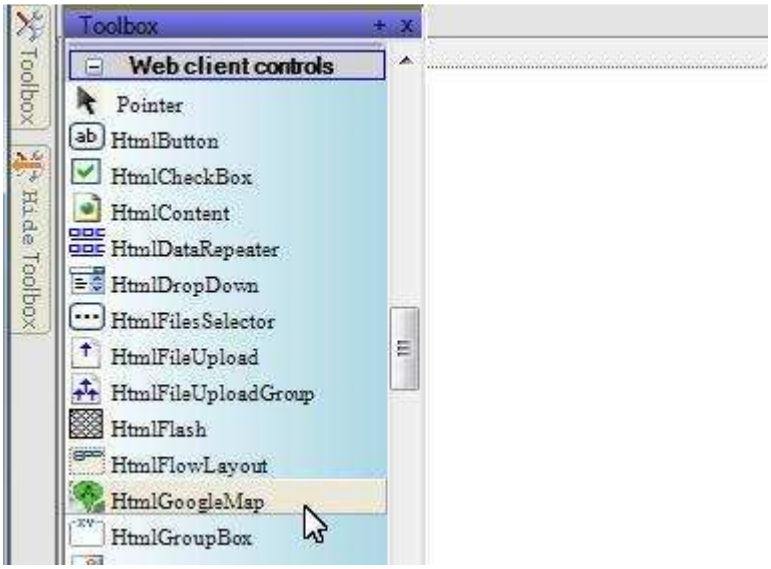
Use Google Maps Control

Contents

Introduction

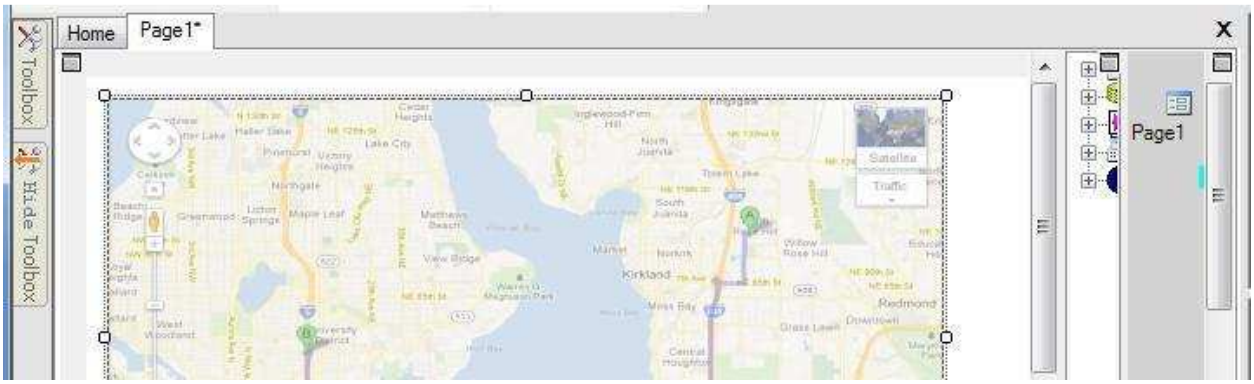
HtmlGoogleMap is a web control for using Google Maps services on web pages.



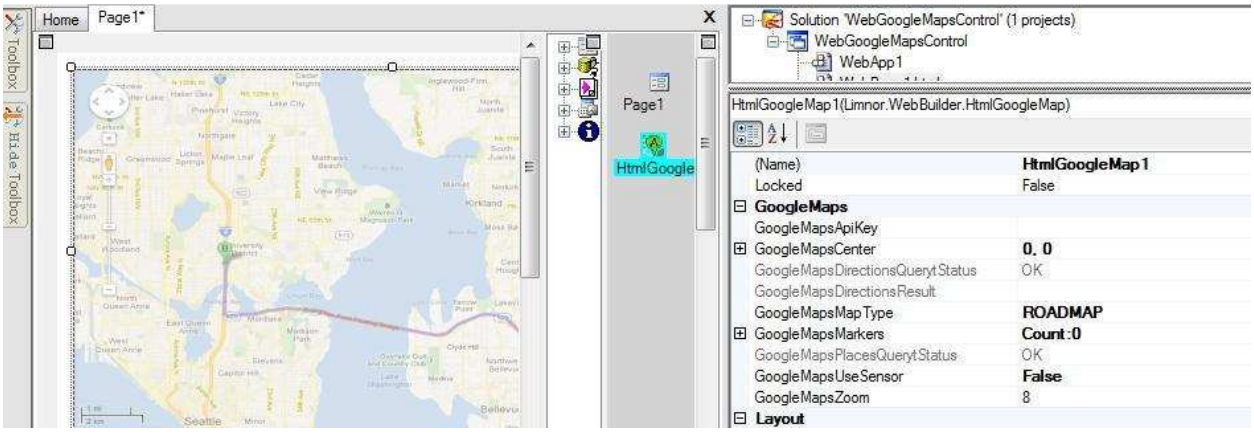
This document shows examples of using HtmlGoogleMap.

Setup Google Maps

Drag and drop HtmlGoogleMap to a web page to use it:



You may set properties of HtmlGoogleMap:



- **GoogleMapsApiKey** – This is a key you get from Google. For more information, see https://developers.google.com/maps/documentation/javascript/tutorial#api_key
- **GoogleMapsCenter** – This is a location used as the center of the map when the web page is loaded. For example, use 47.61658, -122.1992 for a place near Bellevue, Washington.
- **GoogleMapsMapType** – It indicates type of map when the web page is loaded.
- **GoogleMapsUseSensor** – It indicates whether sensor is used.
- **GoogleMapsZoom** – It indicates initial zoom factor when the web page is loaded. Use a larger value for a more detailed display.

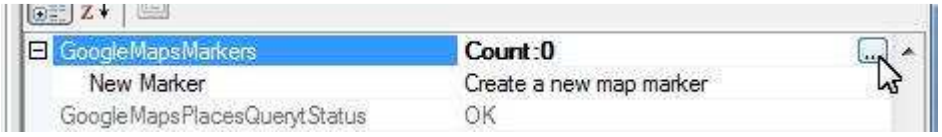
Use Markers

Add markers at design time

GoogleMapsMarkers property allows you to add markers at design time.

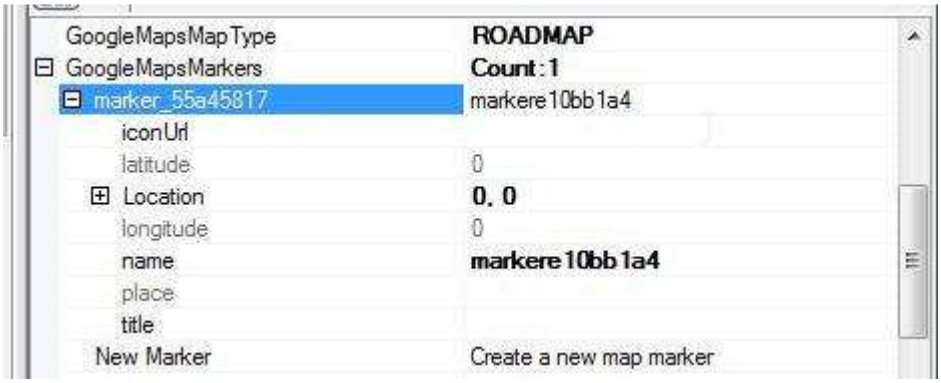
Click "…" on "New Marker" to add a new marker:

Click "…" on "GoogleMapsMarkers" to manage all added markers by deleting existing markers and adding new markers:

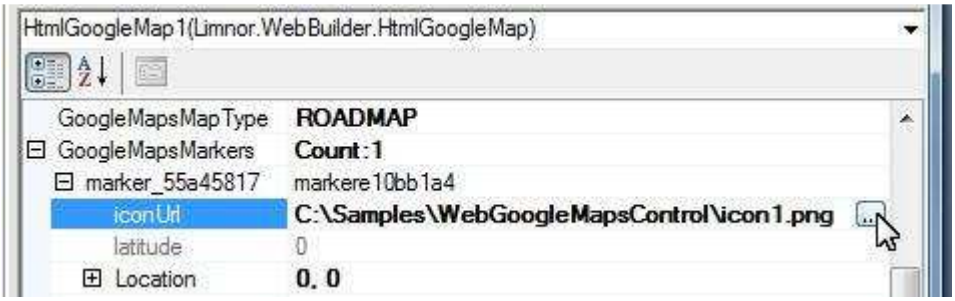


Marker Properties

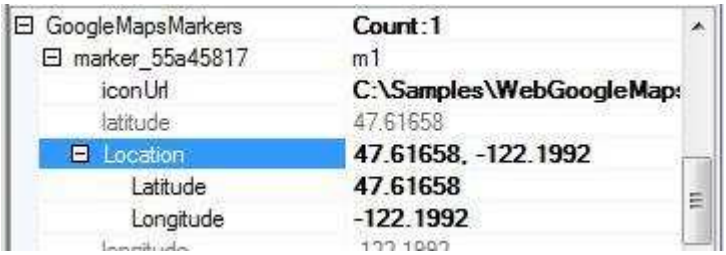
You may set properties for all markers:



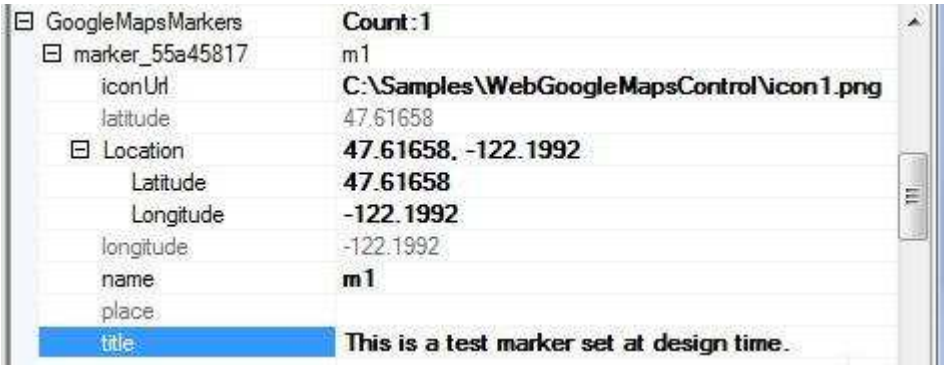
- **iconUrl** – It is an image used as marker icon. You may use a web URL or pick an image file from a local disk.



- **Location** – It is the location of the marker. X indicates latitude and Y indicates longitude:



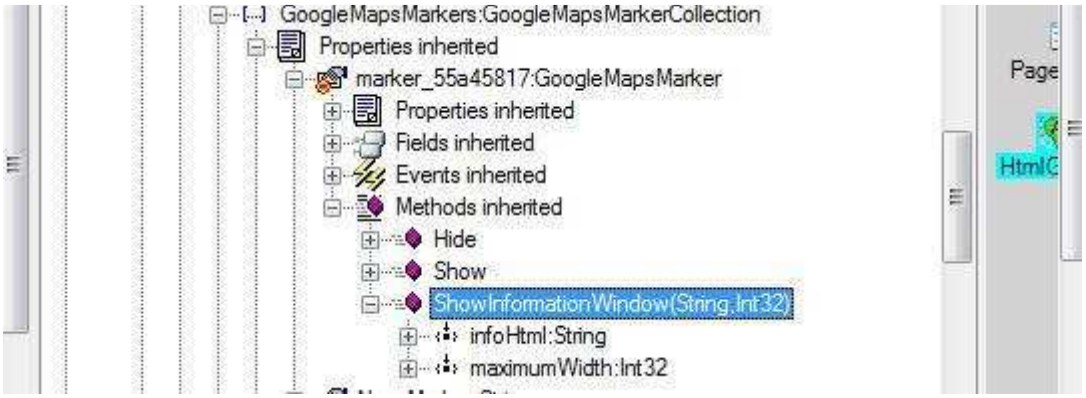
- **name** – You may give a name for the marker
- **title** – You may give a title for the marker



- **makerSize** – You may give a size for the marker

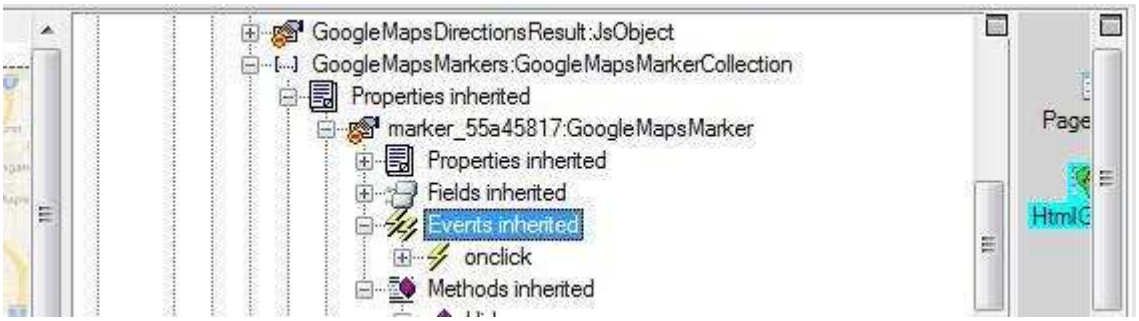
- **makerColor** – You may give a color for the marker if iconUrl is empty

Marker Methods



- **Hide** – make the marker invisible
- **Show** – make the marker visible
- **ShowInformationWindow** – display an information window for the mark.

Marker Events

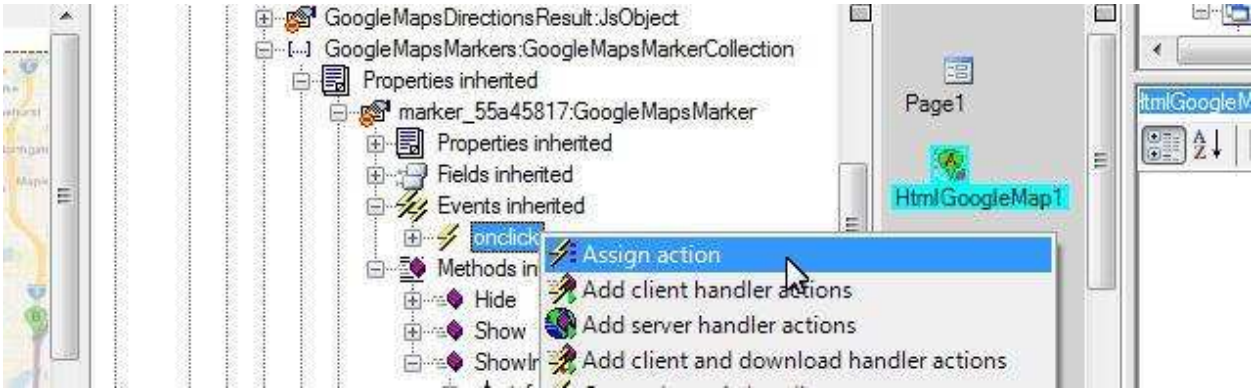


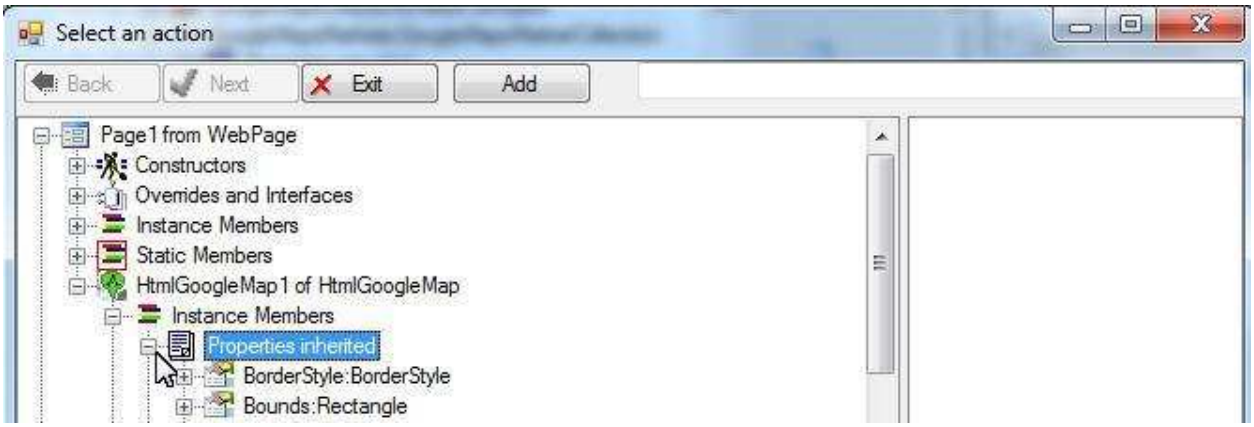
- **onclick** – occurs when the marker is clicked.

