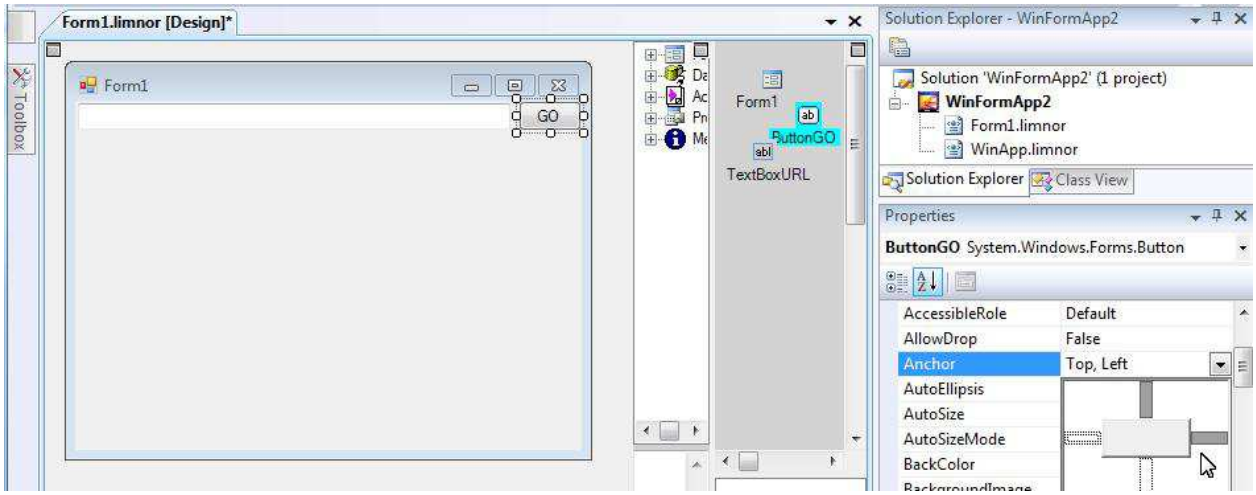
Select Windows Application:



Add a text box for entering web page address;

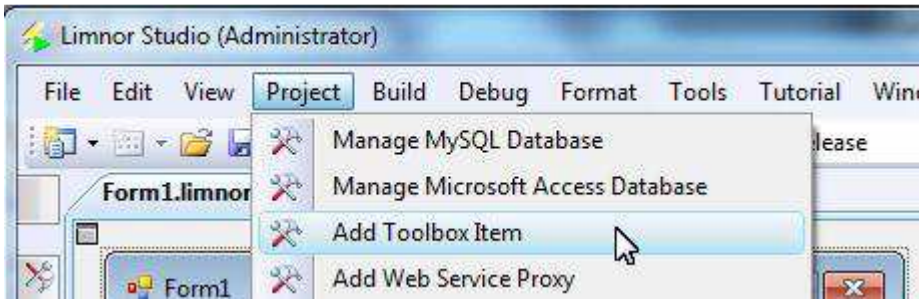


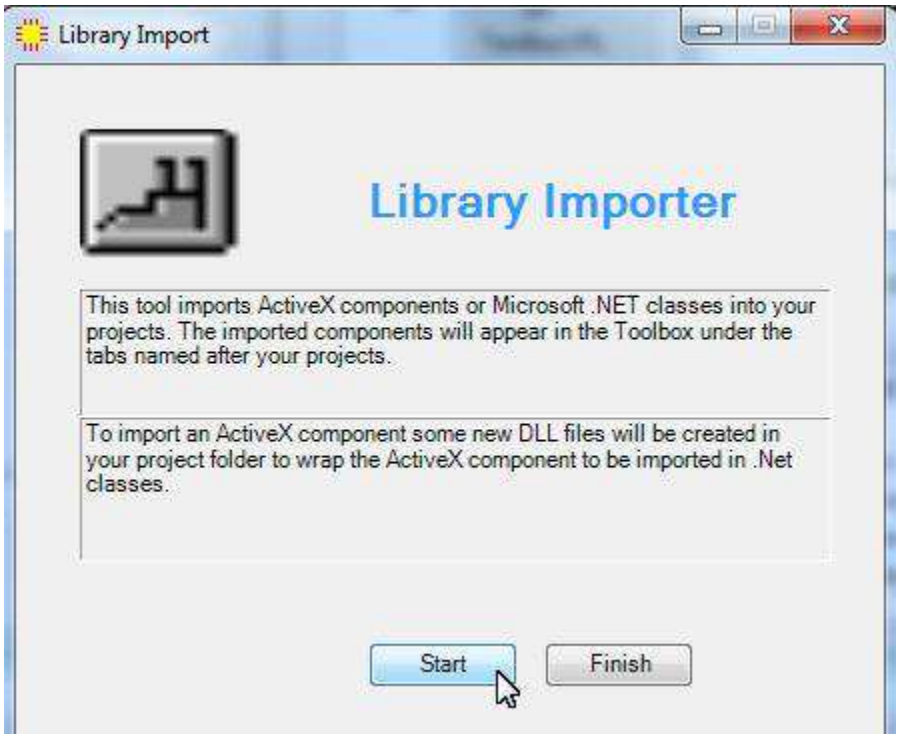
Add a button for using the web page address:

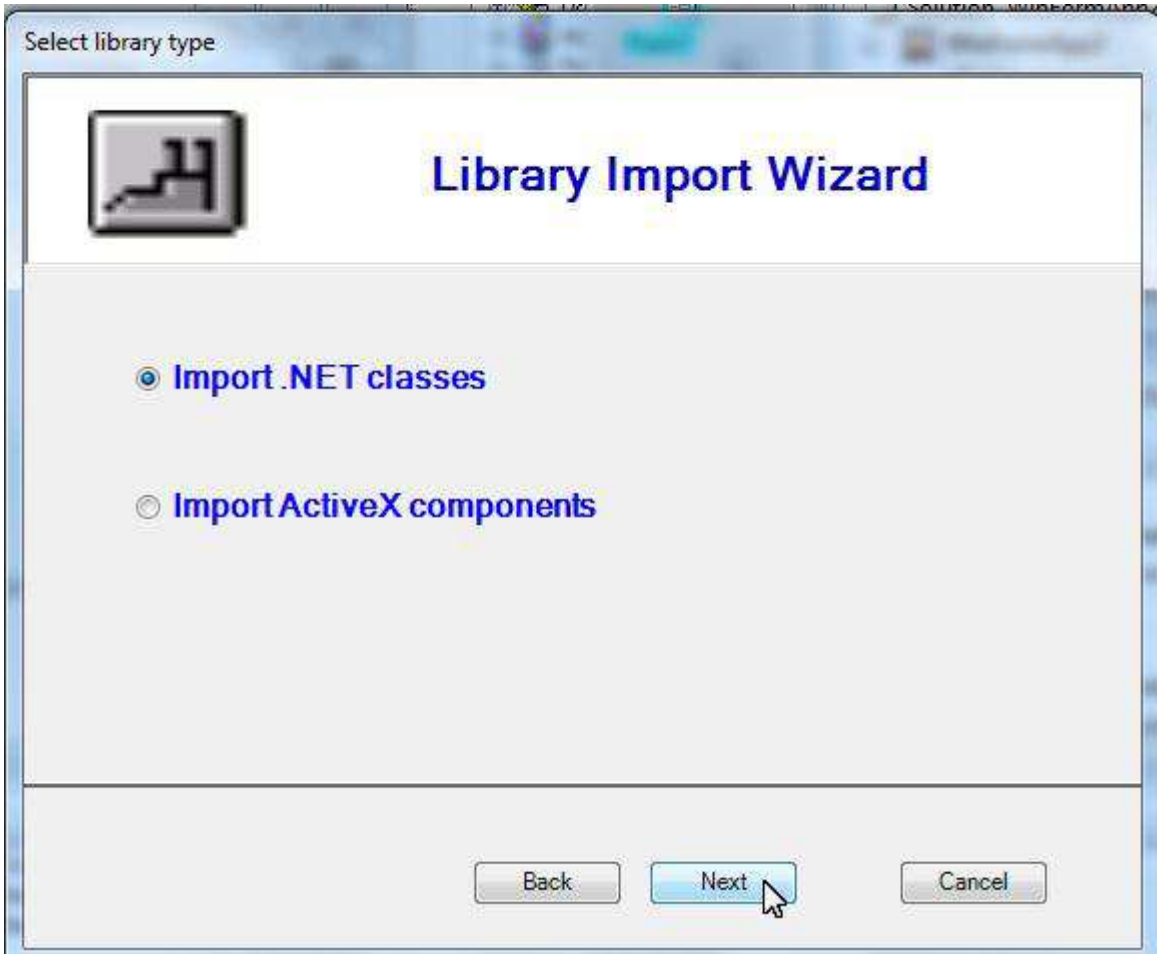


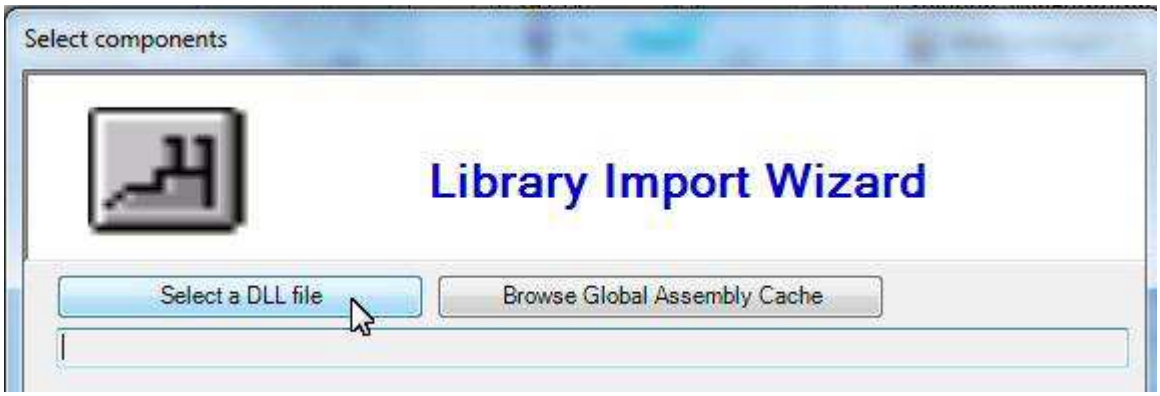
Use TabbedWebbrowser.DLL

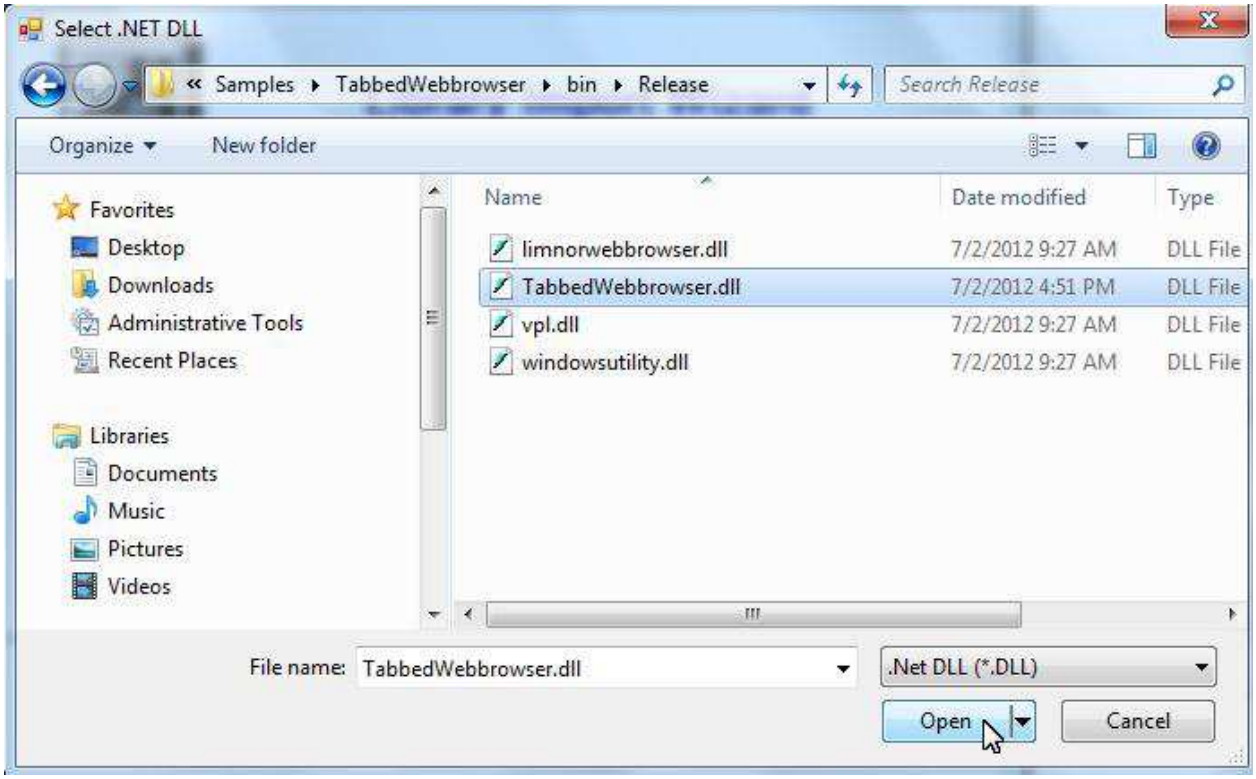
To use TabControlWebBrowser in the DLL, add it to the toolbox:

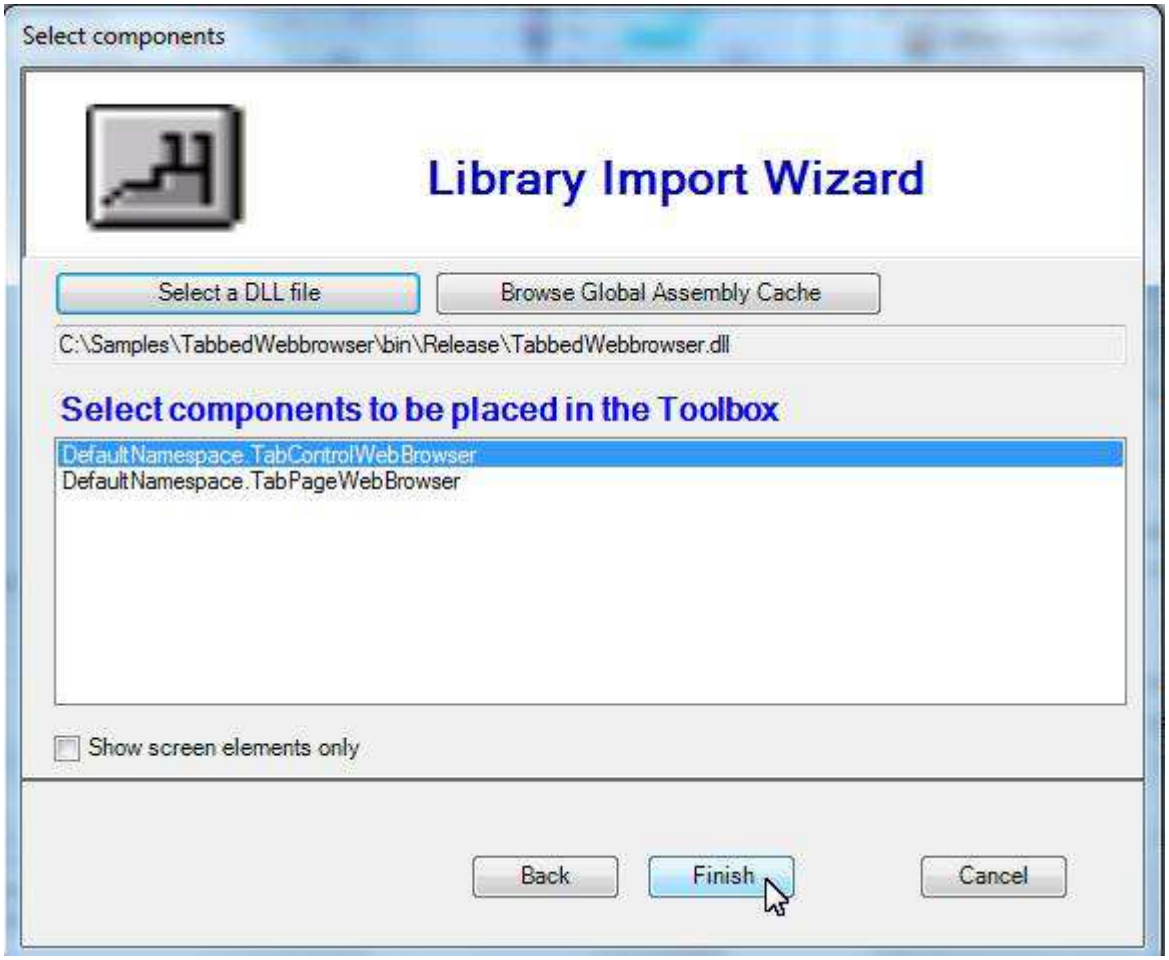




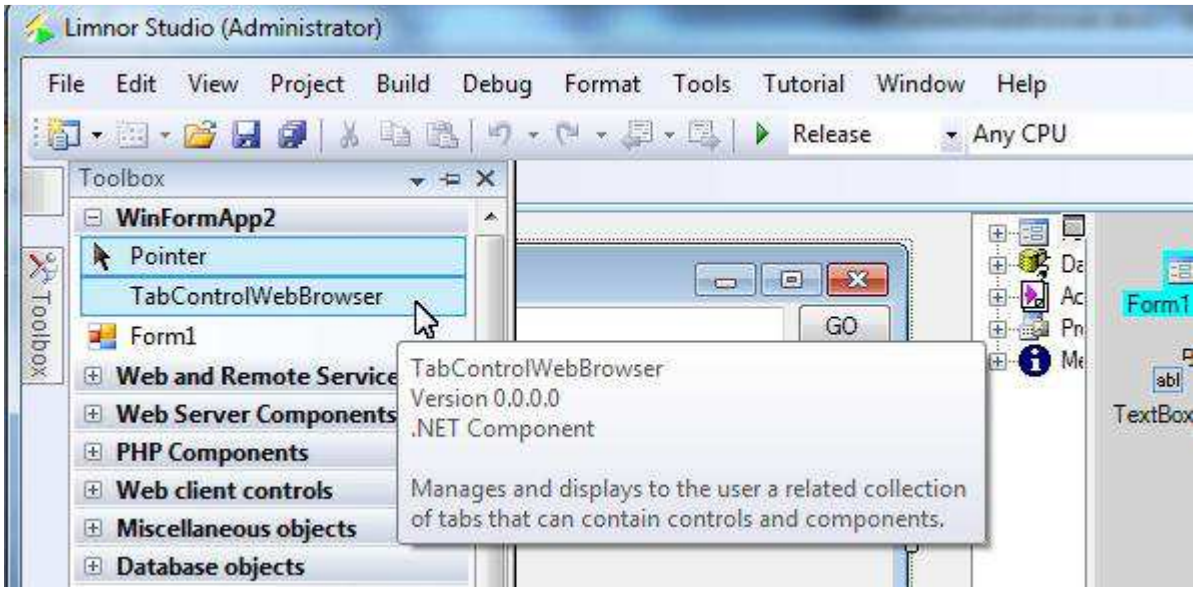

Select library type
Library Import Wizard
Import .NET classes
Import ActiveX components
Back
Next
Cancel


Select components
Library Import Wizard
Select a DLL file
Browse Global Assembly Cache

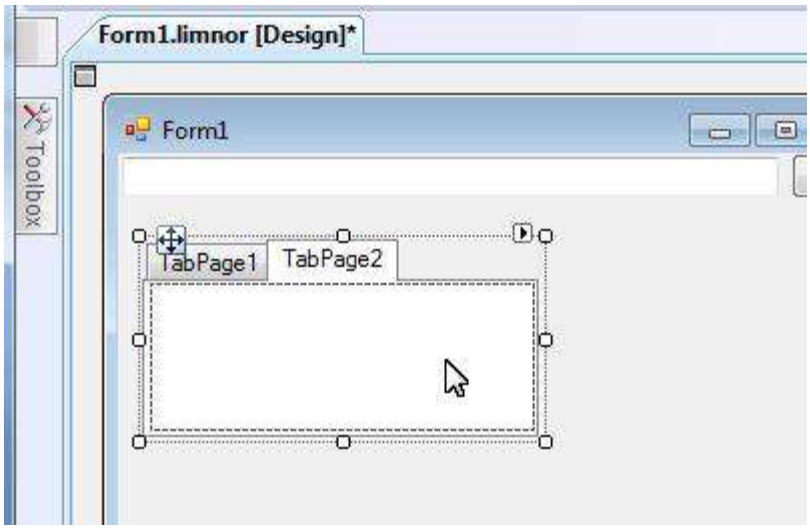

Select .NET DLL
« Samples ▸ TabbedWebbrowser ▸ bin ▸ Release
Search Release
Organize
New folder
Favorites
Desktop
Downloads
Administrative Tools
Recent Places
Libraries
Documents
Music
Pictures
Videos
Name
Date modified
Type
limnorwebbrowser.dll 7/2/2012 9:27 AM DLL File
TabbedWebbrowser.dll 7/2/2012 4:51 PM DLL File
vpl.dll 7/2/2012 9:27 AM DLL File
windowsutility.dll 7/2/2012 9:27 AM DLL File
File name: TabbedWebbrowser.dll
.Net DLL (*.DLL)
Open
Cancel



The class appears in the toolbox. Drop it to the form:

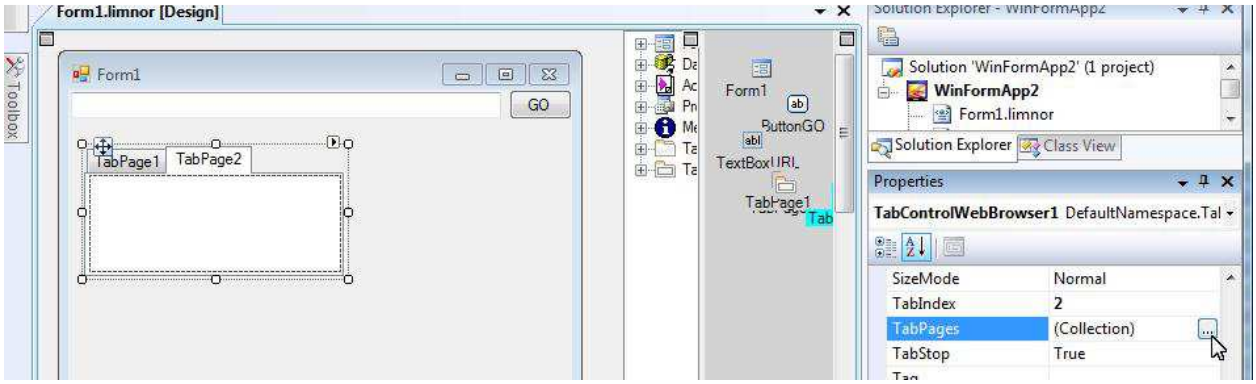


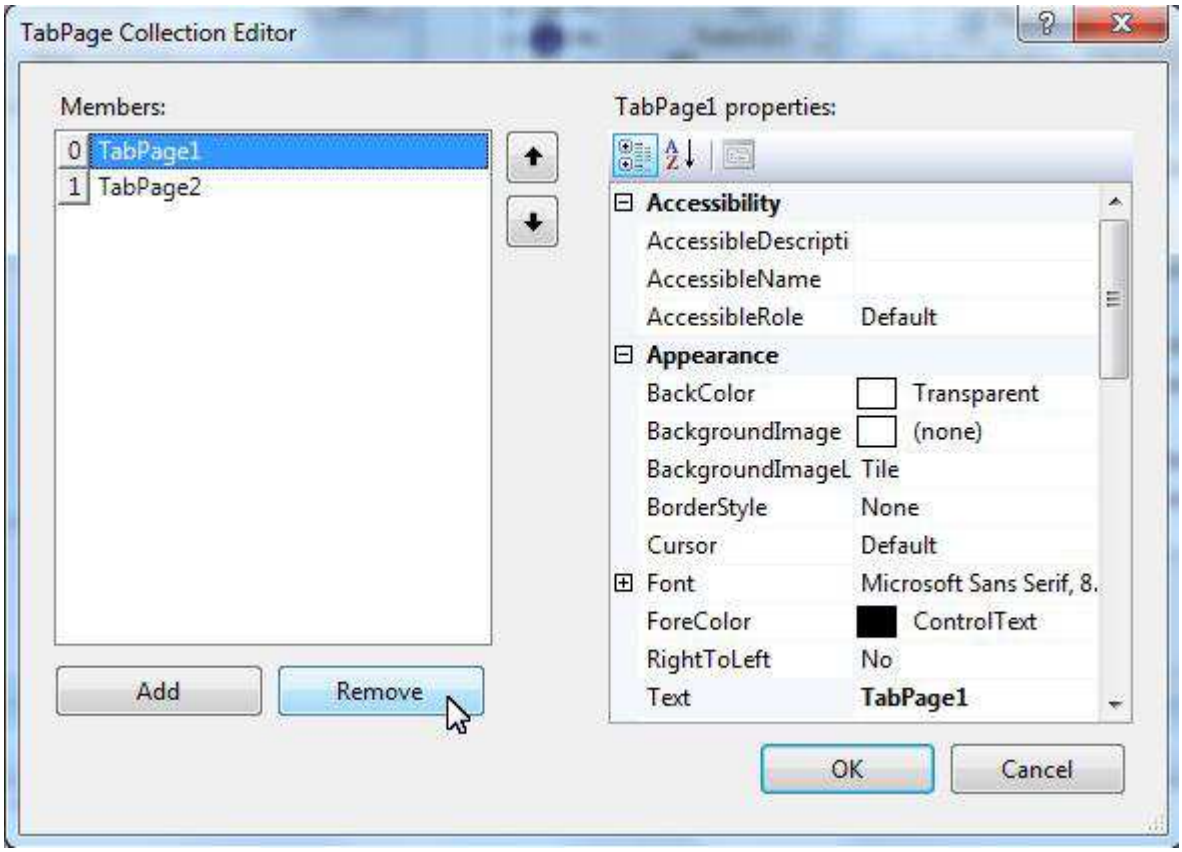
Note that Limnor Studio will automatically add two tab pages to the new tab control:

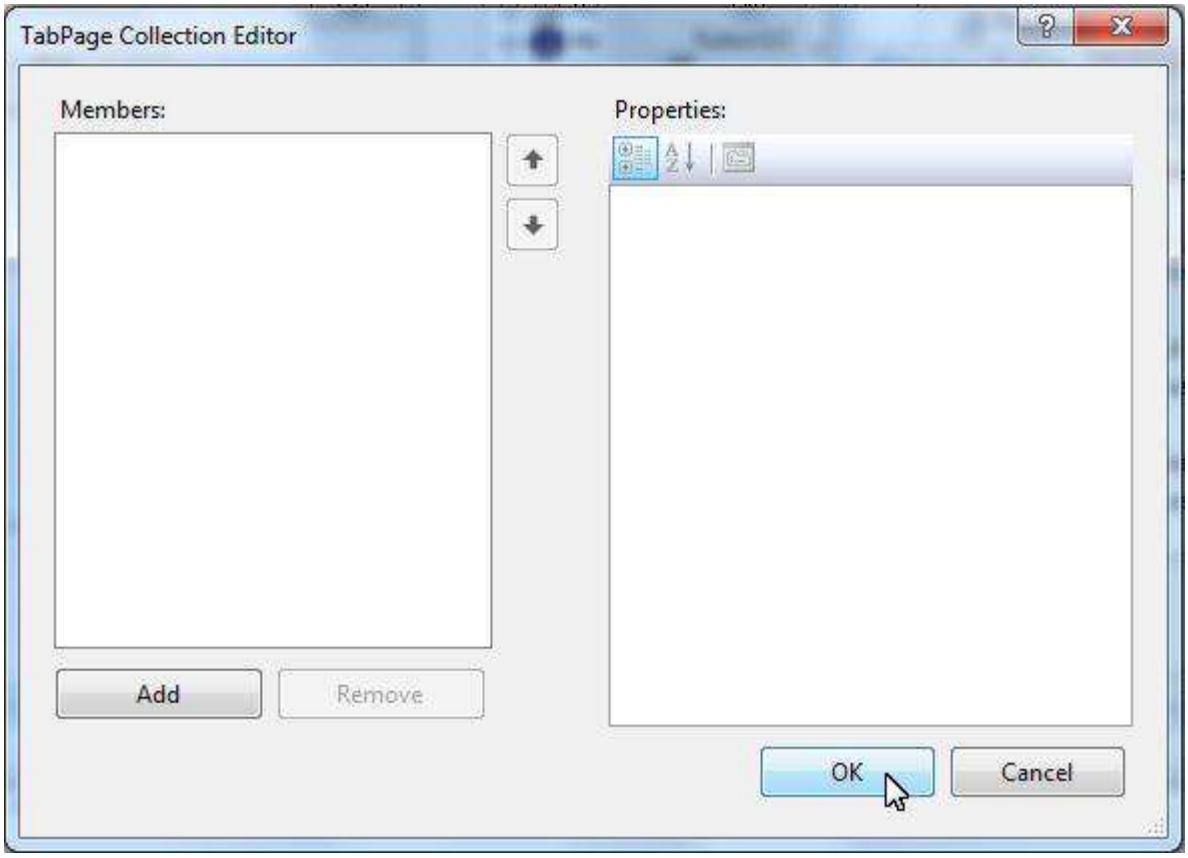


This is bad in our case because we only want TabPageWebBrowser in the tab control. We do not want normal tab pages in the tab control.