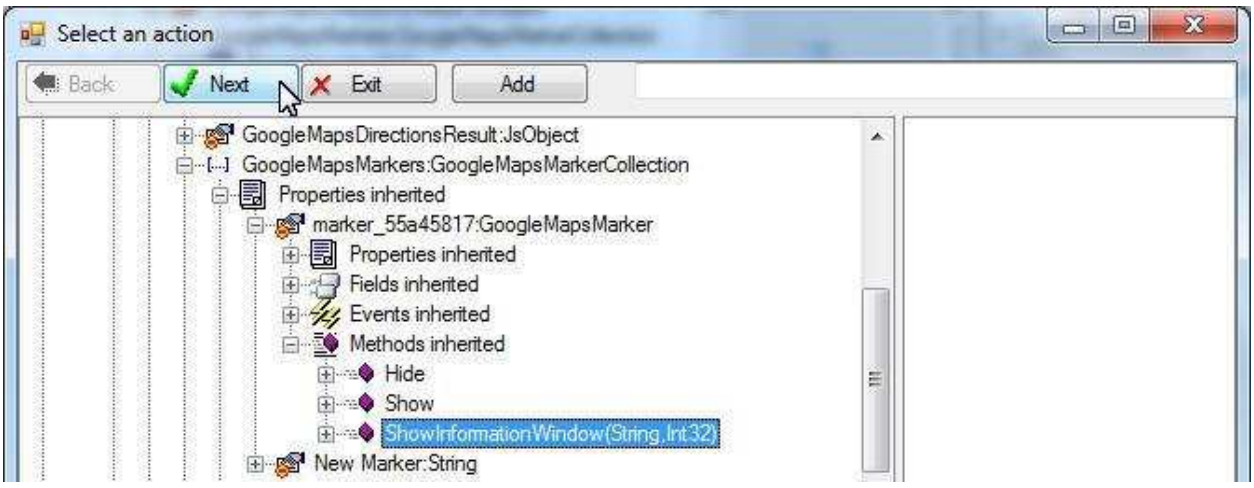
Handle marker click event at design time

A marker has an onclick event which occurs when the user clicks the marker.

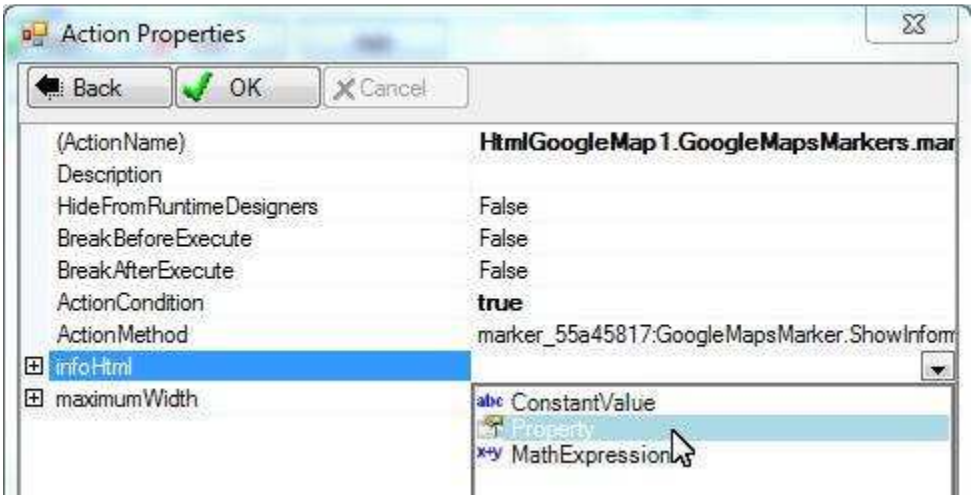
Suppose we want to display an information window when a marker is clicked. We may assign a ShowInformationWindow action to the onclick event.

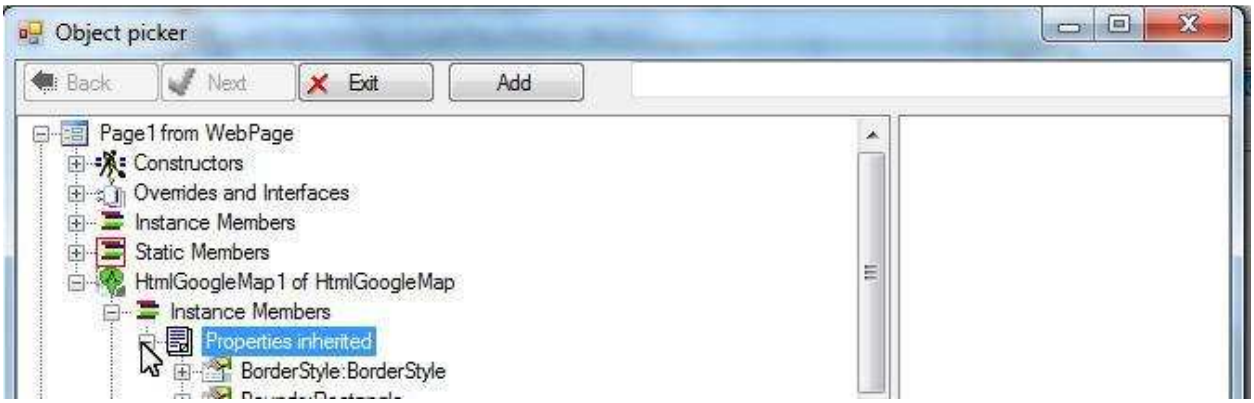


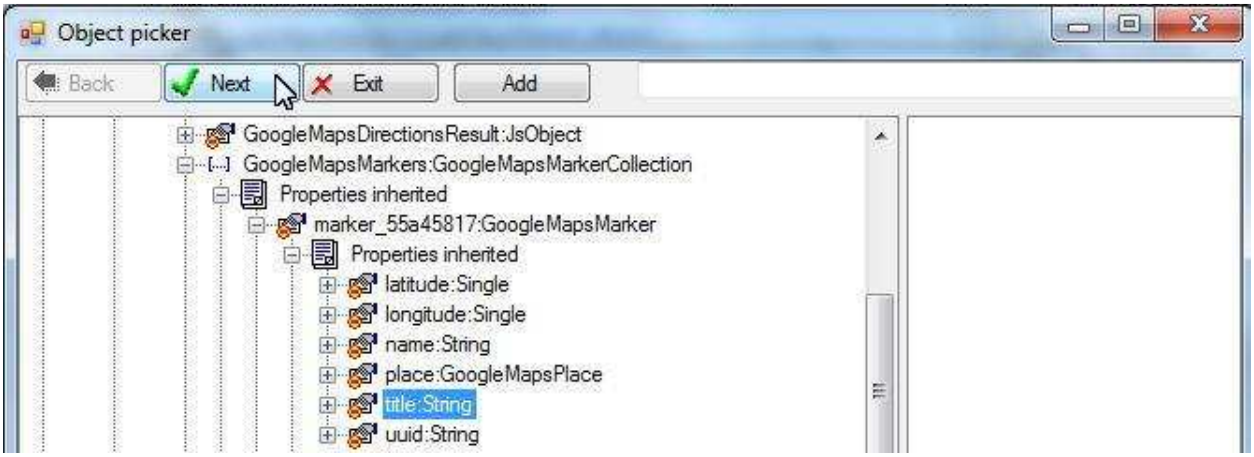




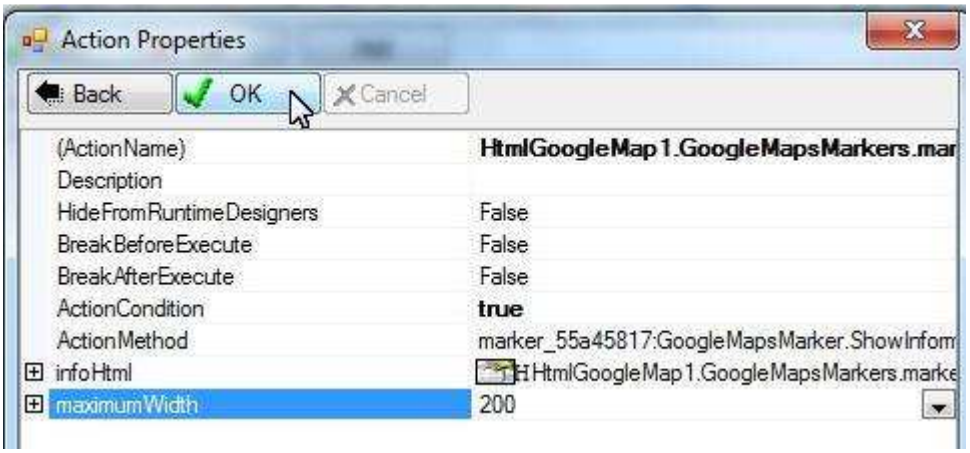
Let's show title of the marker:



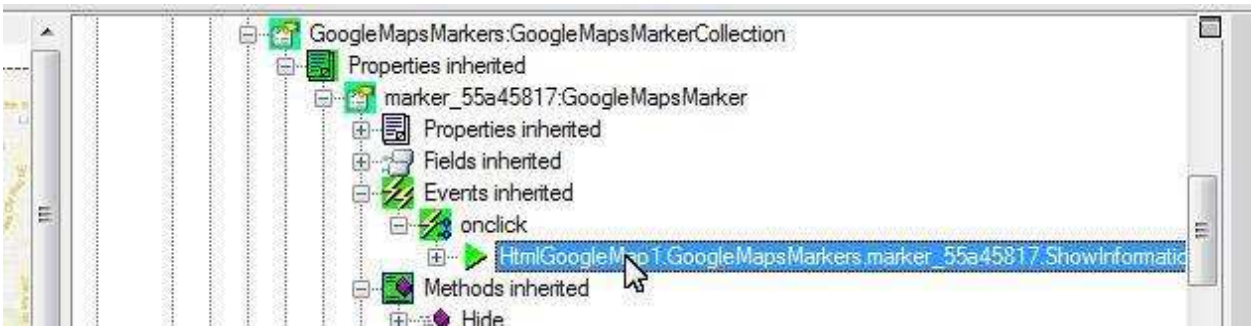




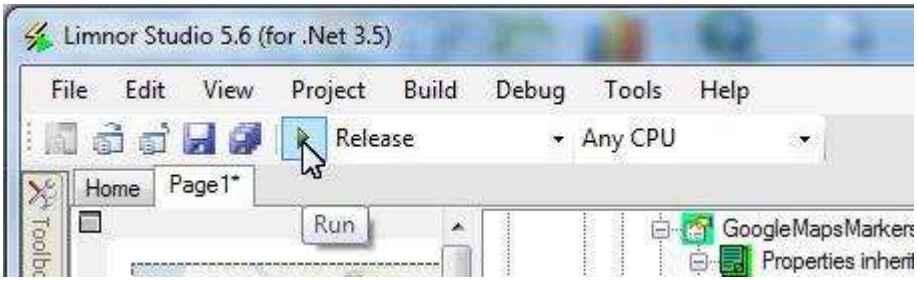
Set maximumWidth to 200. Click OK:



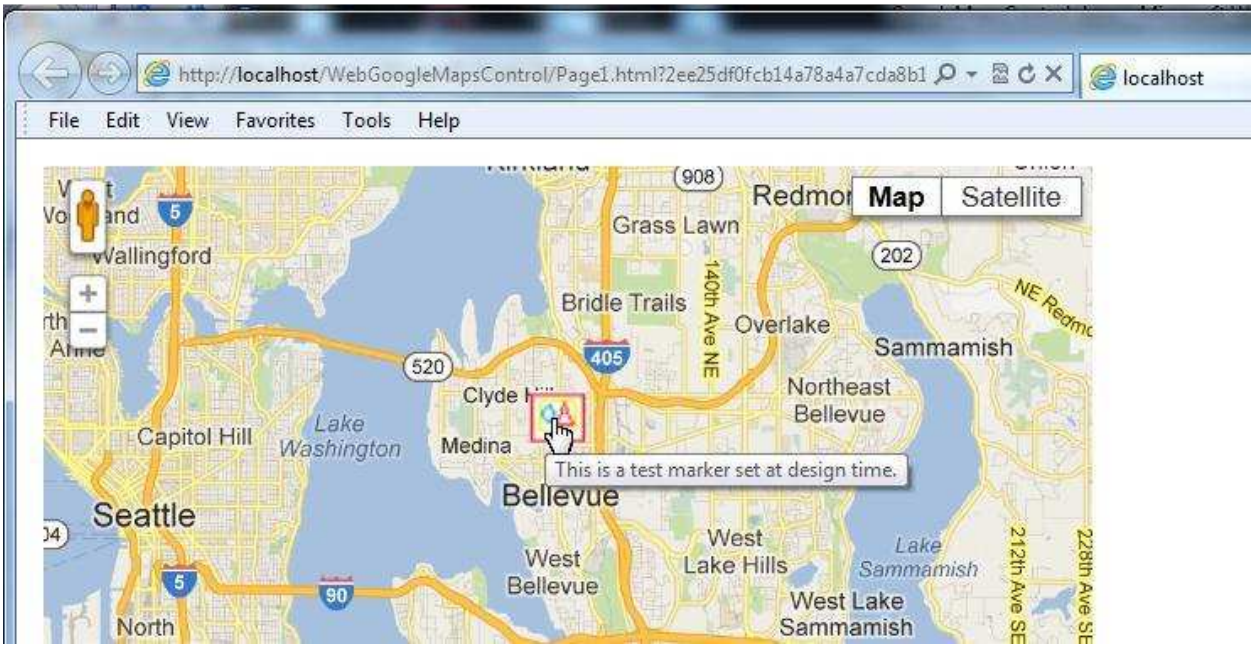
The action is created and assigned to the event:



We may test the web page:



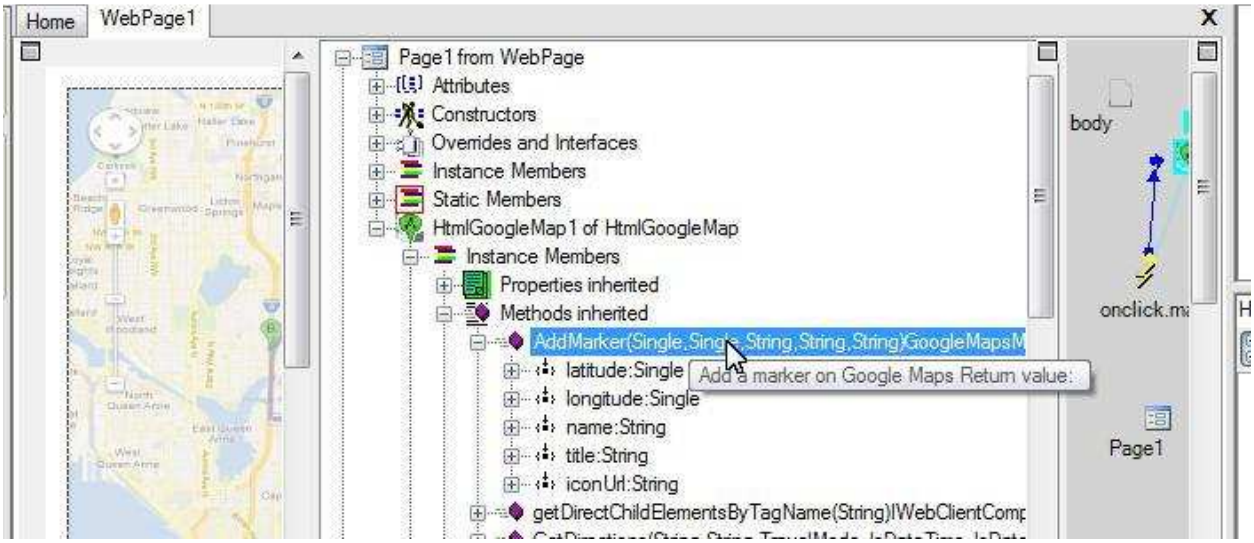
The web page is loaded and showing a Google map. We can see that a marker is on the map using an image we specified:



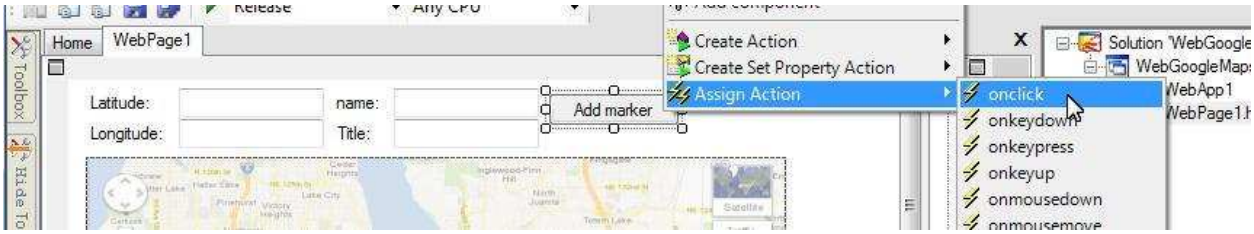
Click the marker, an information window appears:

Add markers at runtime

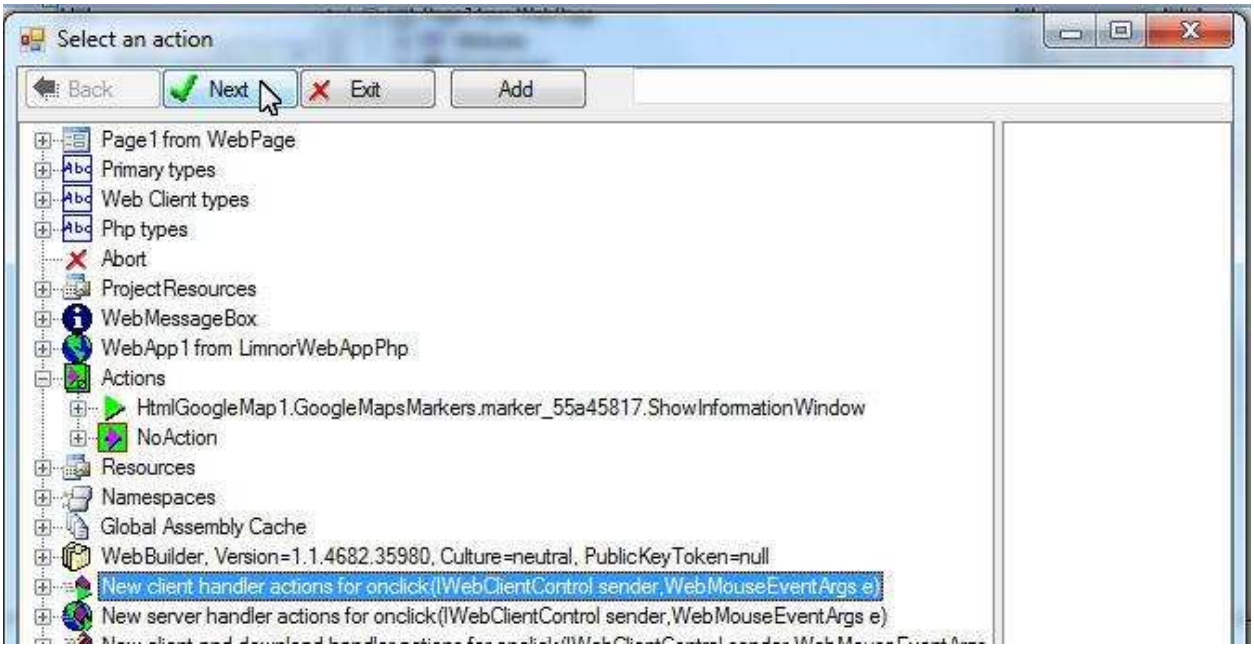
HtmlGoogleMap has an AddMarker for adding markers at runtime.



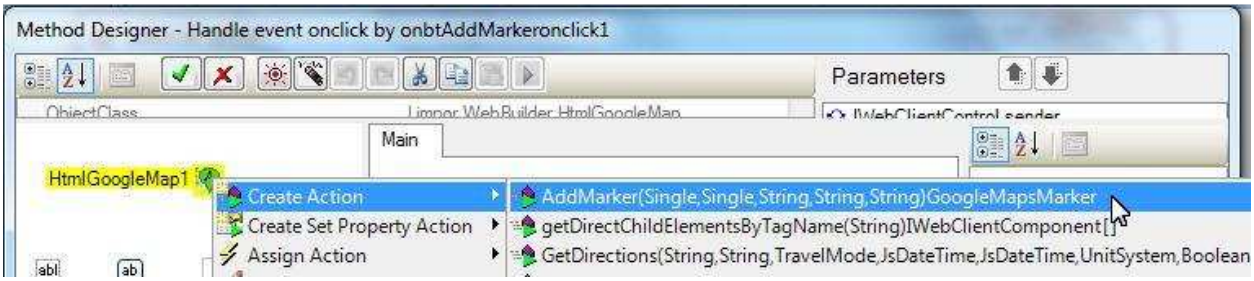
Let's use a button to trigger adding new markers:

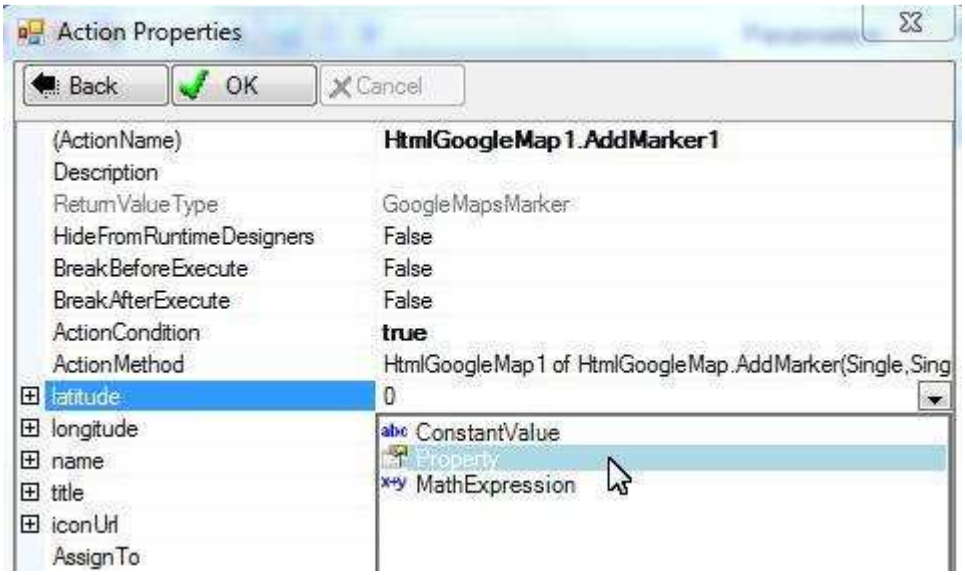


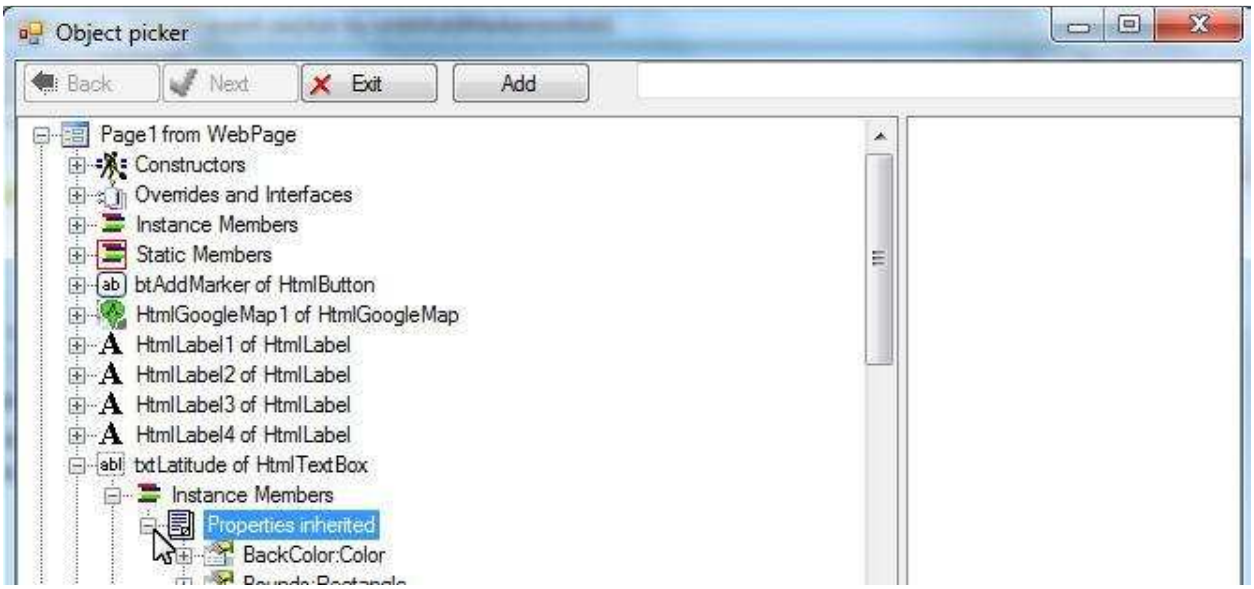
If we want to manipulate newly created marker then we may create an event handler method for the button:

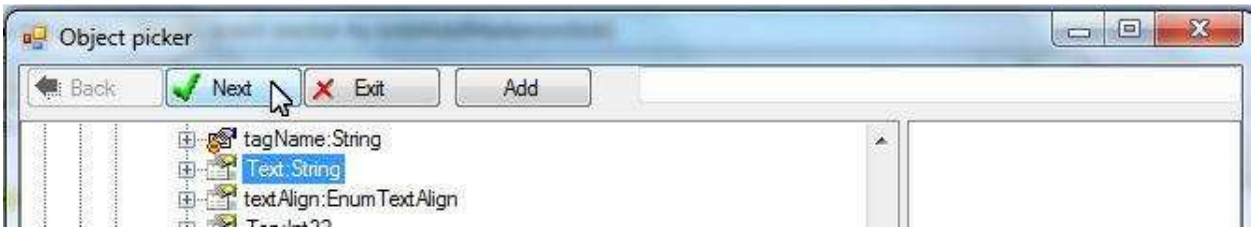


Create an AddMarker action:

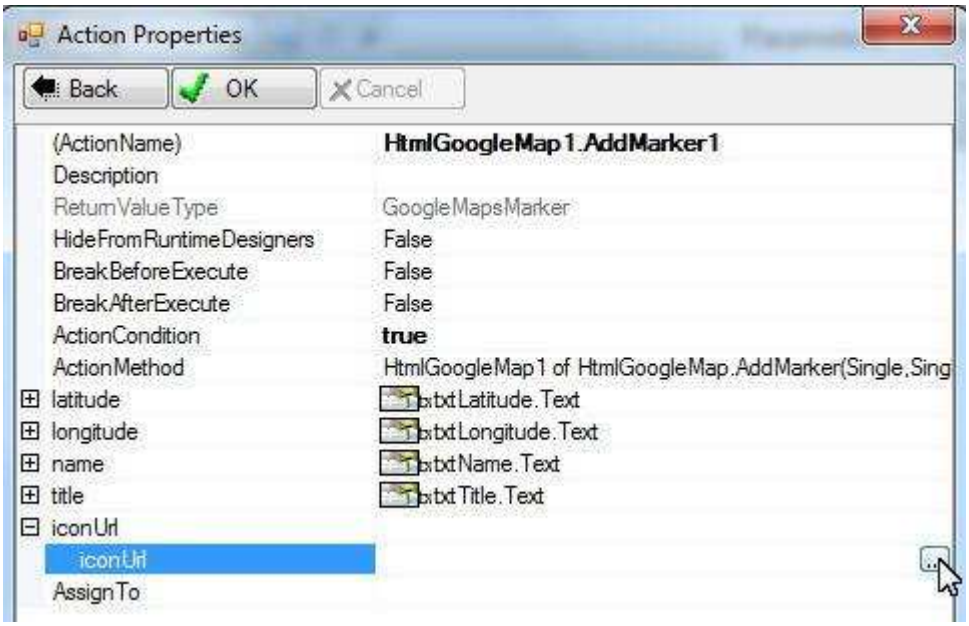

Method Designer - Handle event onclick by onbtAddMarkeronclick1
Parameters
Main
HtmlGoogleMap1
Create Action
Create Set Property Action
Assign Action
AddMarker(Single,Single,String,String,String)GoogleMapsMarker
getDirectChildElementsByTagName(String)IWebClientComponent[]
GetDirections(String,String,TravelMode,JsDateTime,JsDateTime,UnitSystem,Boolean

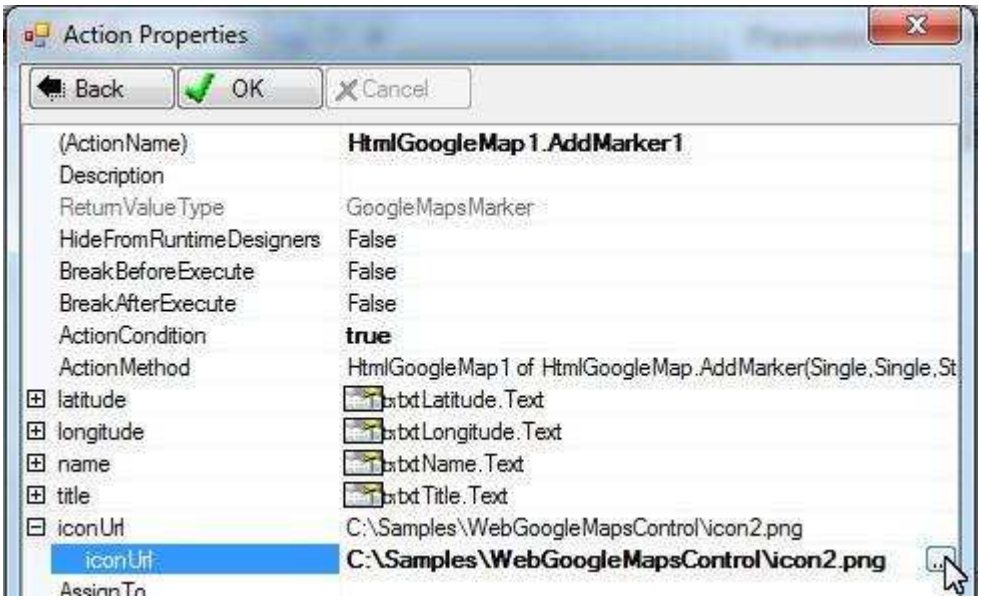

Action Properties
Back
OK
Cancel
(ActionName)
HtmlGoogleMap1.AddMarker1
Description
ReturnValueType
GoogleMapsMarker
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
HtmlGoogleMap1 of HtmlGoogleMap.AddMarker(Single,Sing
latitude
0
longitude
name
title
iconUrl
AssignTo
ConstantValue
Property
MathExpression


Object picker
Back
Next
Exit
Add
Page1 from WebPage
Constructors
Overrides and Interfaces
Instance Members
Static Members
btAddMarker of HtmlButton
HtmlGoogleMap1 of HtmlGoogleMap
HtmlLabel1 of HtmlLabel
HtmlLabel2 of HtmlLabel
HtmlLabel3 of HtmlLabel
HtmlLabel4 of HtmlLabel
txtLatitude of HtmlTextBox
Instance Members
Properties inherited
BackColor:Color