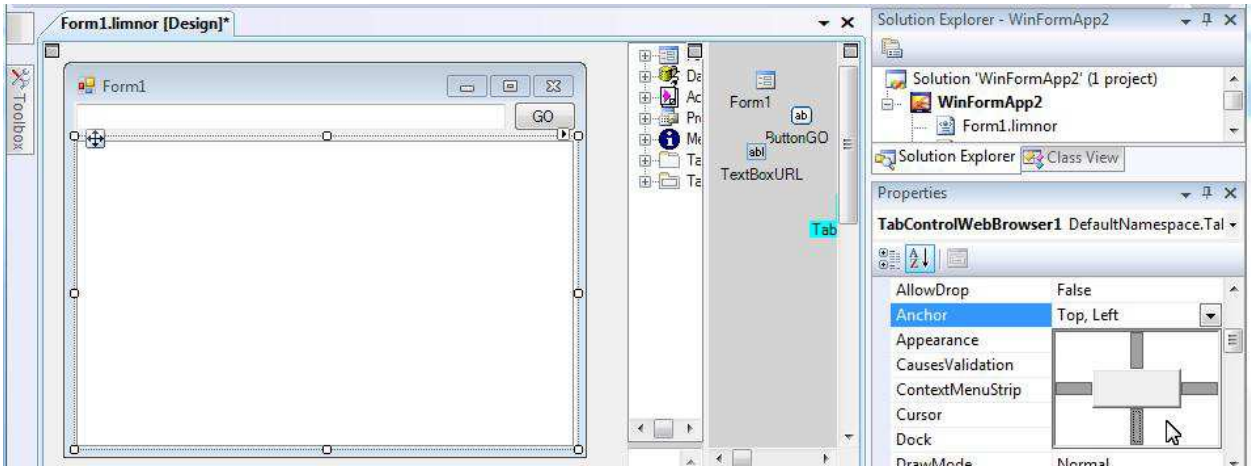
Remove these tab pages:




TabPage Collection Editor
Members:
0 TabPage1
1 TabPage2
Add
Remove
TabPage1 properties:
Accessibility
AccessibleDescripti
AccessibleName
AccessibleRole Default
Appearance
BackColor Transparent
BackgroundImage (none)
BackgroundImageL Tile
BorderStyle None
Cursor Default
Font Microsoft Sans Serif, 8.
ForeColor ControlText
RightToLeft No
Text TabPage1
OK
Cancel


TabPage Collection Editor
Members:
Add
Remove
Properties:
OK
Cancel

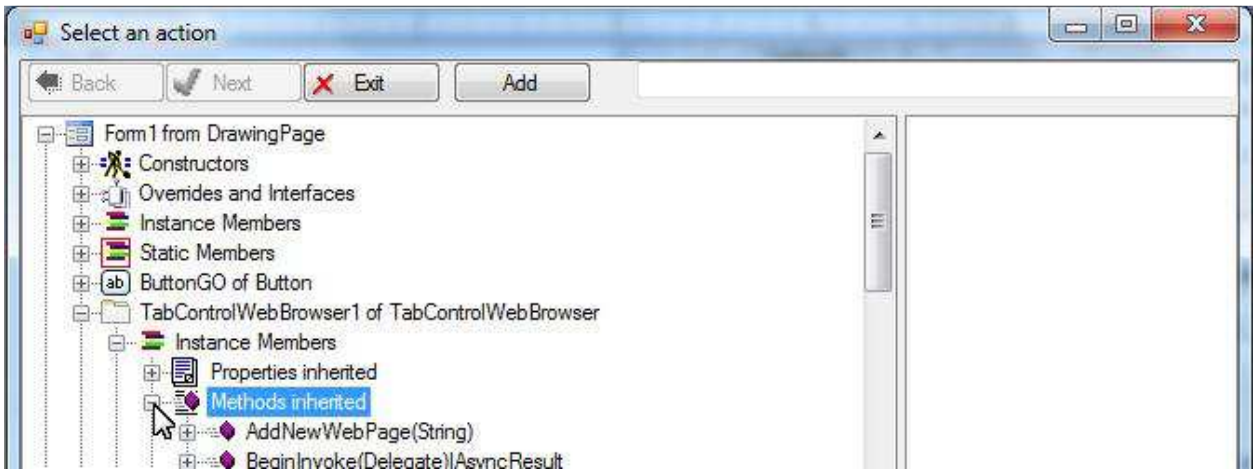
Set its Anchor property:

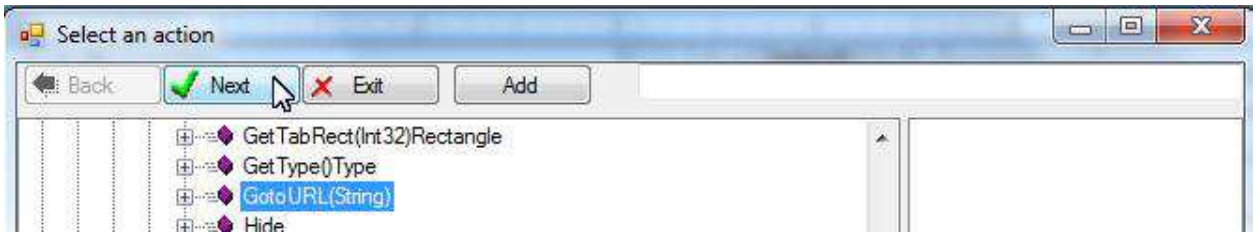


Show Web Page by GO button

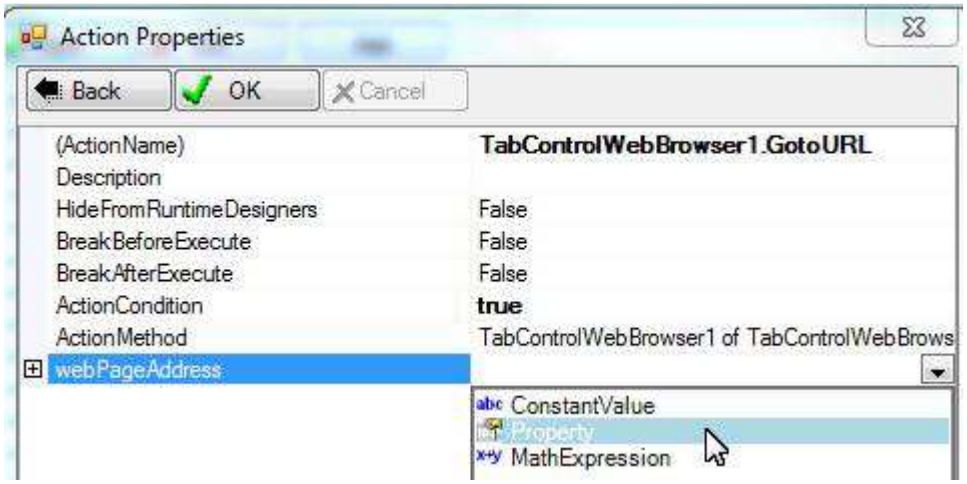
Execute GotoURL when clicking “GO” button:

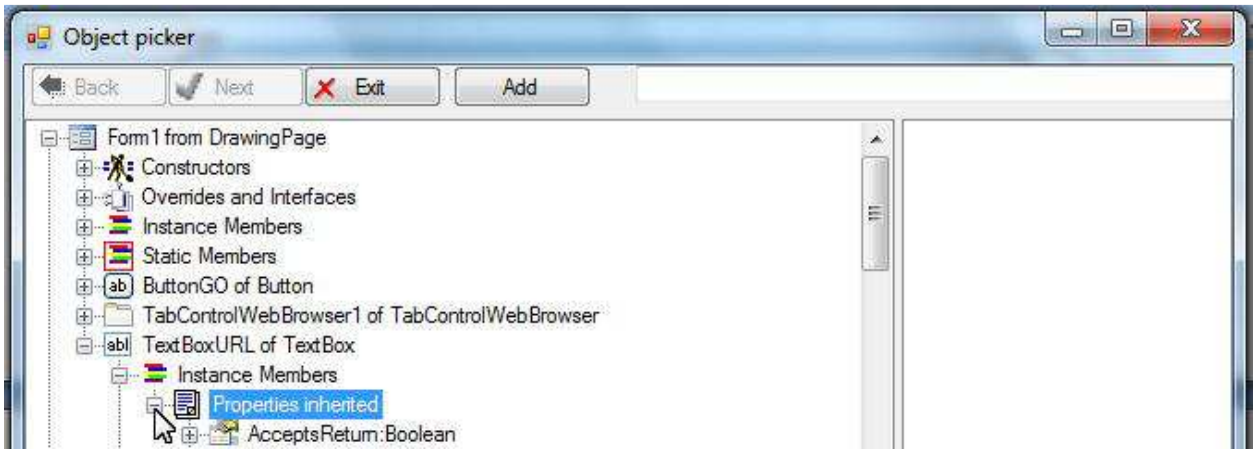
Select GotoURL:

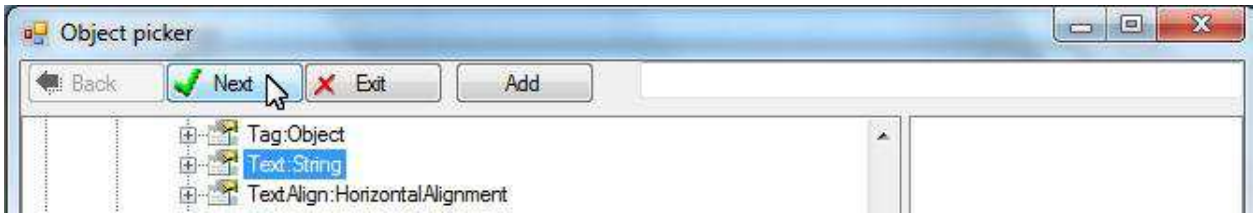




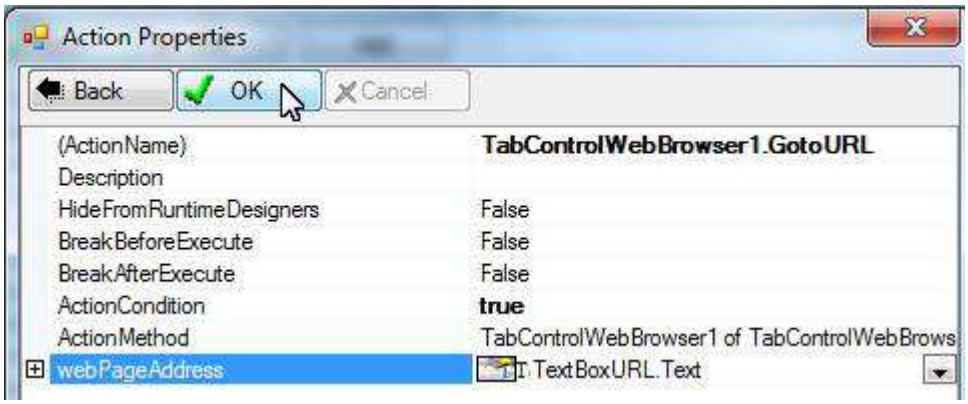
Pass Text property of the text box to the action:



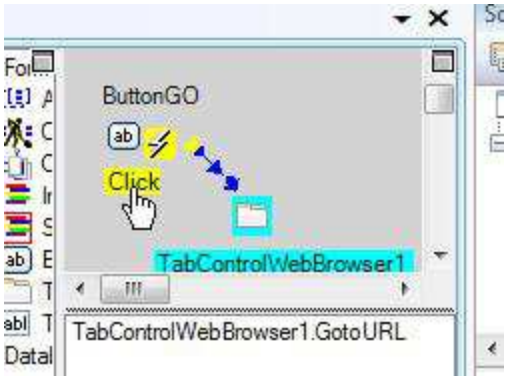




Link OK:

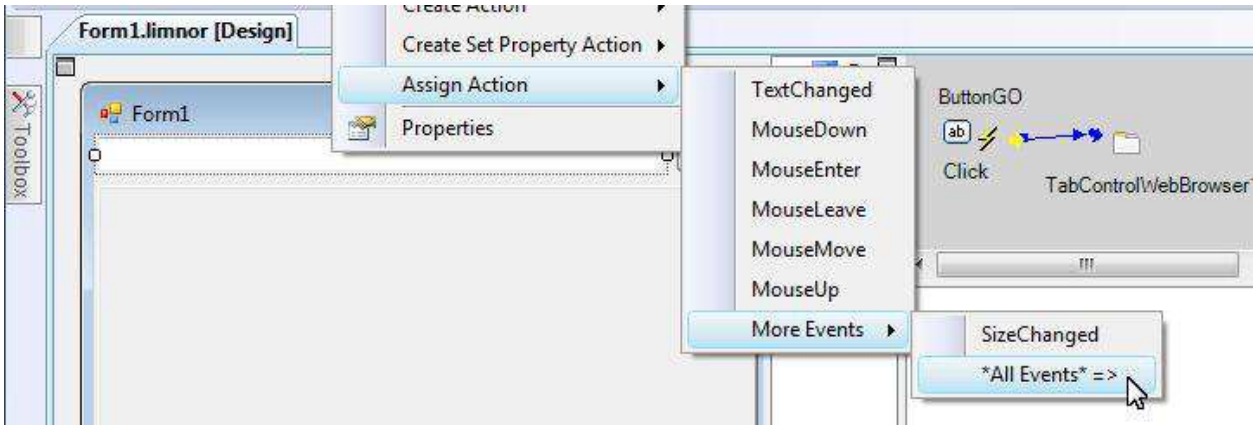


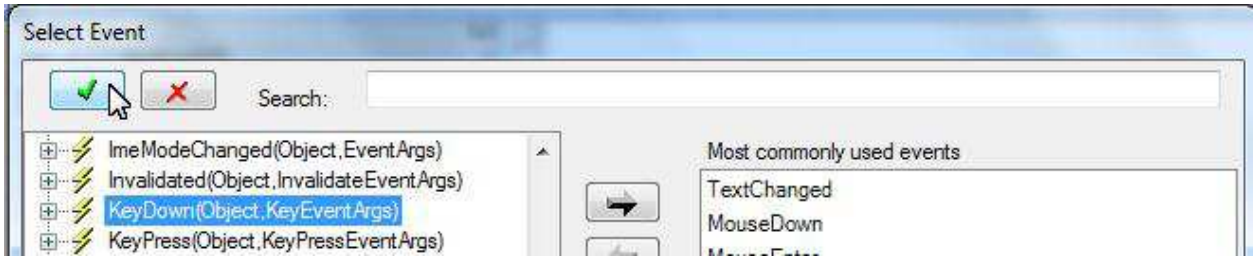
The action is created and linked to the event:

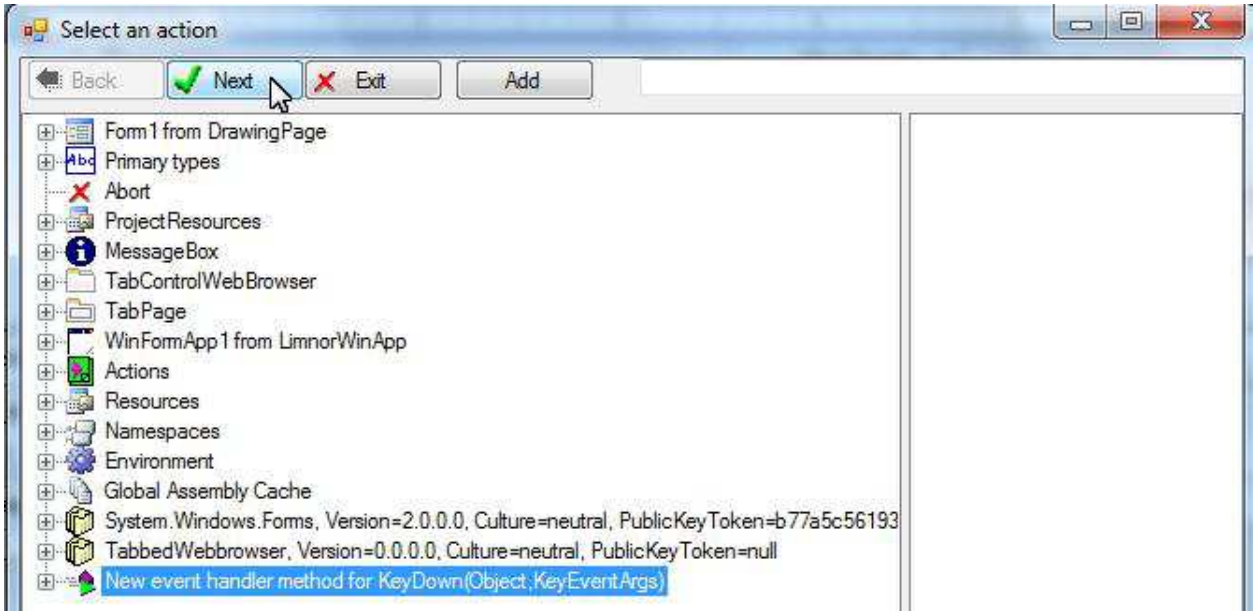


Show Web Page By Enter Key

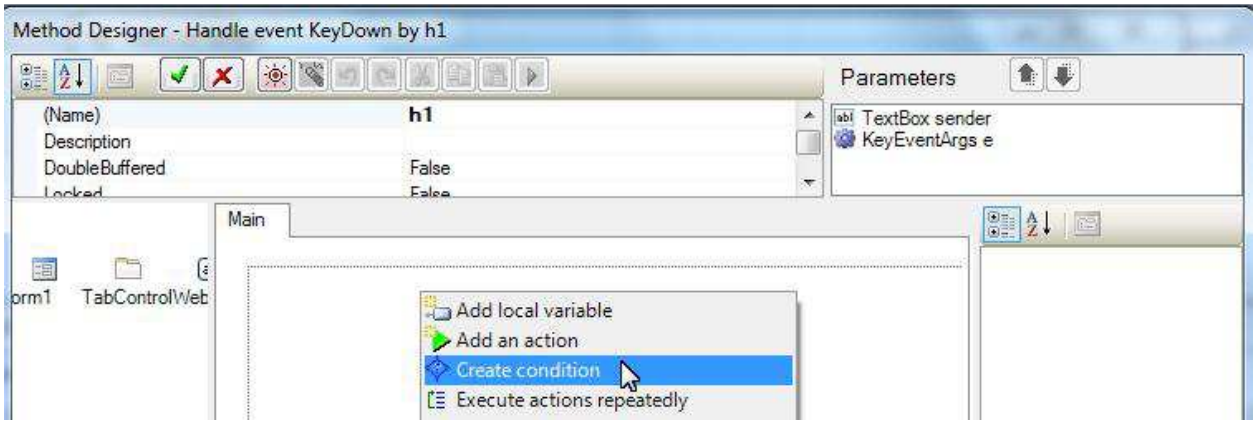
We may also assign this action to the KeyDown event when Enter key is pressed:

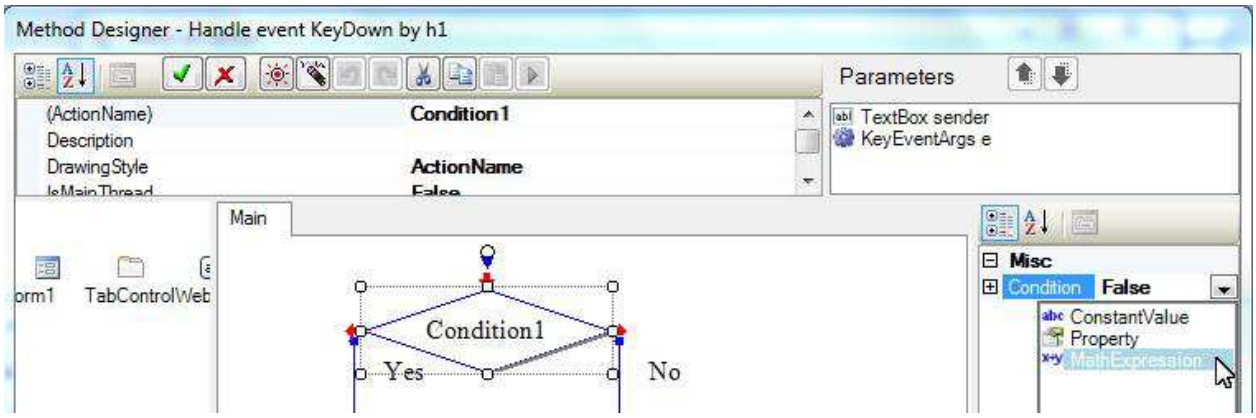


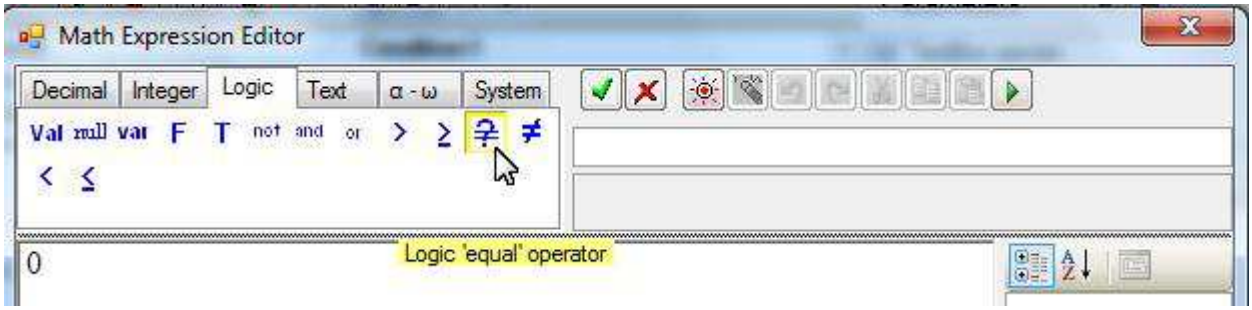


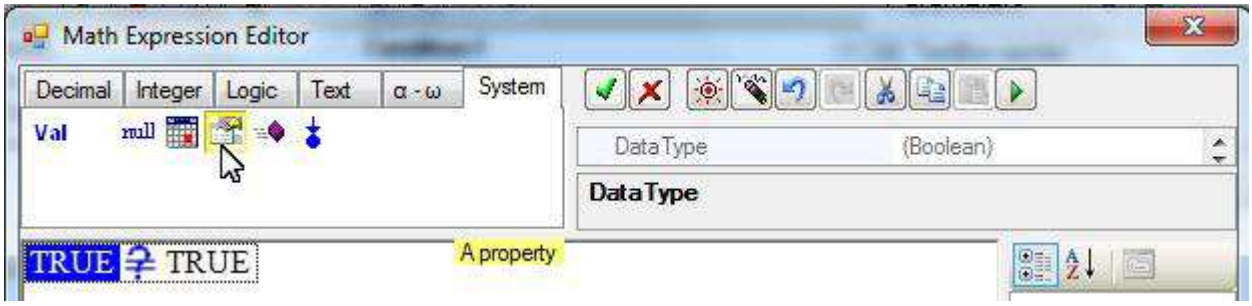


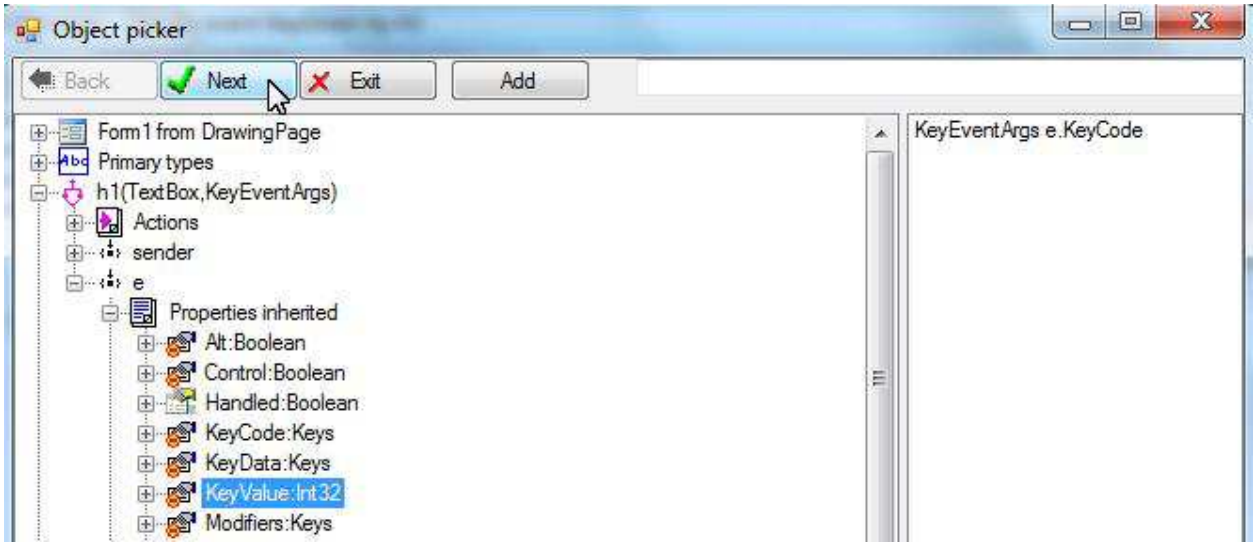
Add a Condition to check for Enter key:

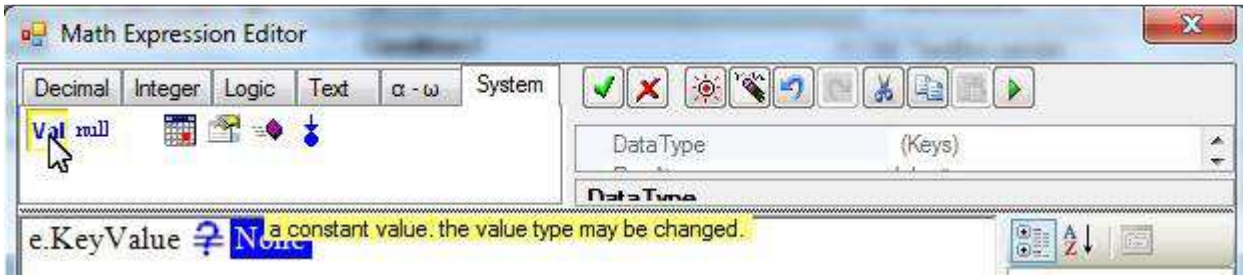




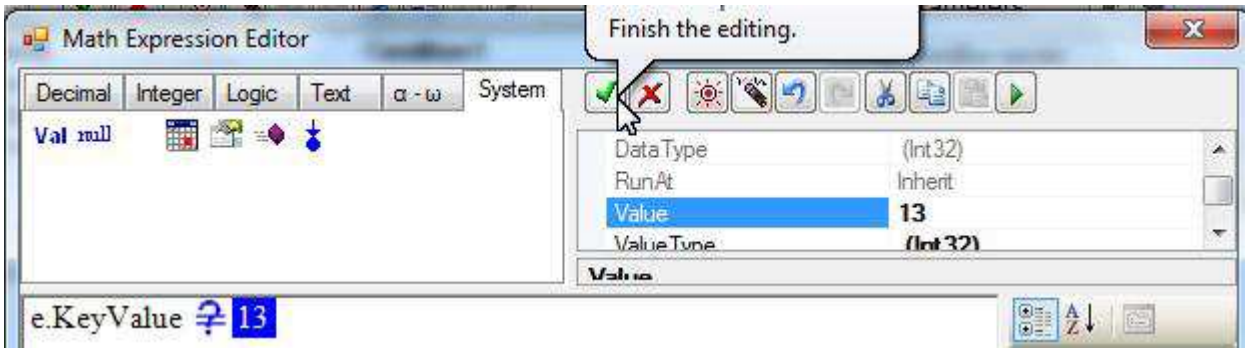




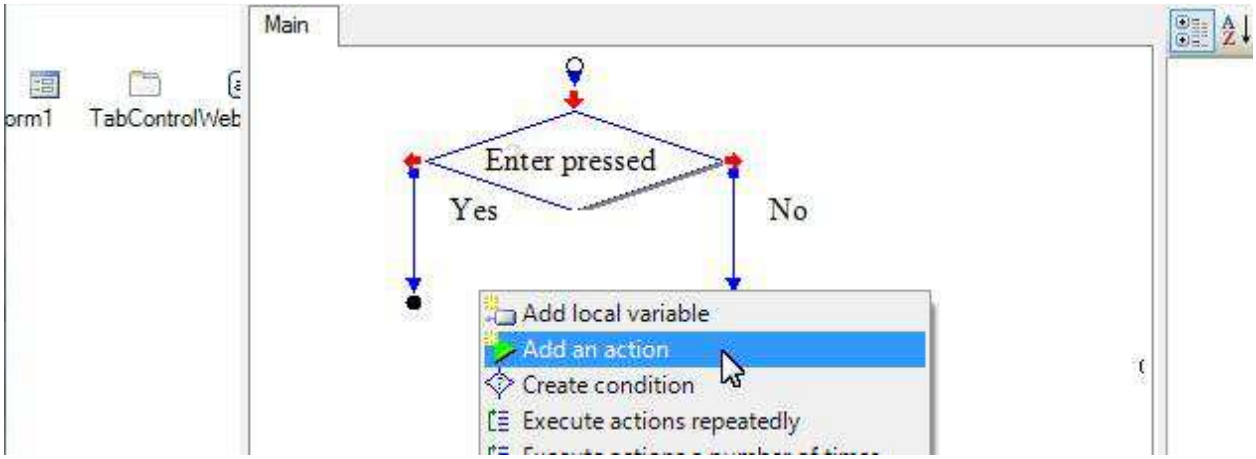


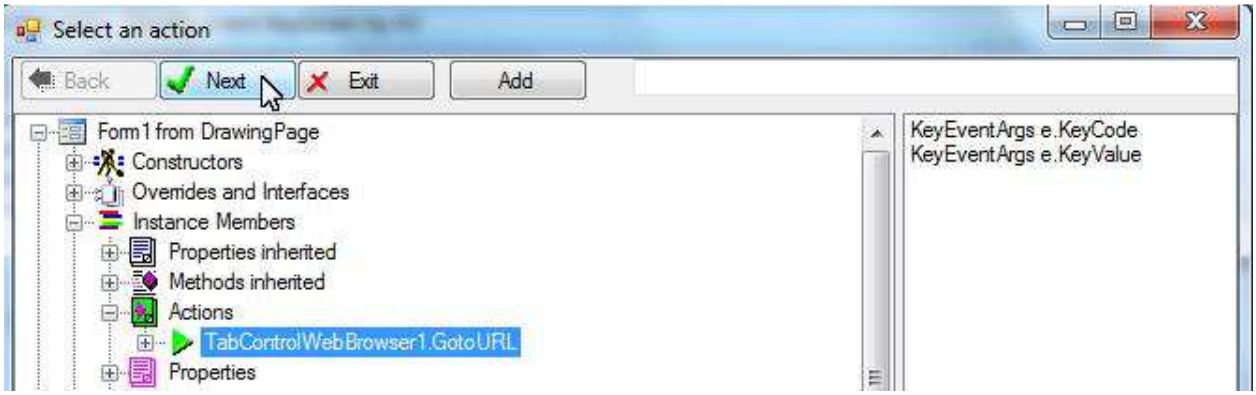


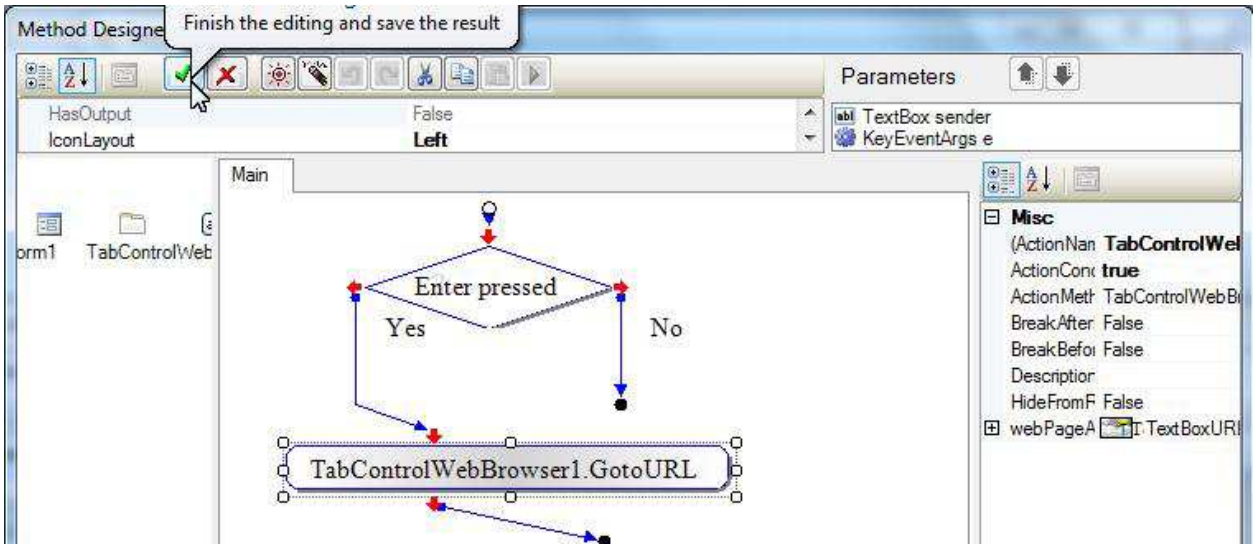
Number 13 represents the Enter key:



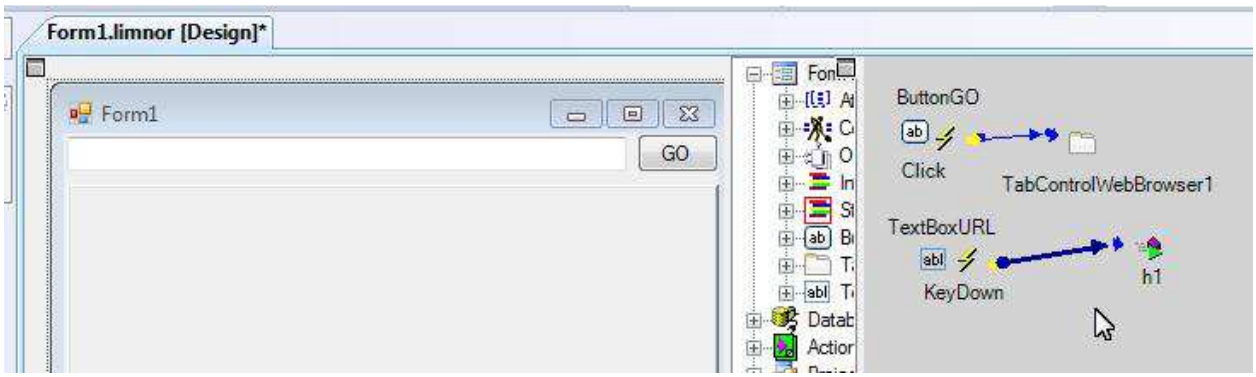
Link GotoURL action to “Yes”:





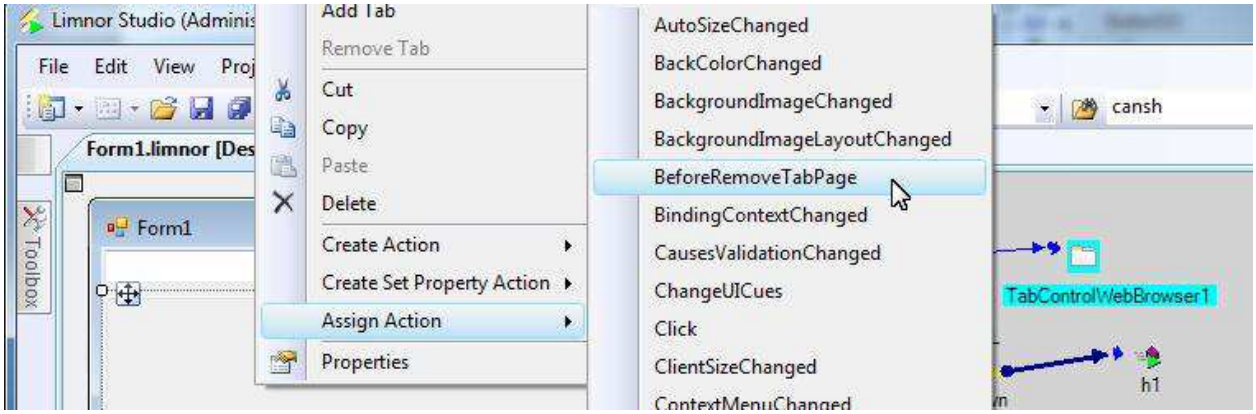


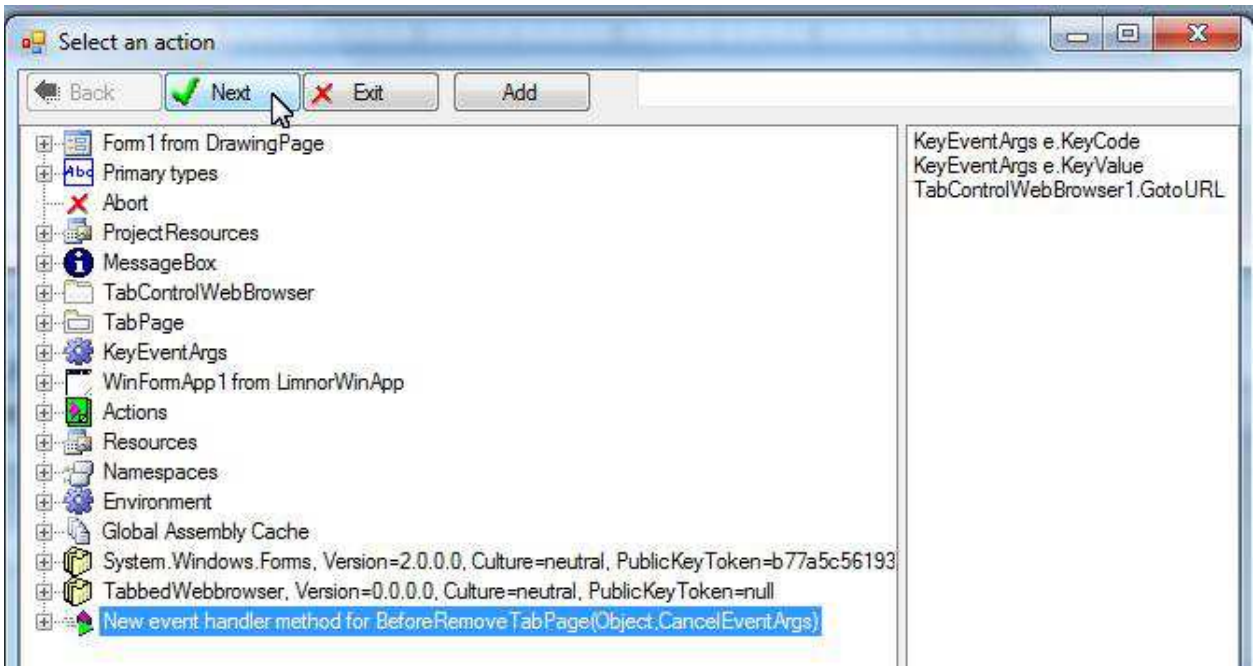
The action is linked to the text box:



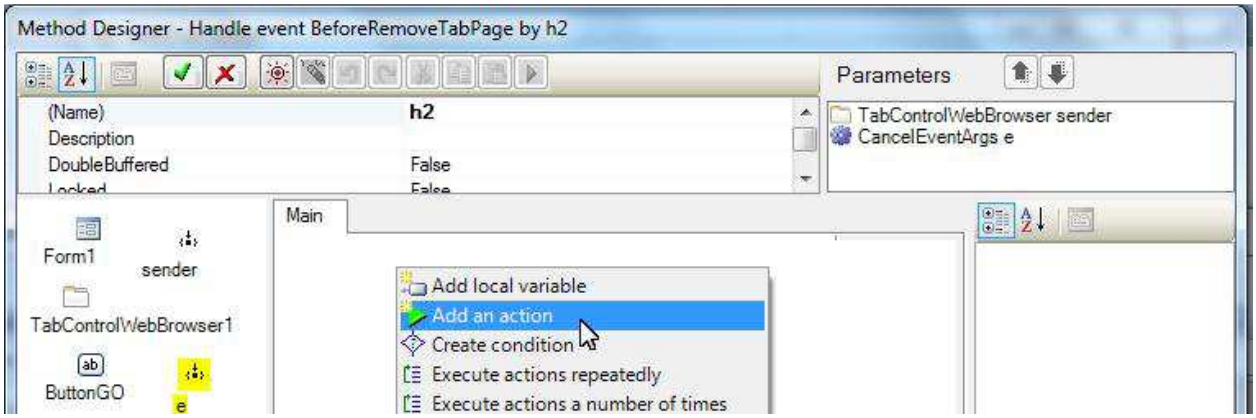
Confirm/Cancel Tab Remove

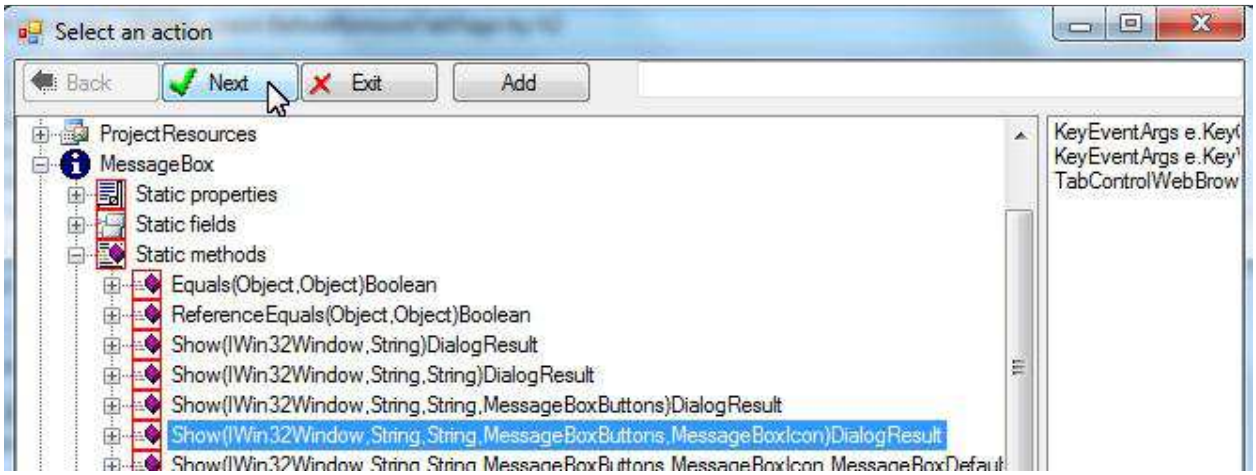
We may handler event BeforeRemoveTabPage to provide a cancelation prompt:



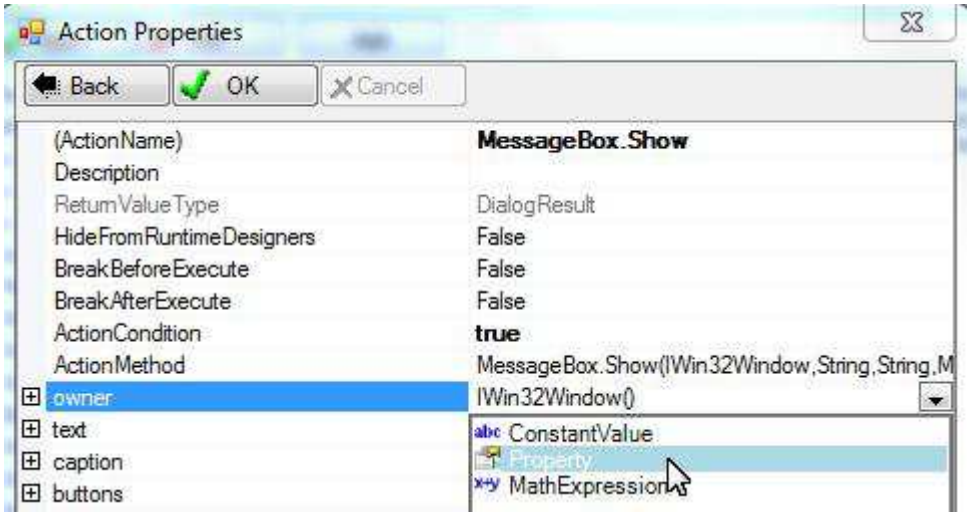


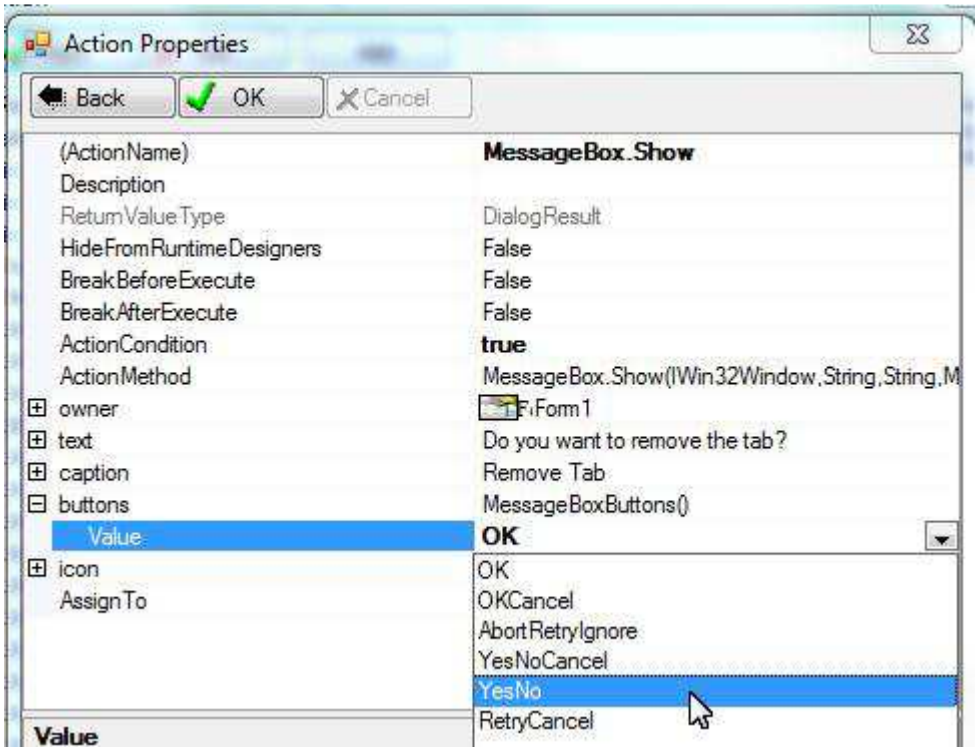
Add message box to ask for confirmation:

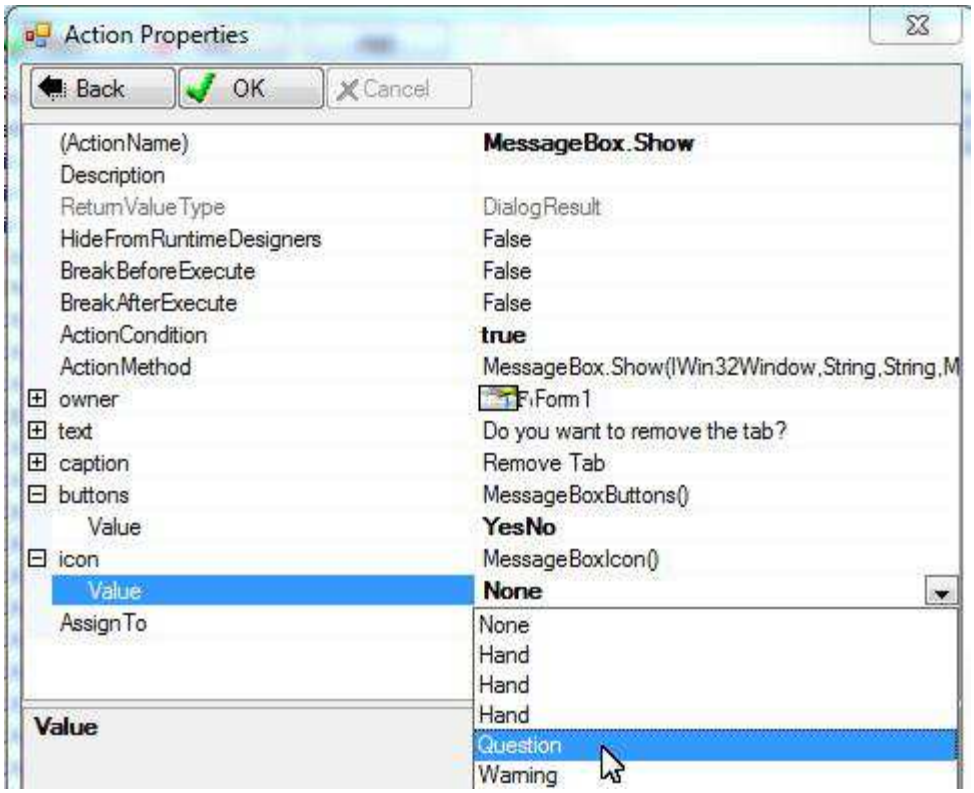


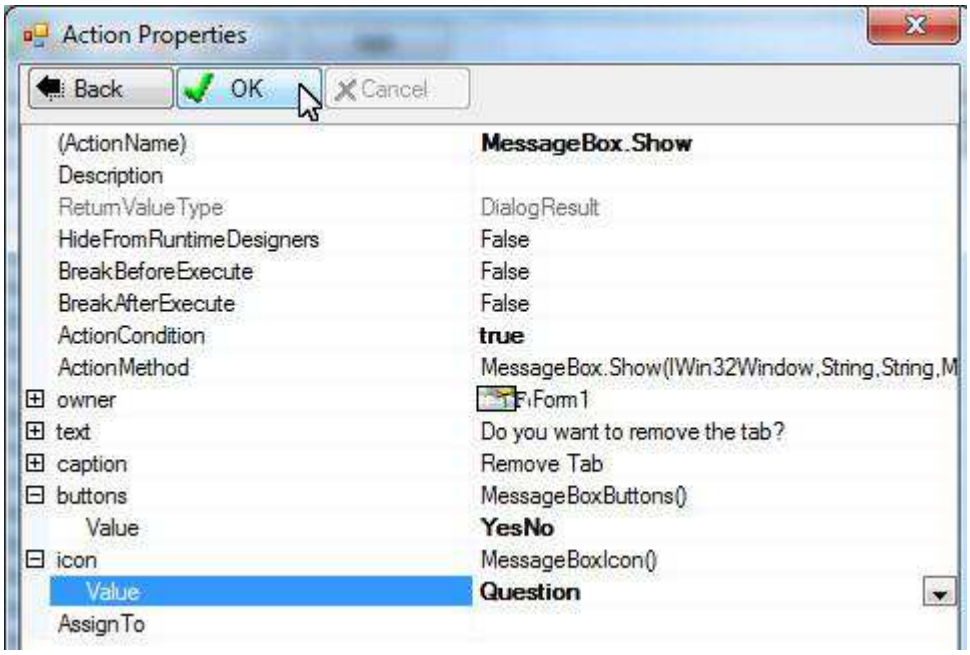


Set the parameters for the action:

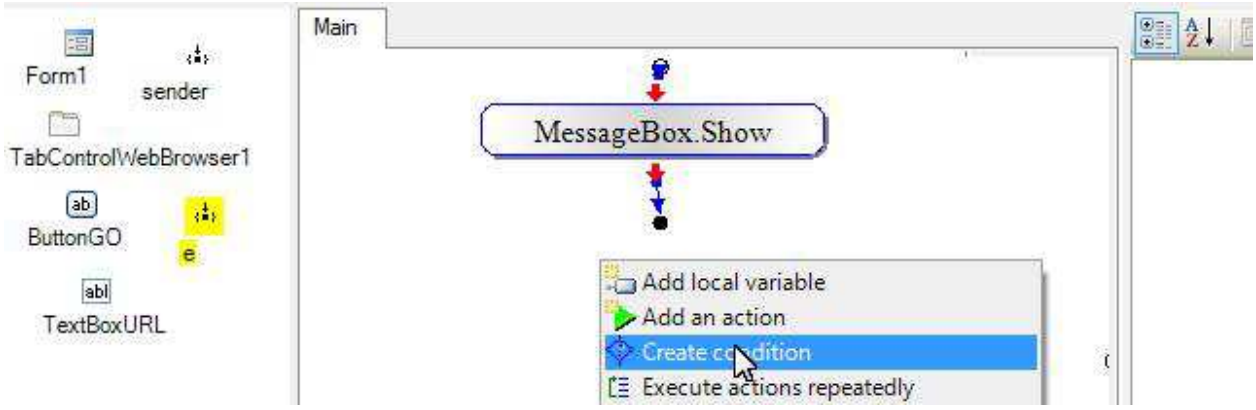




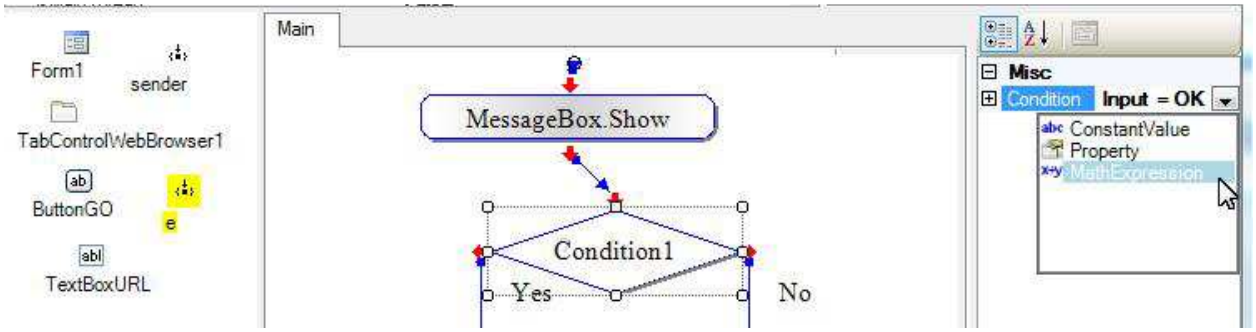


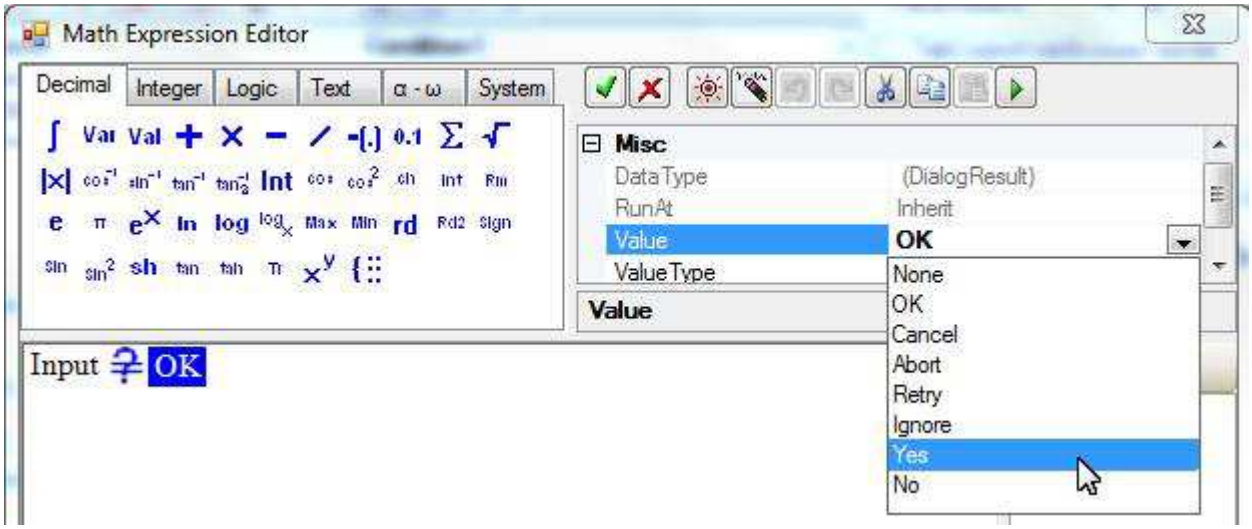


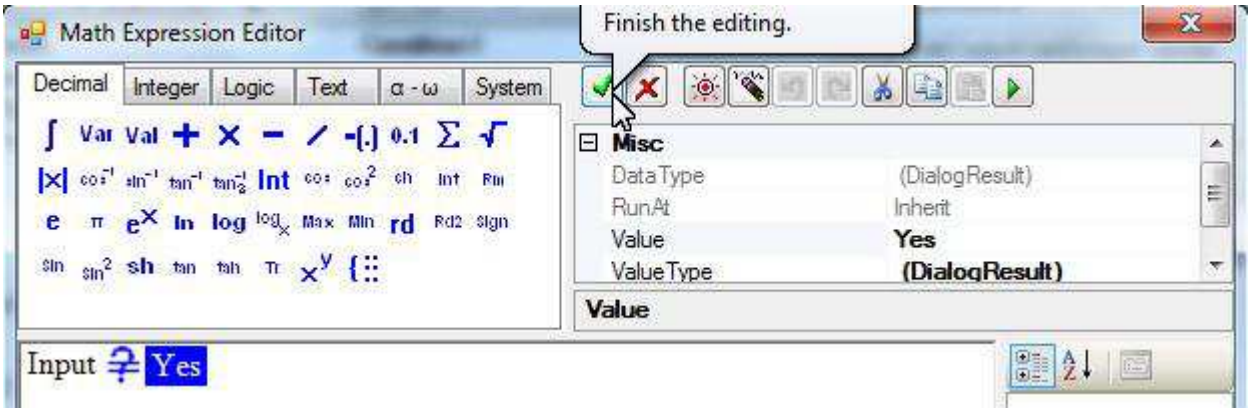
Add a condition to check the message box result:



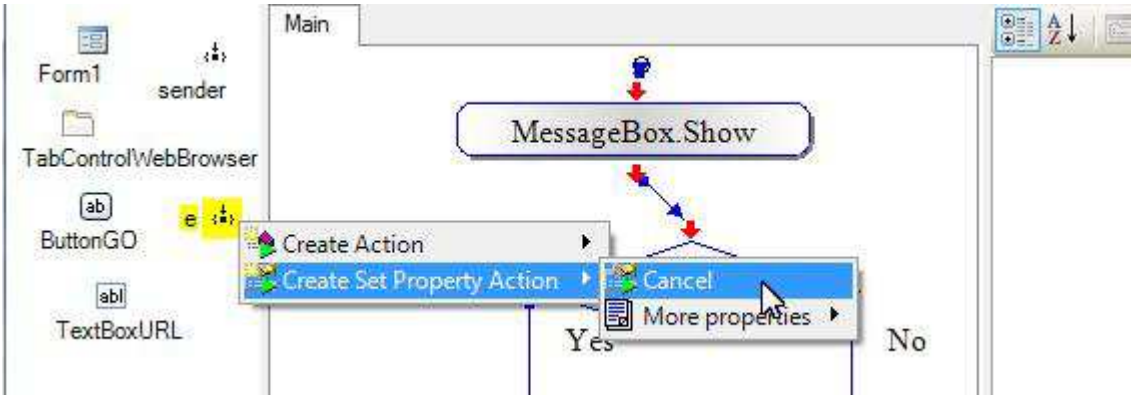
Adjust the Condition:

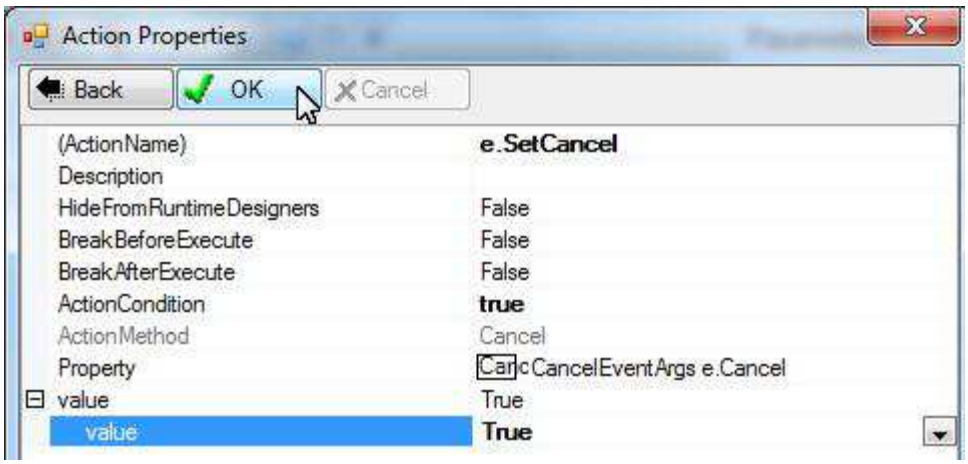




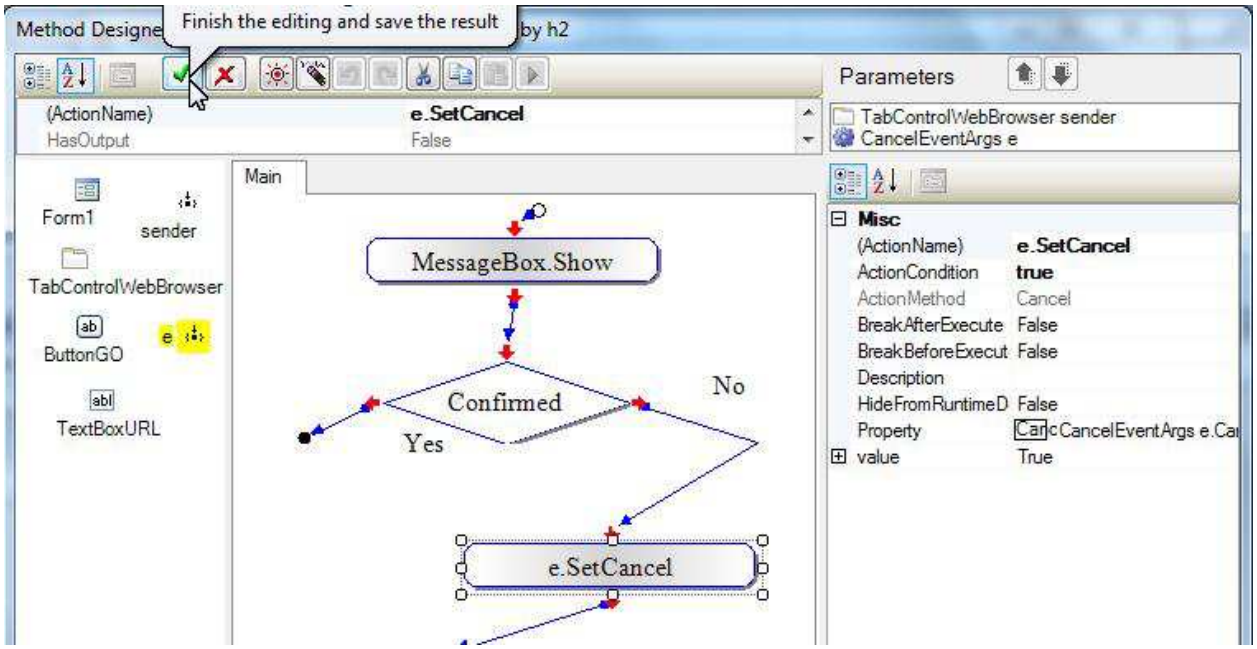


Create a Cancel action:

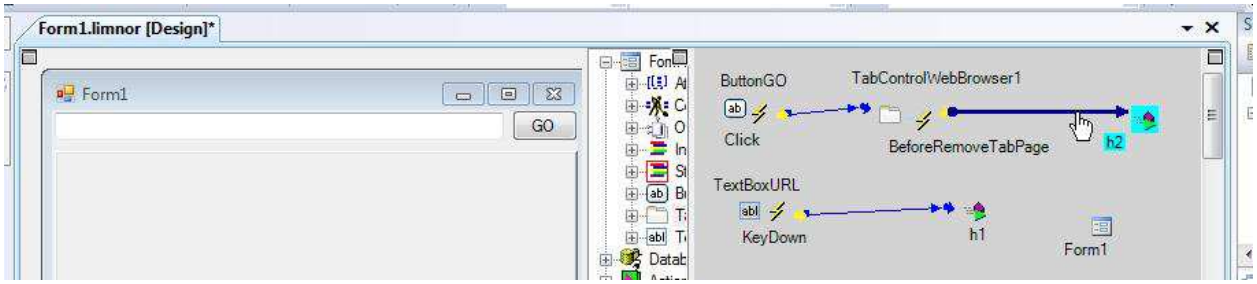




Link the action to “No” port:

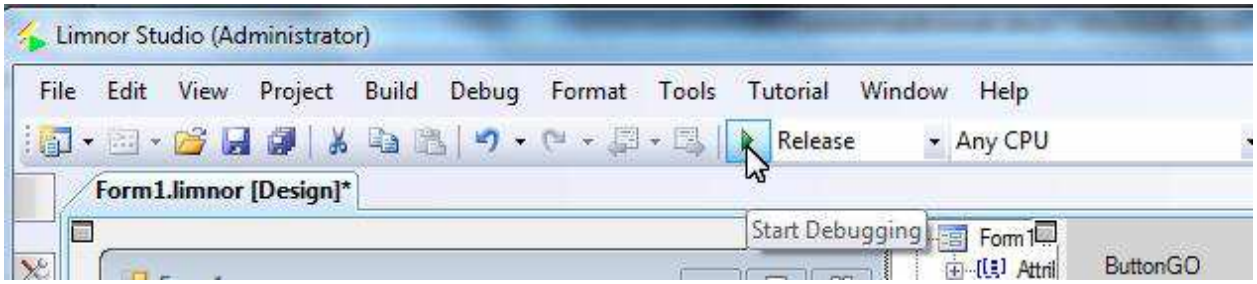


The handler method appears in the Event Map:

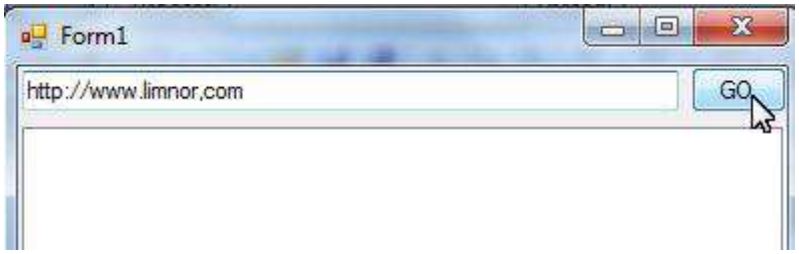


Test

We may compile and test the program:



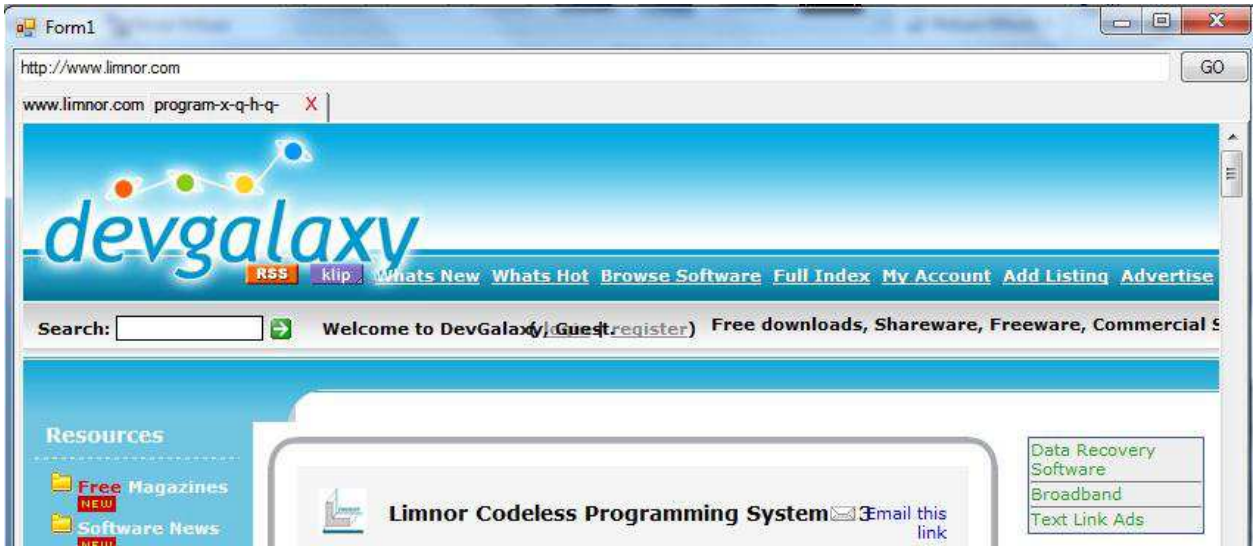
The form appears. Enter an URL in the text box. Click “GO”:



The web page is displayed in the tab page:

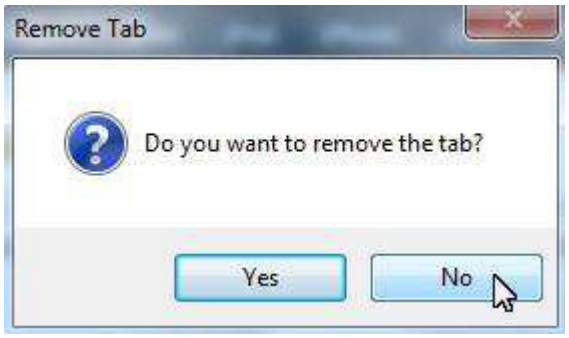
Go to a web page containing popup links. Click a popup link:

A new tab page appears displaying the popup page:



We may keep navigating to popup links and get more tabs:

Click the red X, a message box appears:

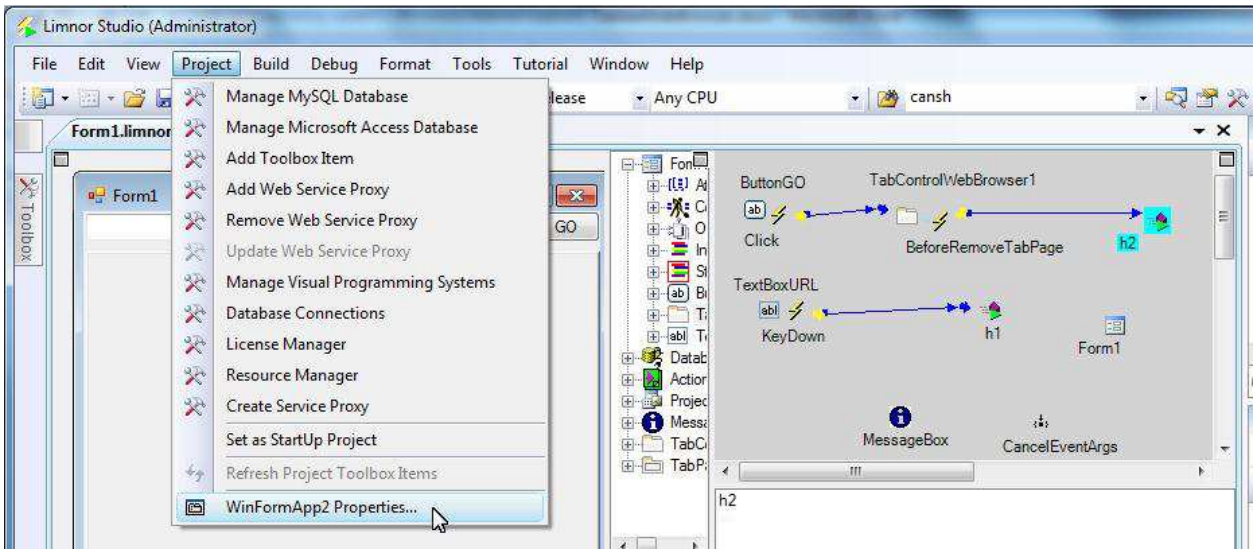


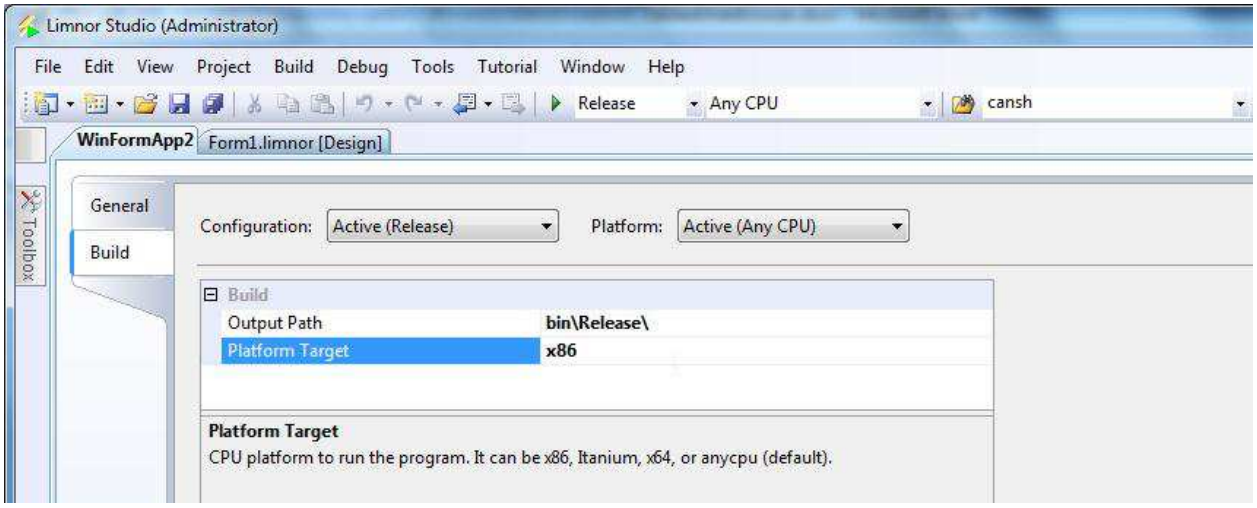
Click No. The tab is not removed.

Click the red X again. Click “Yes”. The tab is removed.

Adobe problem for reading PDF file

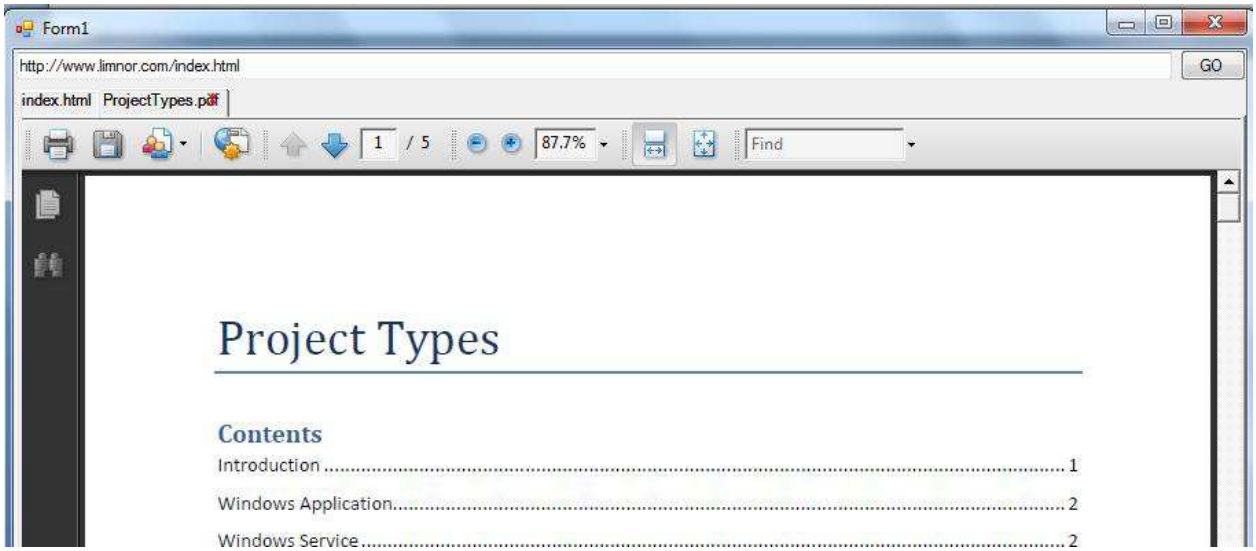
Adobe does not provide a 64-bit PDF reader at this time. You have to force your program to be 32-bit if you want to show PDF files inside tab pages. You can do it by setting the target platform to “x86”:





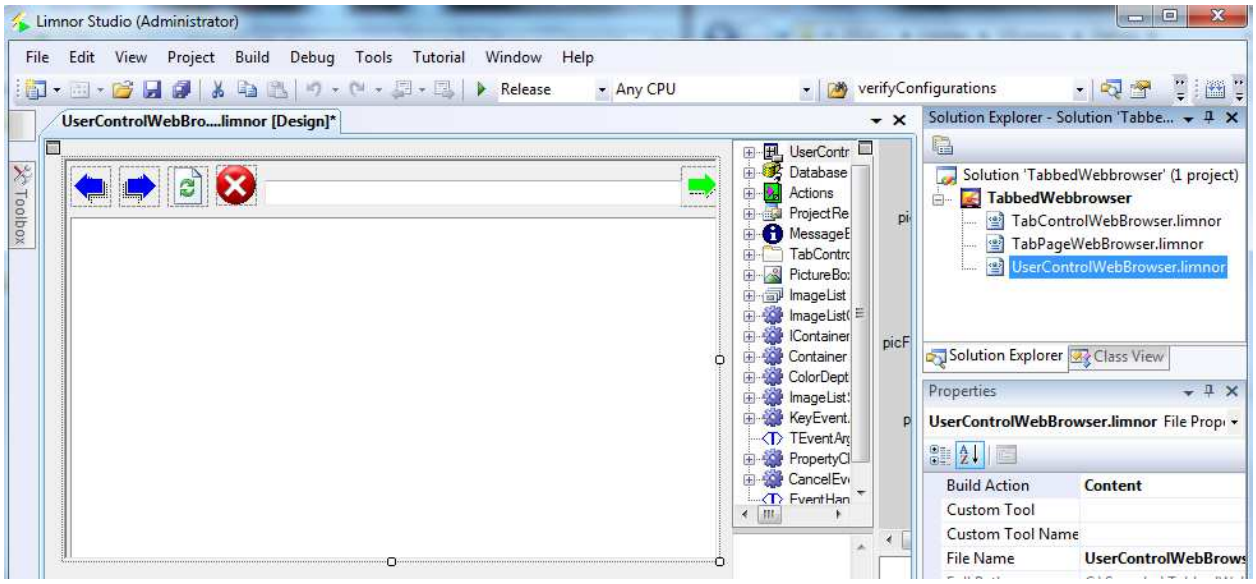
For more information on setting target platform, see http://www.limnor.com/studio_x86.html

Now the PDF file may be displayed in tabs:

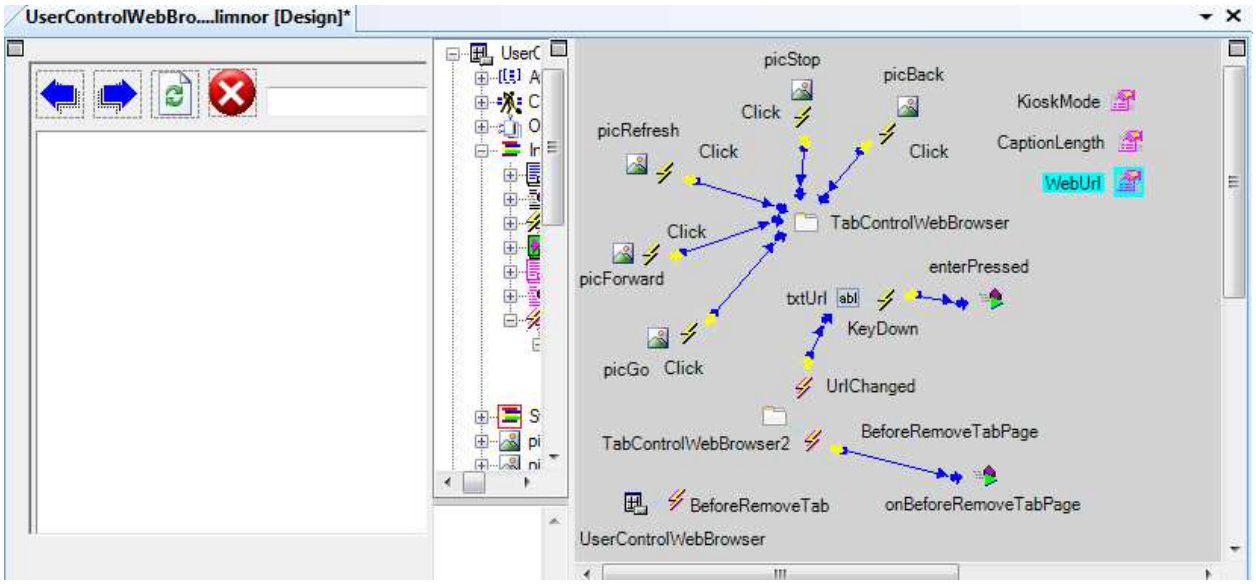


Put together a web browser

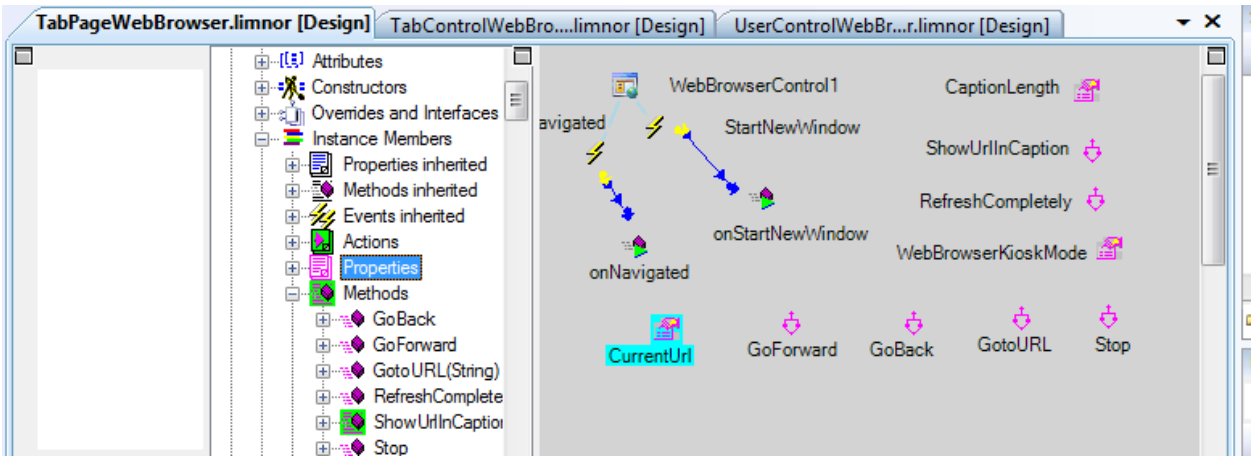
We may create a user control to put together a web browser. It uses a Tabbed Web Browser control to display web pages. It uses picture boxes for executing web browser commands: go forward, go backward, refresh, stop, and go to web address. It uses a text box for web address.



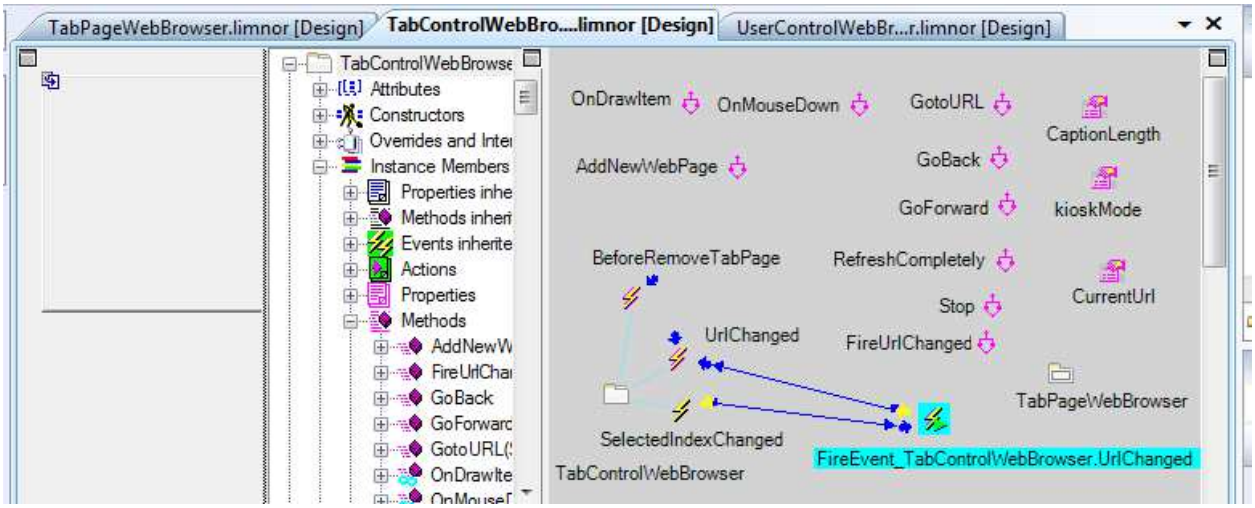
It creates a KioskMode property, a CaptionLength property and a WebUrl property. It creates a BeforeRemoveTab event and a UrlChanged event.



The functionality of all the commands, properties, and events are published from the web browser control in the TabPageWebBroser:



The methods, properties and events are further published by TabControlWebBrowser so that the user control may use them:



Feedback

Please send your feedback to support@limnor.com