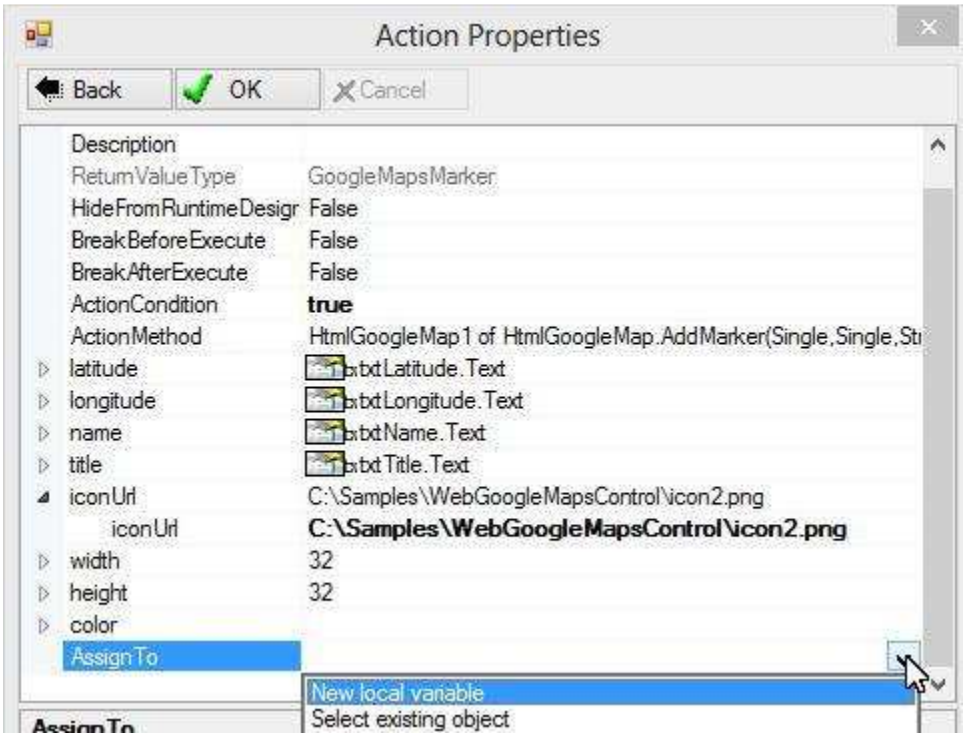


We may pass other text box values to the action. We may also select an image file for iconUrl:

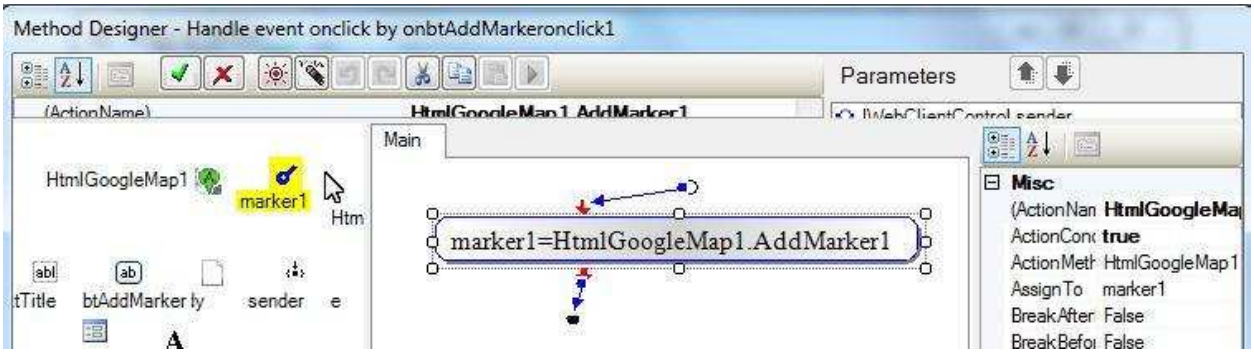




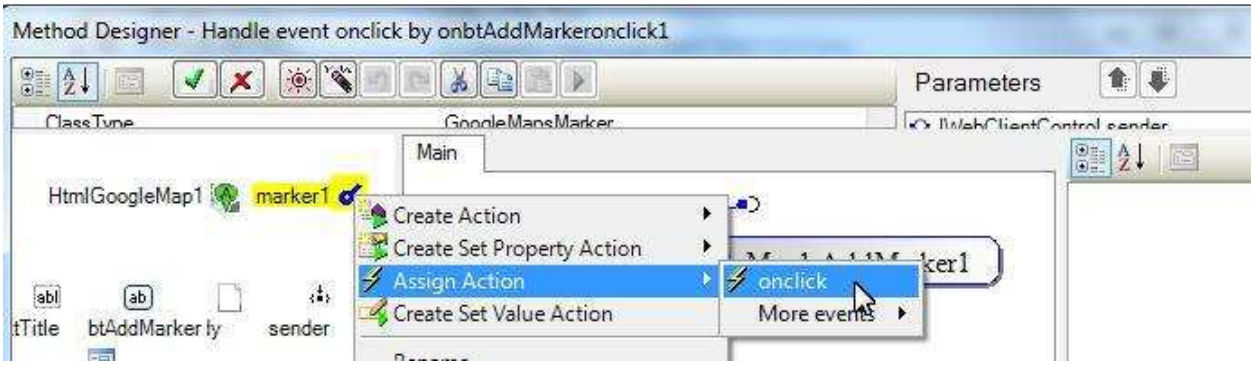
Set “AssignTo” to create a new local variable:



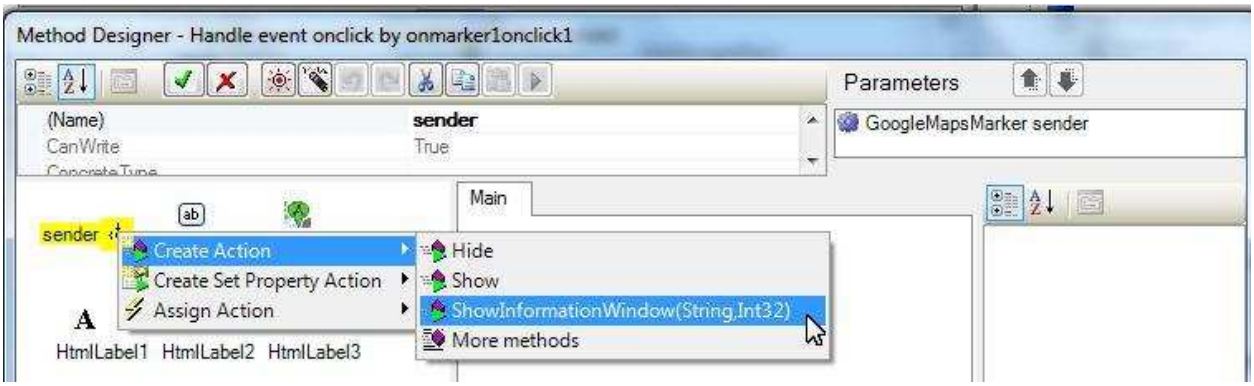
Click OK. The action is created. Note that variable marker1 is the marker added to web page:



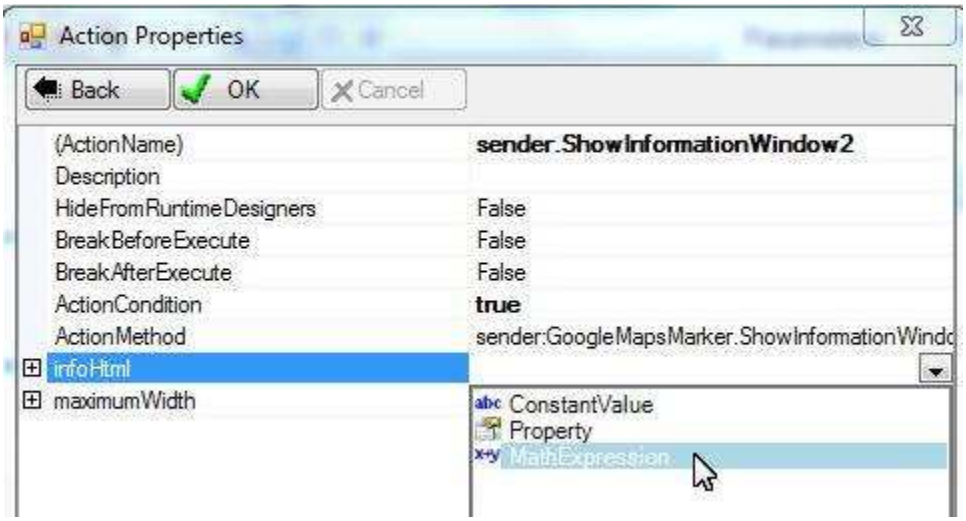
We may assign actions to onclick event of the newly created marker:

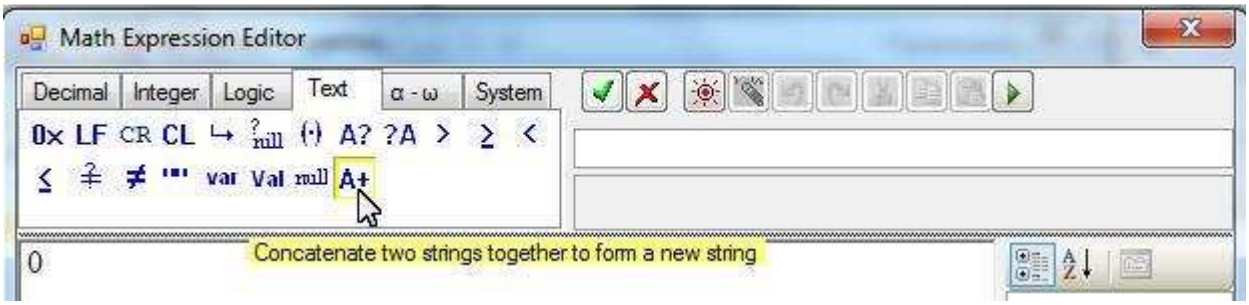


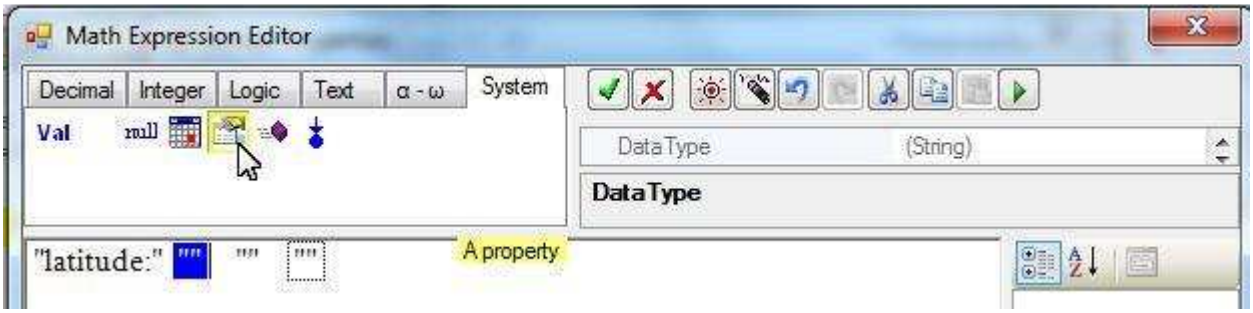
A method editor appears to let you add actions to be executed when onclick event occurs. Note that "sender" is the marker object which generated the onclick event. Suppose we want to show an information window at this event:

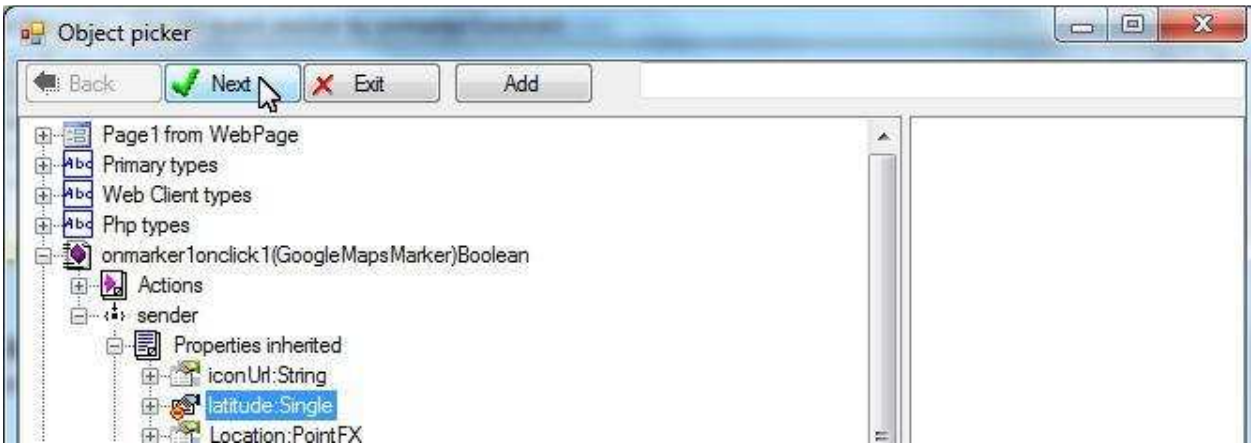


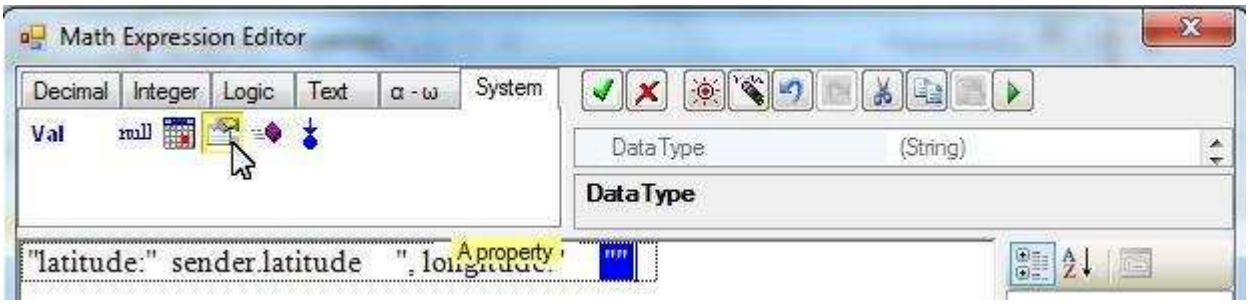
Suppose we want to show the location of the marker:

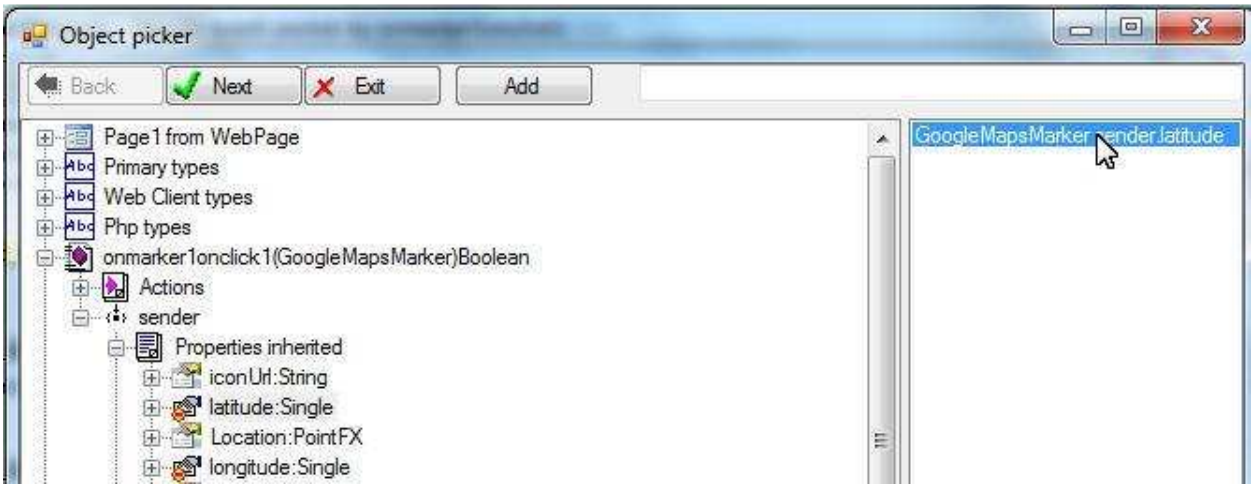


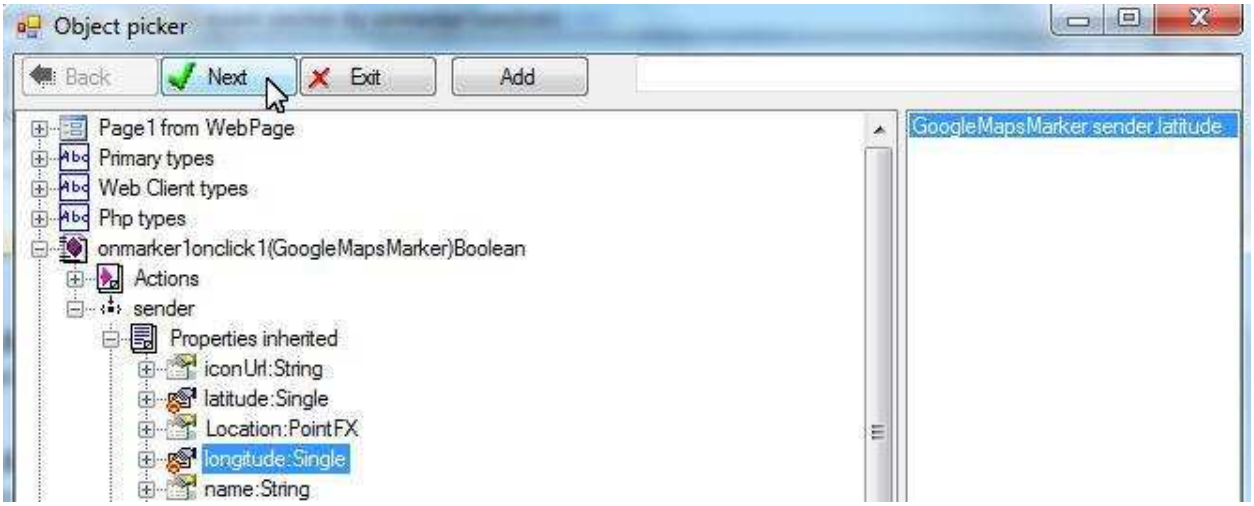

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Concatenate two strings together to form a new string

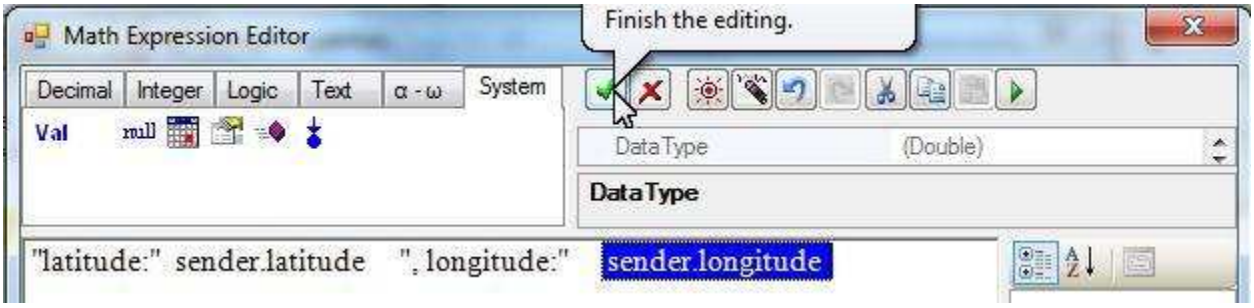

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(String)
DataType
"latitude:"
A property

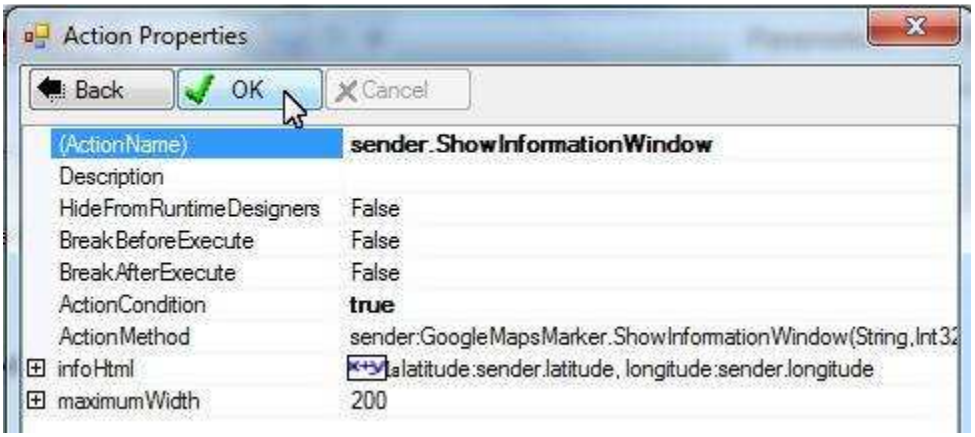

Object picker
Back
Next
Exit
Add
Page1 from WebPage
Primary types
Web Client types
Php types
onmarker1onclick1(GoogleMapsMarker)Boolean
Actions
sender
Properties inherited
iconUrl:String
latitude:Single
Location:PointFX


Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(String)
DataType
"latitude:"
sender.latitude
A property

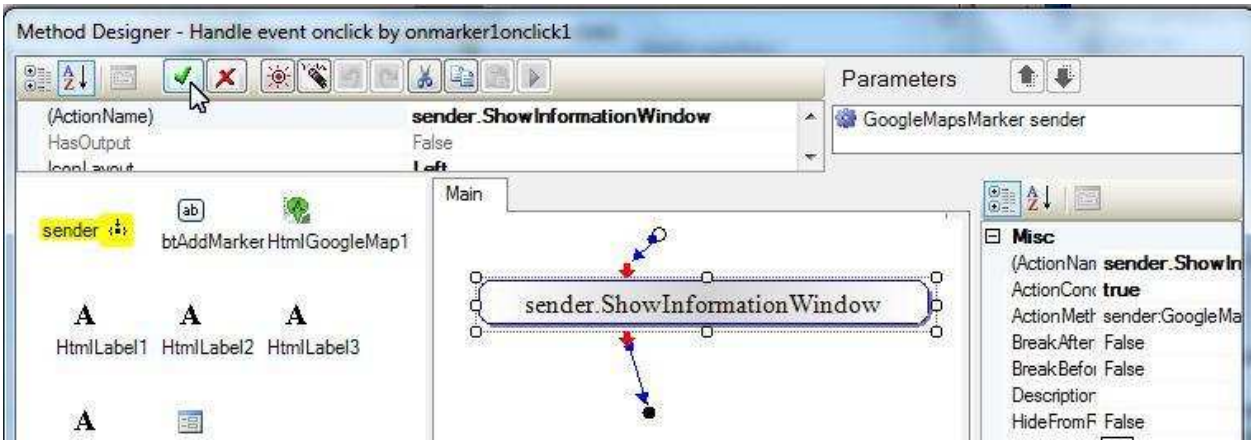

Object picker
Back
Next
Exit
Add
Page1 from WebPage
Primary types
Web Client types
Php types
onmarker1onclick1(GoogleMapsMarker)Boolean
Actions
sender
Properties inherited
iconUrl:String
latitude:Single
Location:PointFX
longitude:Single
GoogleMapsMarker.sender.latitude


Object picker
Back
Next
Exit
Add
Page1 from WebPage
Primary types
Web Client types
Php types
onmarker1onclick1(GoogleMapsMarker)Boolean
Actions
sender
Properties inherited
iconUrl:String
latitude:Single
Location:PointFX
longitude:Single
name:String
GoogleMapsMarker.sender.latitude

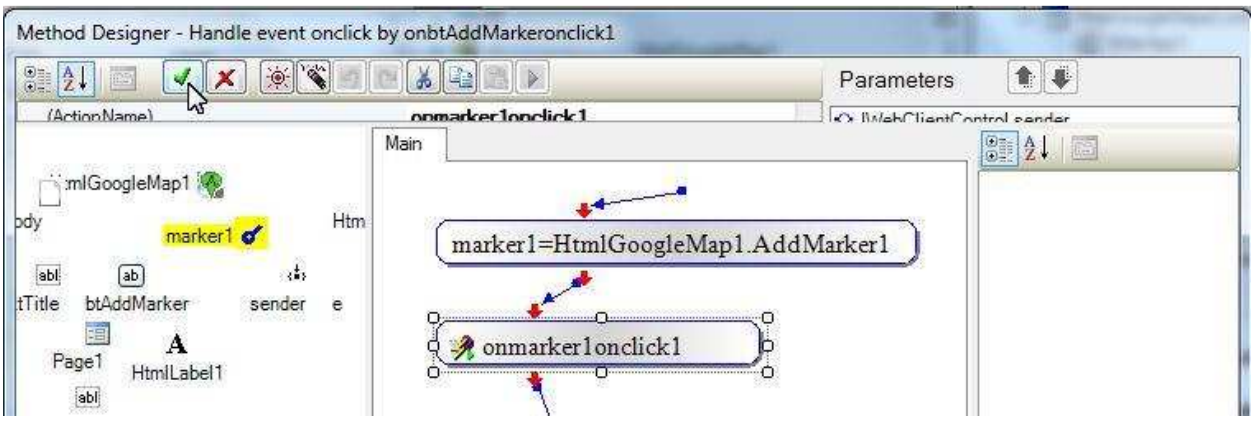

Math Expression Editor
Finish the editing.
Decimal
Integer
Logic
Text
α - ω
System
Val
nil
DataType
(Double)
DataType
"latitude:" sender.latitude ", longitude:" sender.longitude



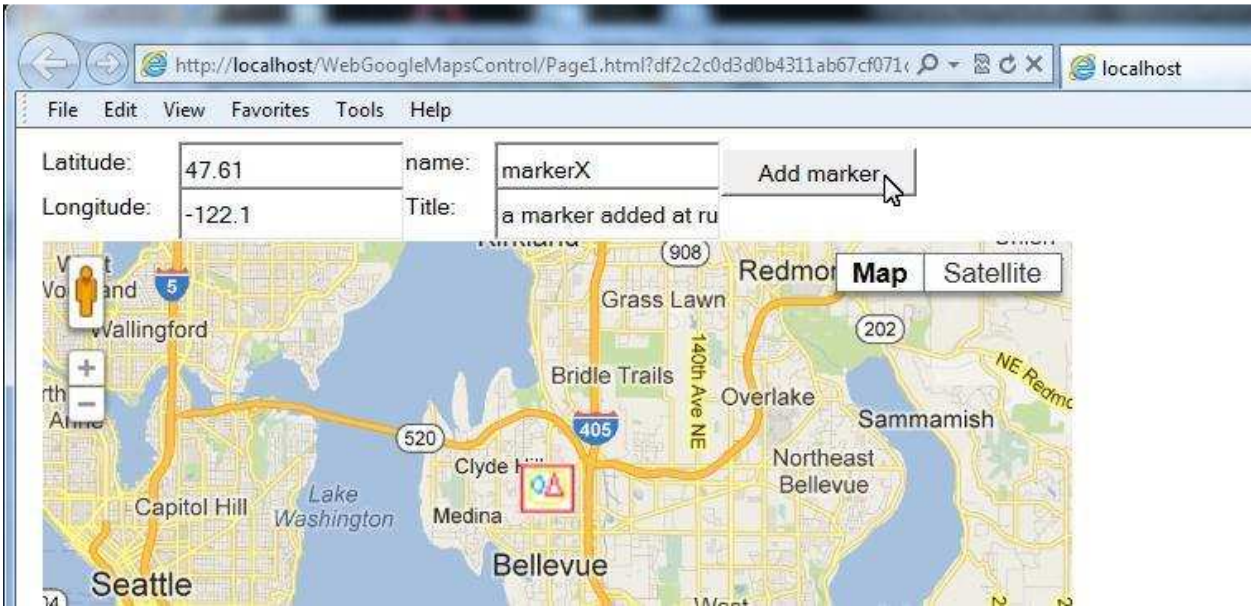
For this sample, we just use this one action:



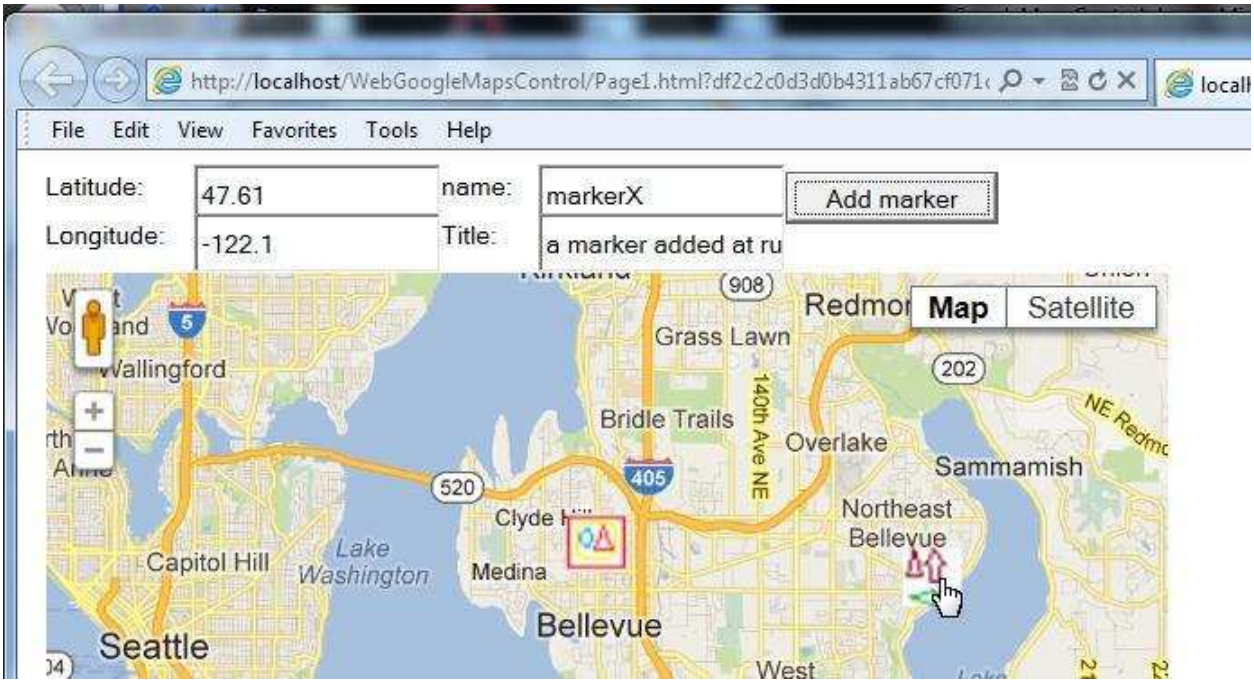
For this sample, that is all we need to manipulate the newly created marker:



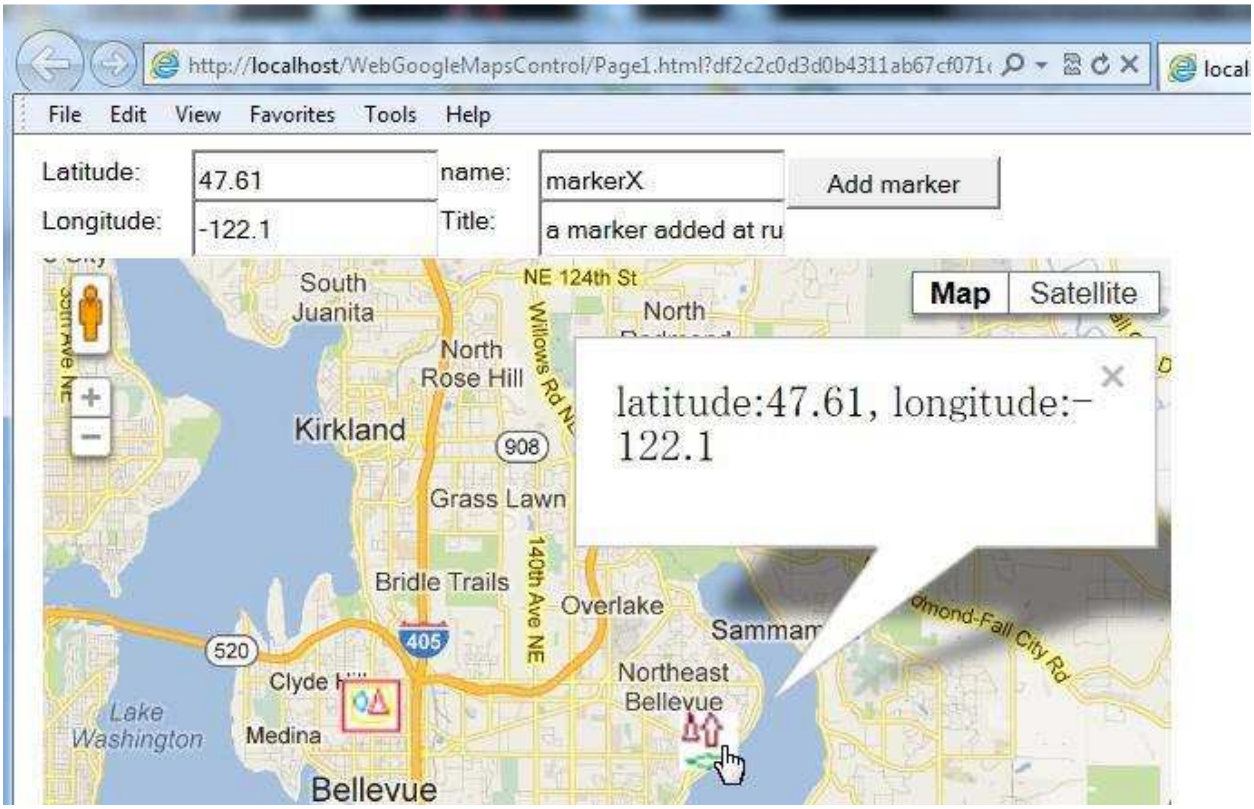
Let's launch the web page to try it. Enter parameters for a new marker. Click "Add marker":



A new marker appears. Click on the new marker:



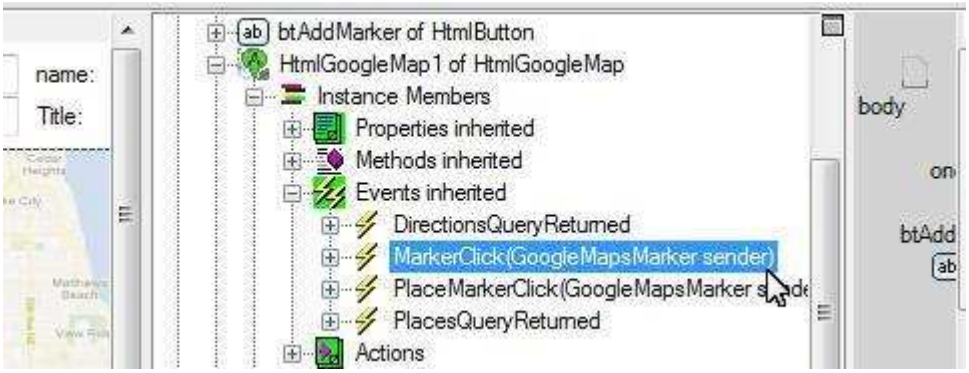
An information window appears showing marker location:



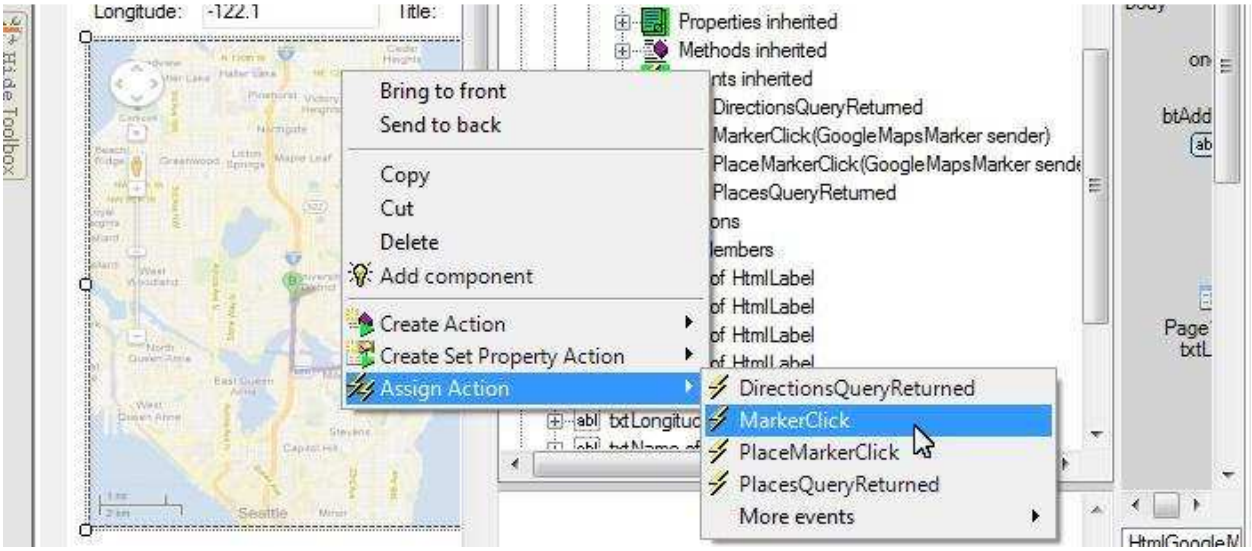
Handle event for all markers

In previous samples, we handled onclick event for each individual marker, added both at design time and runtime.

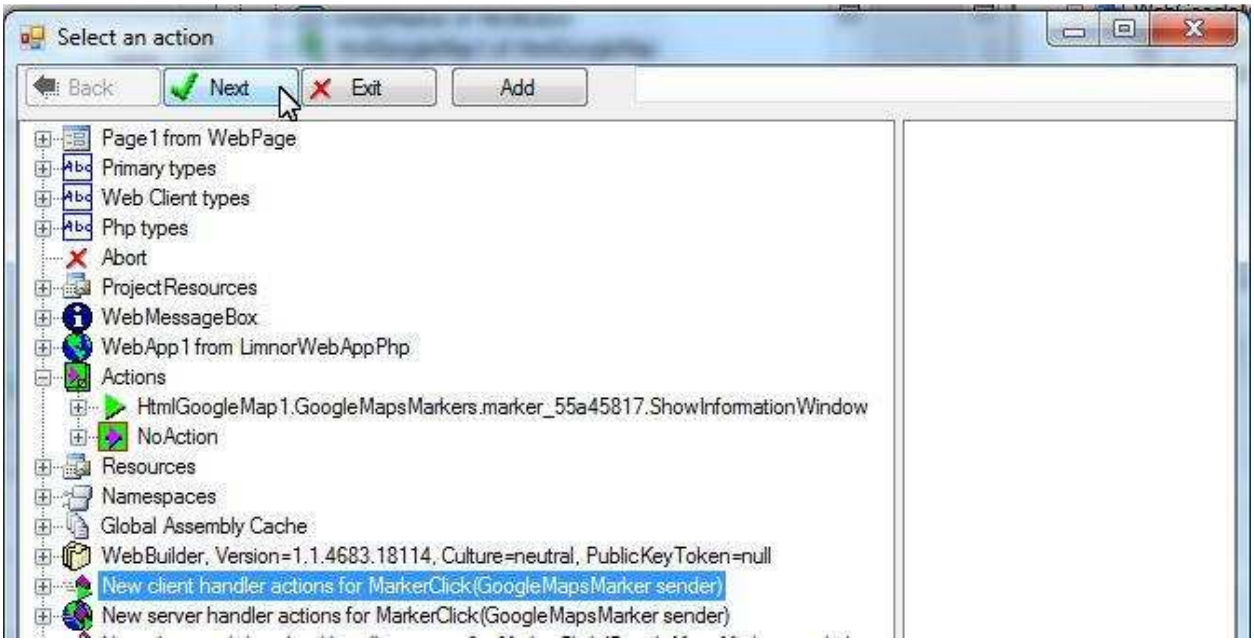
If your event handling logic is similar for all markers then you do not need to add event handling logic to each marker one by one. HtmlGoogleMap has an event, `MarkerClick`, which occurs when any marker is clicked on.



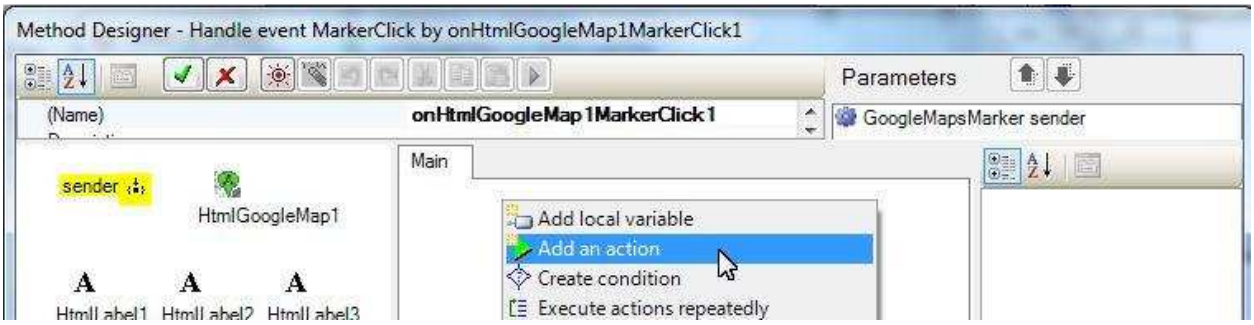
Suppose we want to show a message box displaying marker name and title when any marker is clicked:

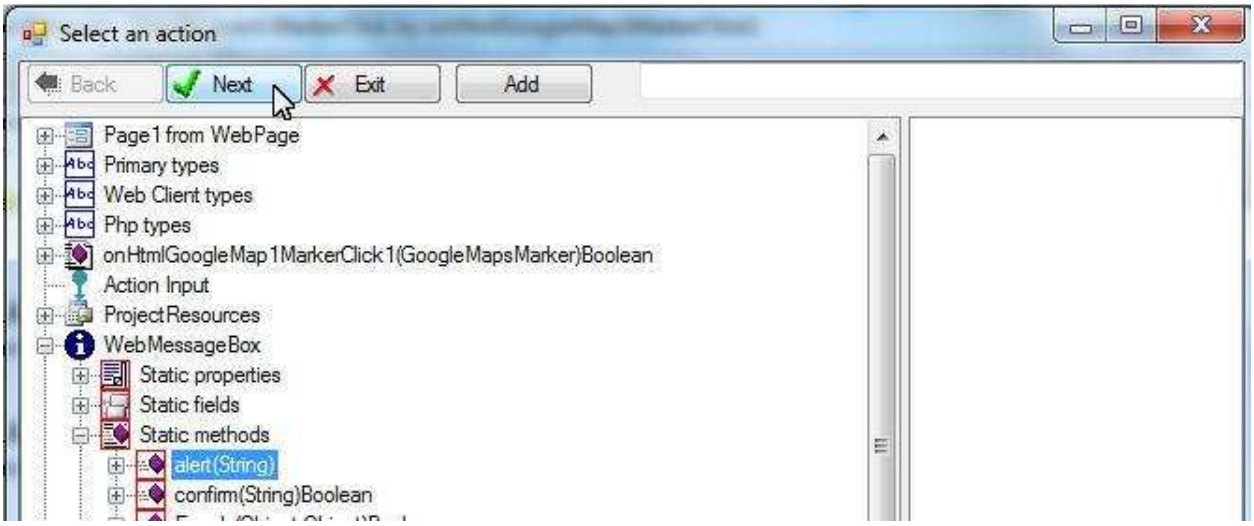


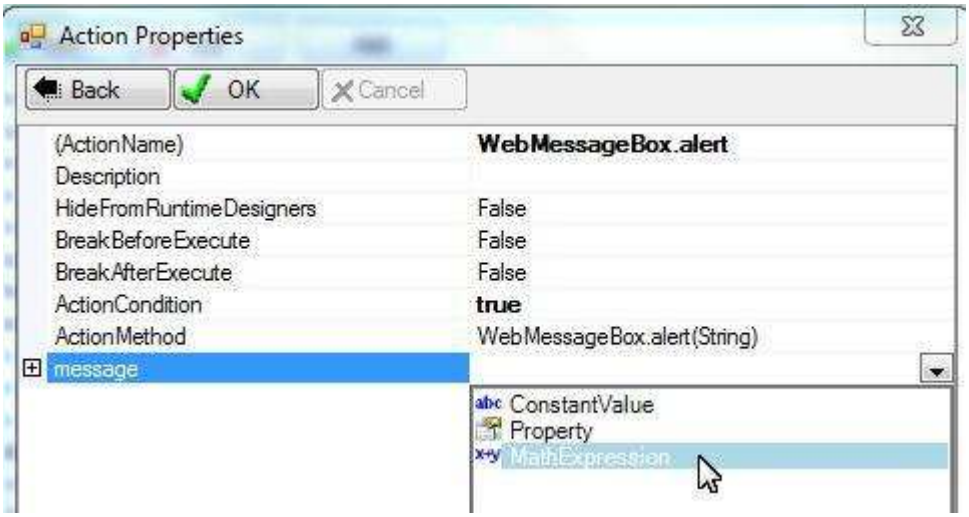
Because we want to access the event sender, we need to create an event handler method:

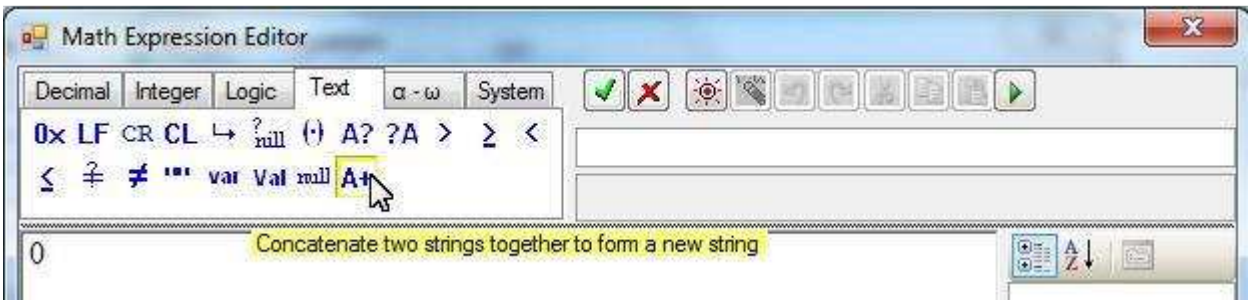


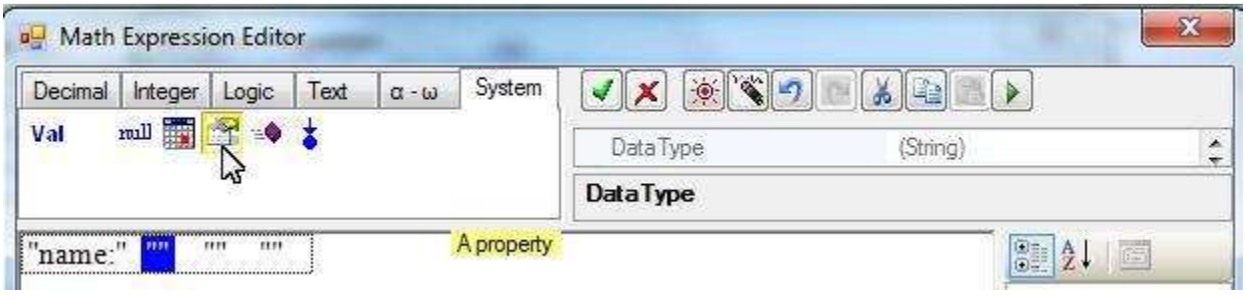
A method editor appears to let you add actions. Let's add a message box action:

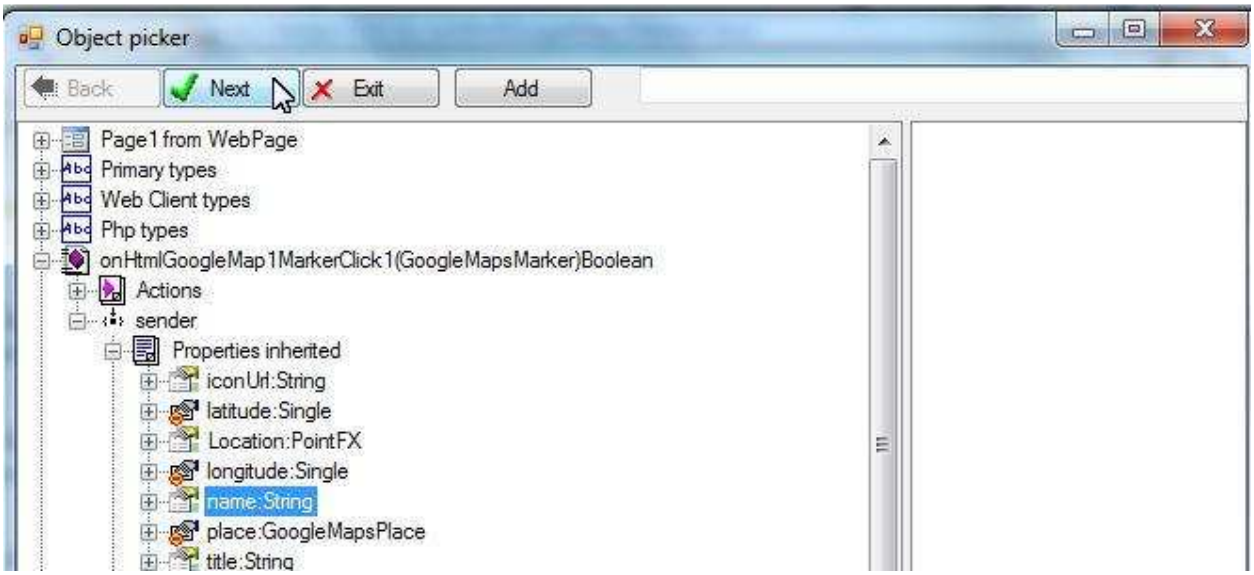


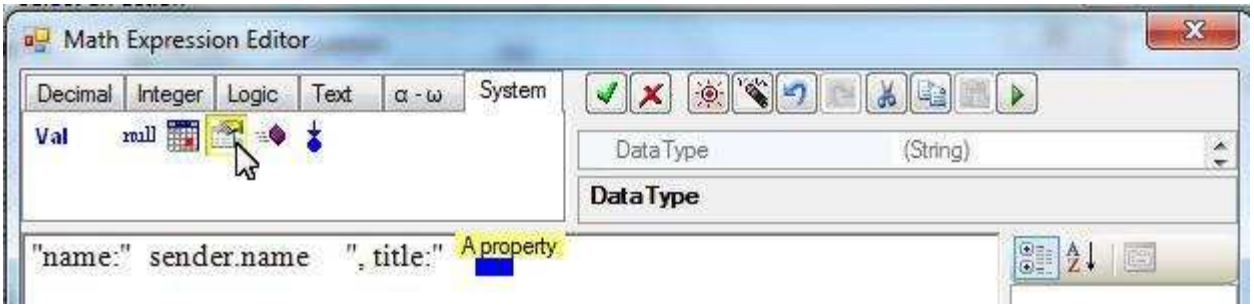

Select an action
Back
Next
Exit
Add
Page1 from WebPage
Primary types
Web Client types
Php types
onHtmlGoogleMap1MarkerClick1(GoogleMapsMarker)Boolean
Action Input
Project Resources
WebMessageBox
Static properties
Static fields
Static methods
alert(String)
confirm(String)Boolean


Action Properties
Back
OK
Cancel
(ActionName)
WebMessageBox.alert
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
WebMessageBox.alert(String)
message
ConstantValue
Property
MathExpression

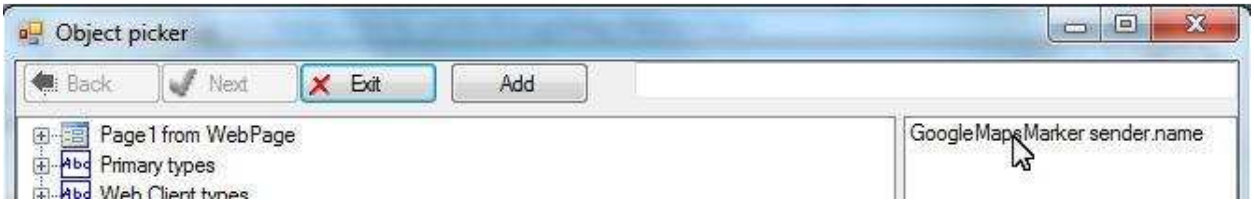

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Concatenate two strings together to form a new string
0

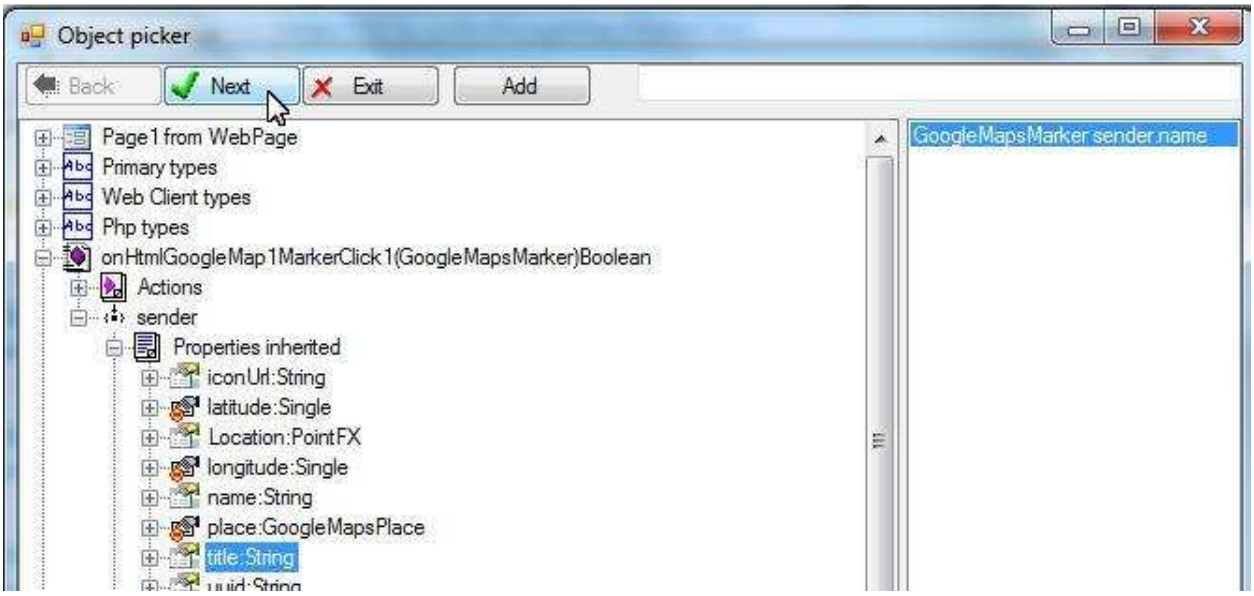

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(String)
DataType
"name:"
A property

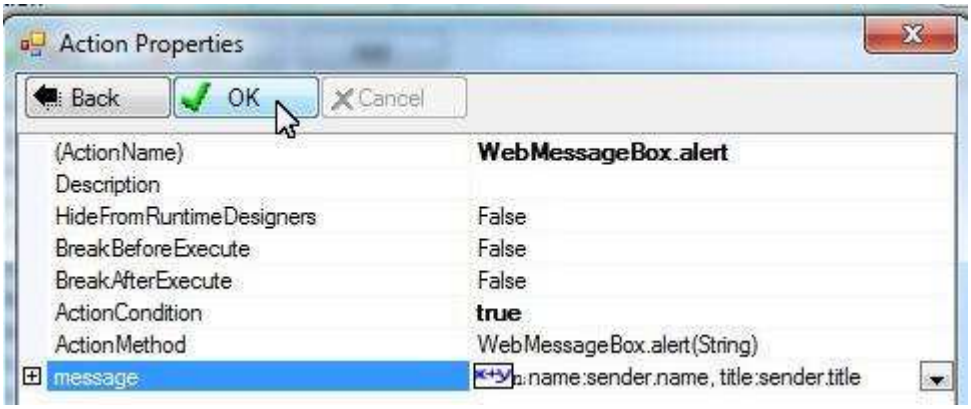




Select a recent selection to help us quickly locate a property:

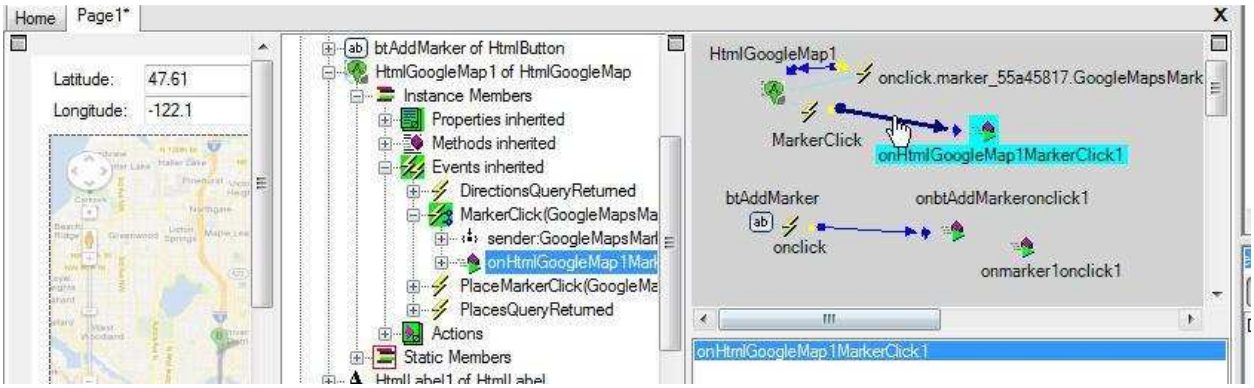




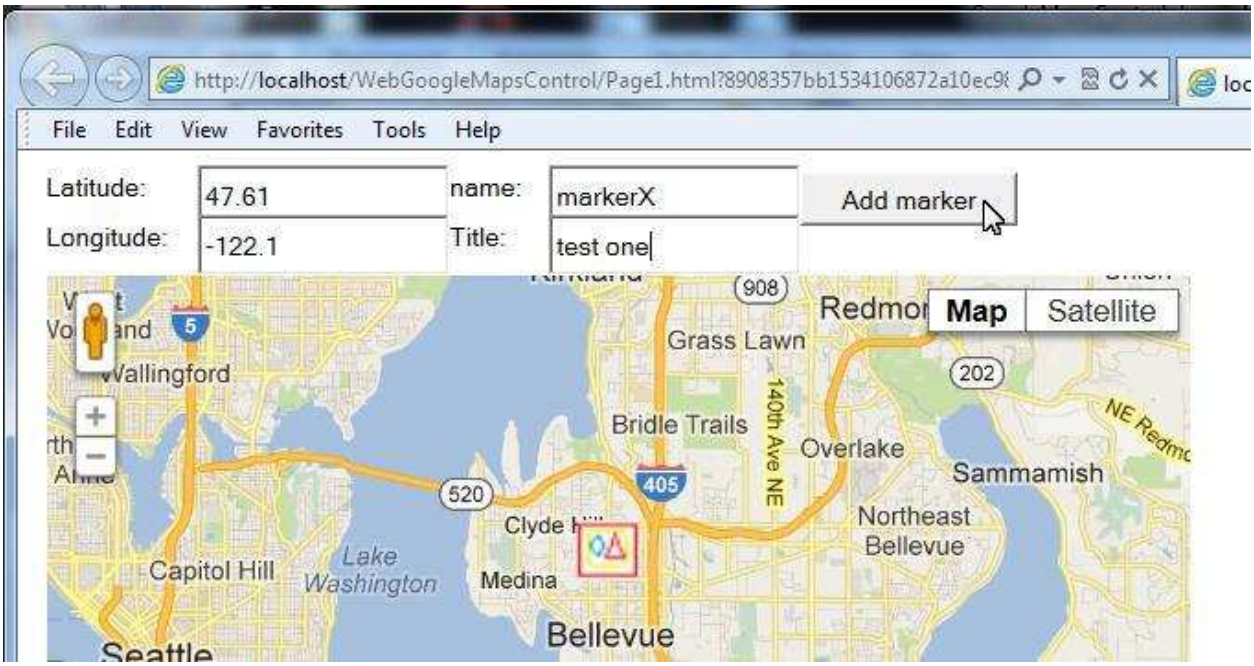


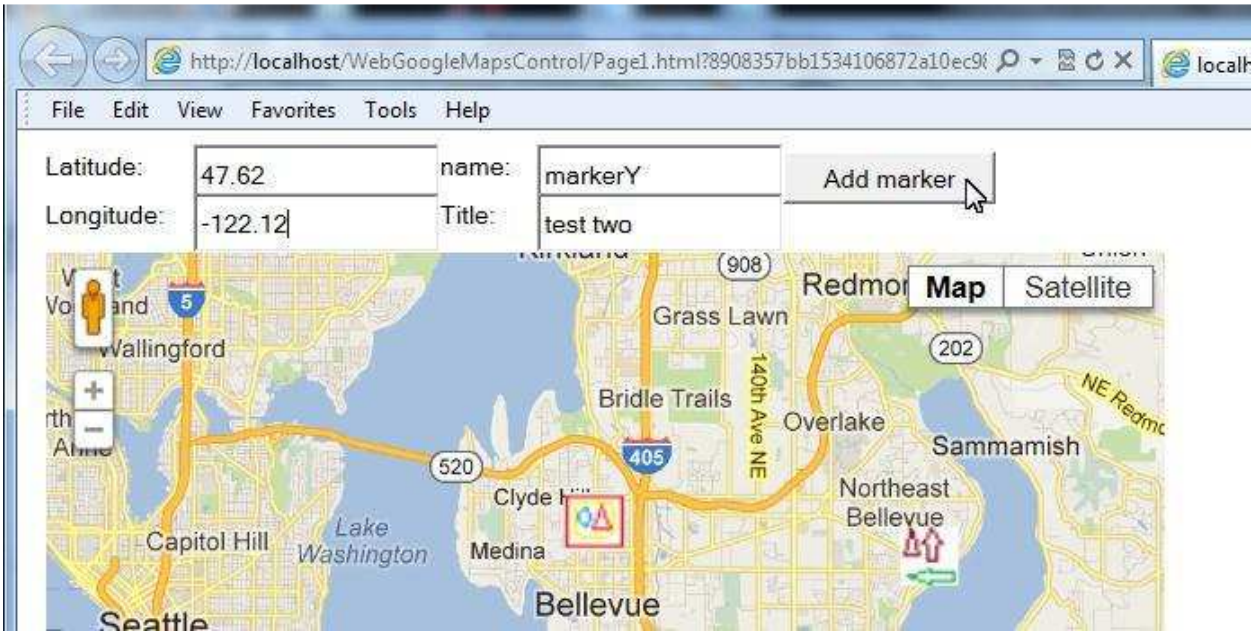
For this sample, we just use this one action on this event:

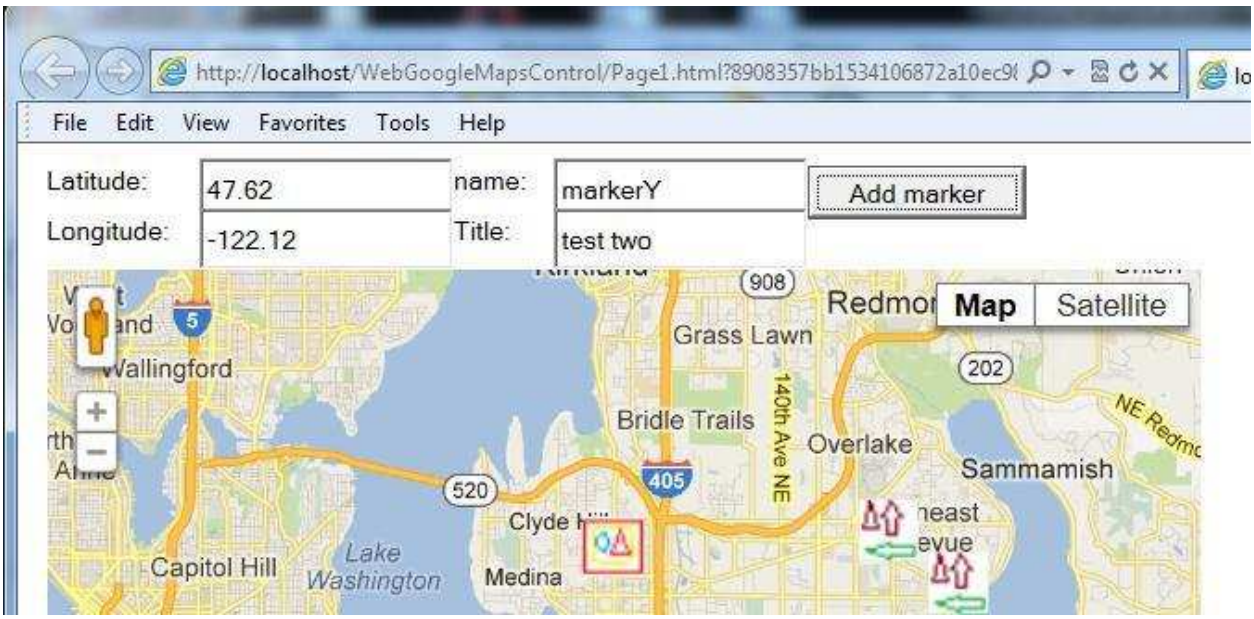
The event handler method is created:



Let's launch the web page to try it. The web page appears with one marker added at design time. We may add more markers:

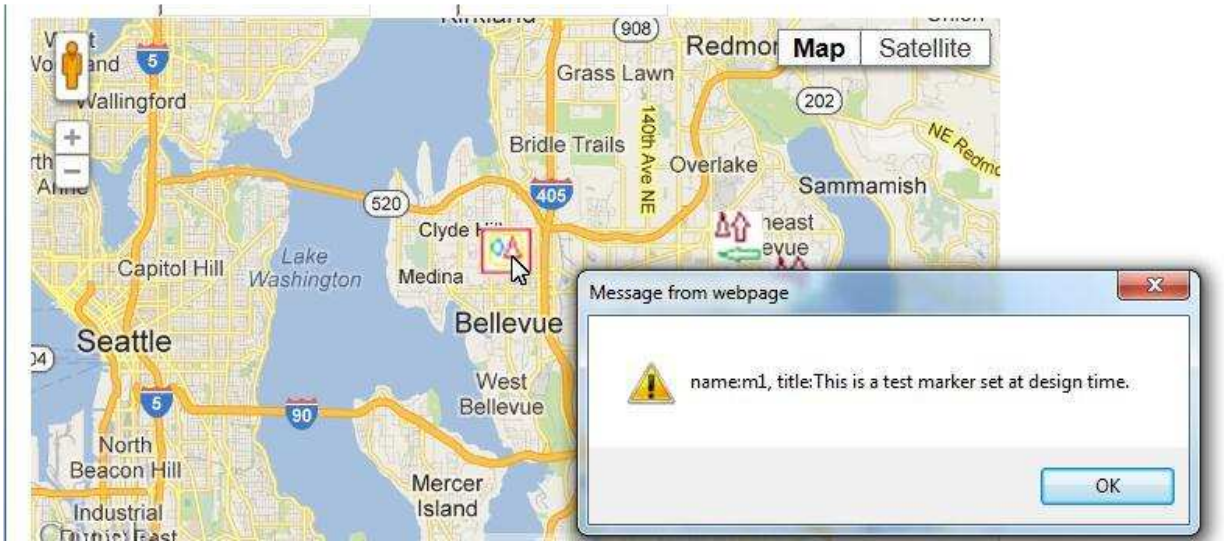






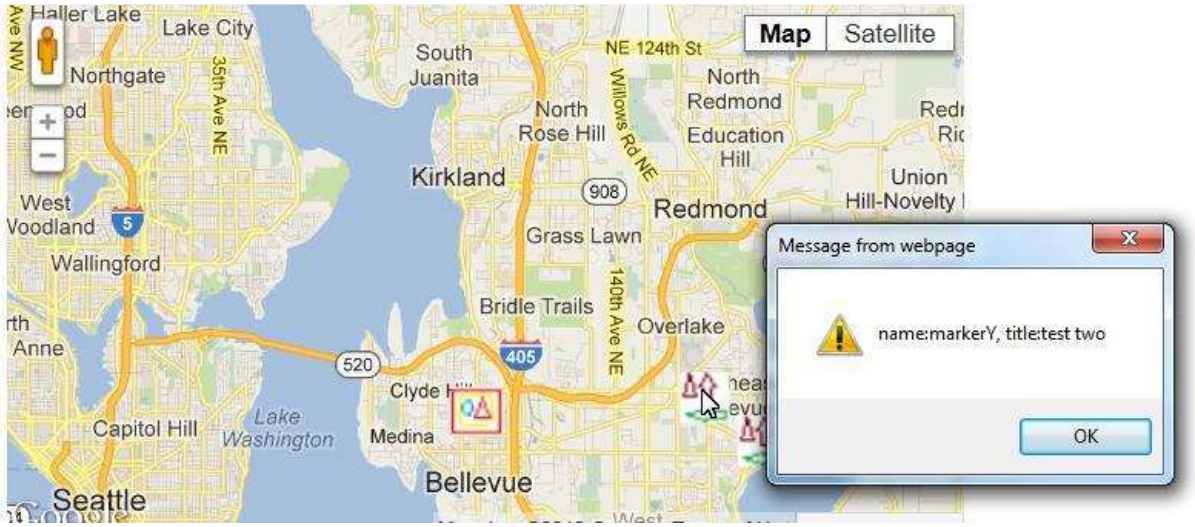
Click on each marker to see what will happen.

Click on the marker added at design time. A message box appears showing name and title of the marker, as the result of handling MarkerClick event of HtmlGoogleMap control:



Close the message box. An information window appears showing title of the marker, as the result of handling onclick event of the marker:

Click a marker we added at runtime. A message box appears showing name and title of the marker, as the result of handling MarkerClick event of HtmlGoogleMap control:



Close the message box. An information window appears showing marker location, as the result of handling onclick event of the marker:

Marker Related Methods

HtmlGoogleMap has following marker-methods:

- **`AddMarker`** – Add a new marker
- **`HideAllMarkers`** – Make all markers invisible
- **`ShowAllMarkers`** – Make all markers visible
- **`RemoveAllMarkers`** – Remove all markers

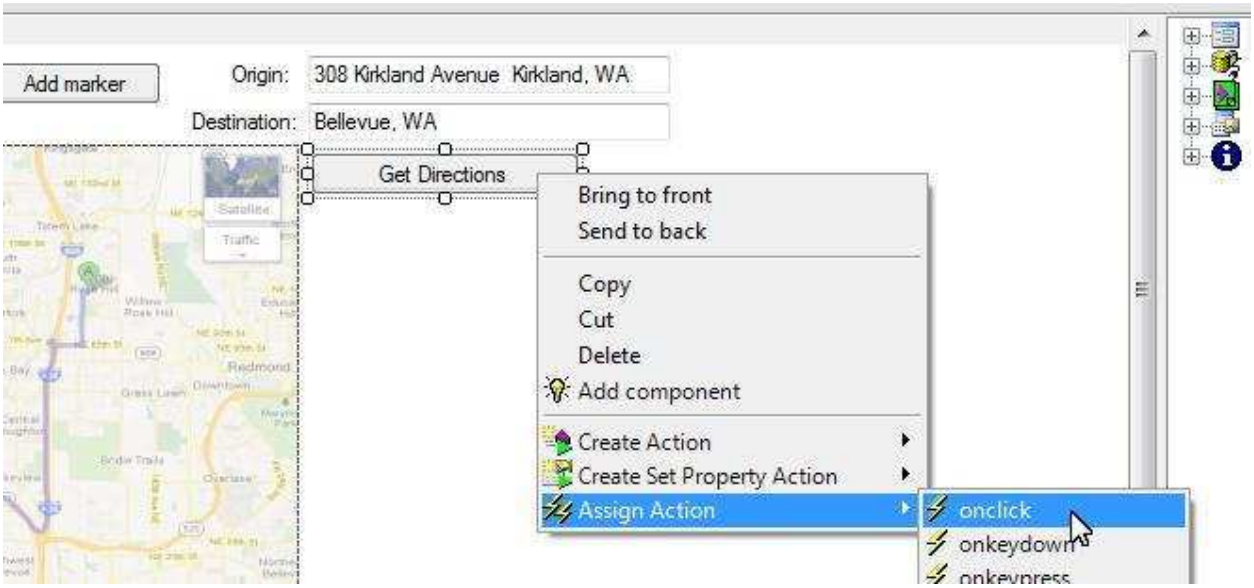
Get Directions

An HtmlGoogleMap control uses following procedure to get directions.

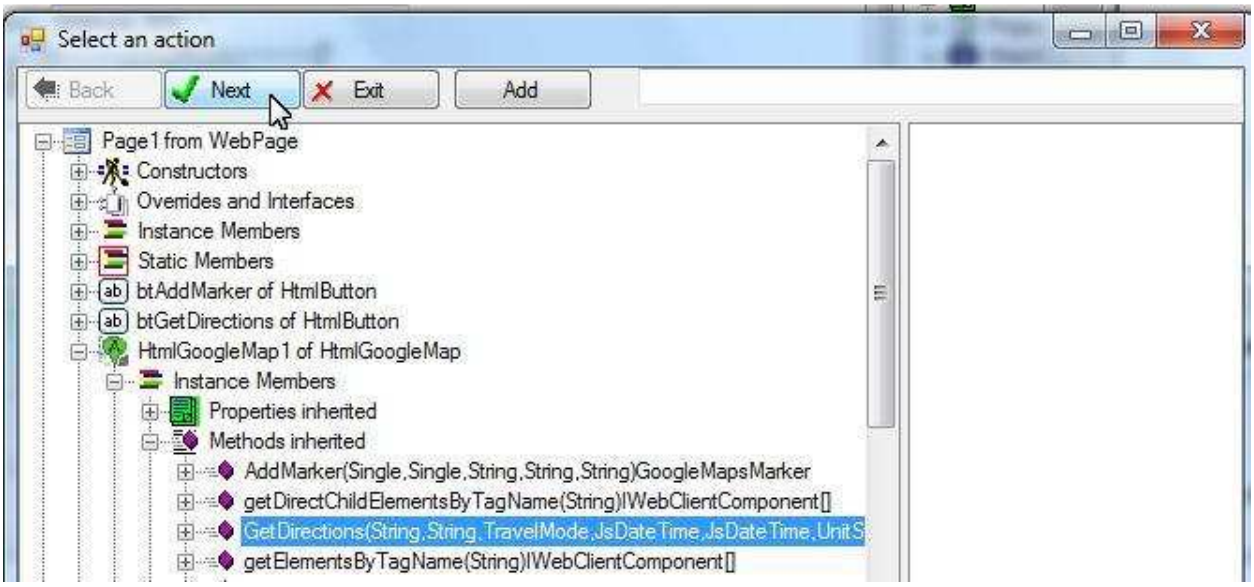
- A **`GetDirections`** action requests directions from Google Maps
- **`DirectionsQueryReturned`** event occurs when Google Maps responds to the request.
- **`GoogleMapsDirectionsQuerytStatus`** property indicates how Google Maps responded to the request.
- **`GoogleMapsDirectionsResult`** property contains all routes of directions.
- **`RouteCount`** property indicates how many routes returned.
- **`FirstRoute`** property contains the first route if RouteCount is larger than 0.

Create GetDirections action

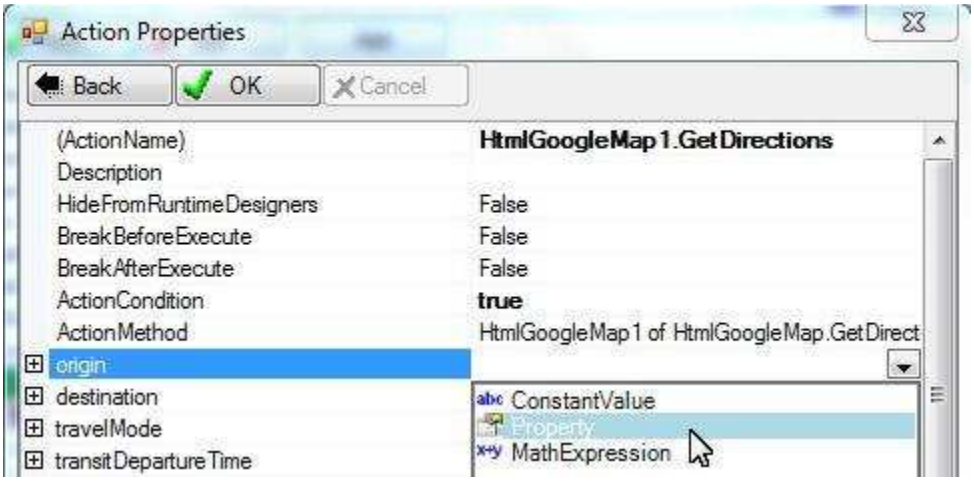
We use a button to initiate a get-directions action:

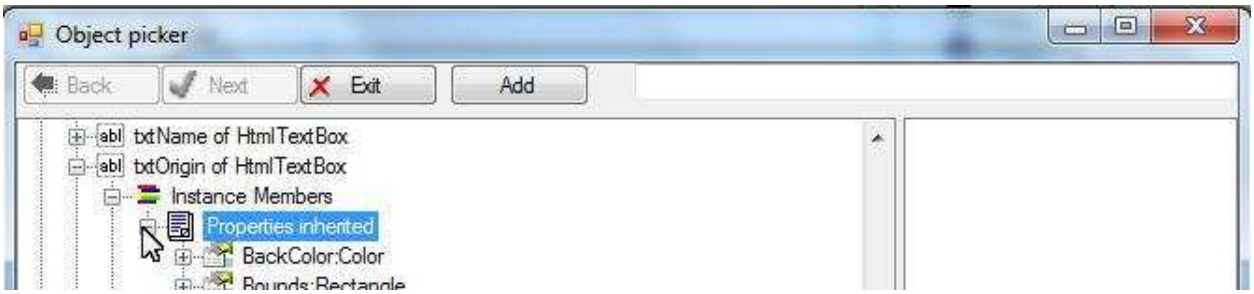


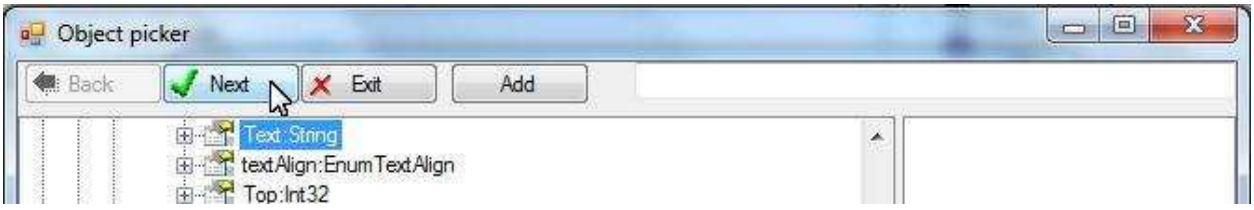
Select GetDirections method:

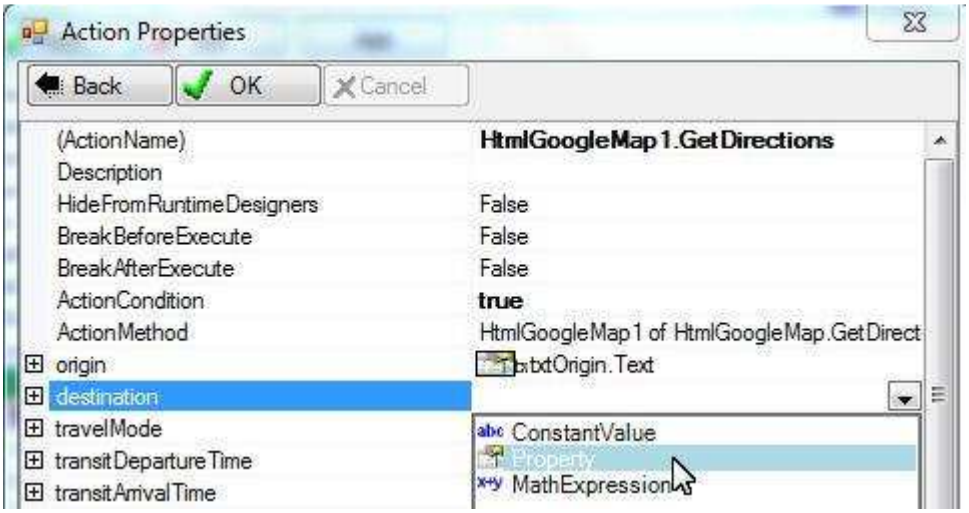


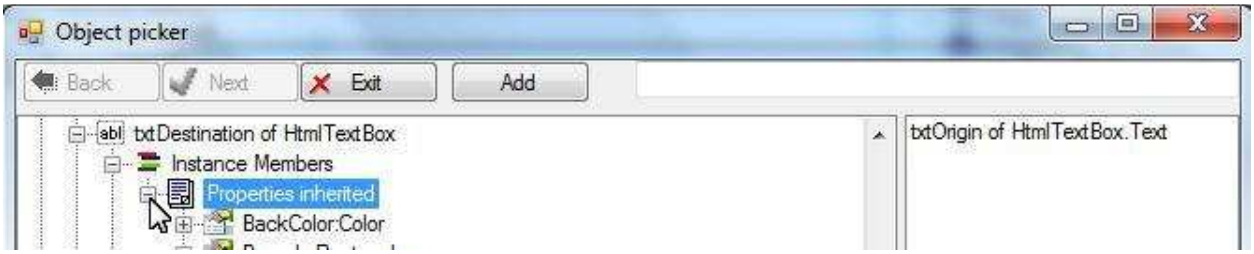
Use text box values for origin and destination:

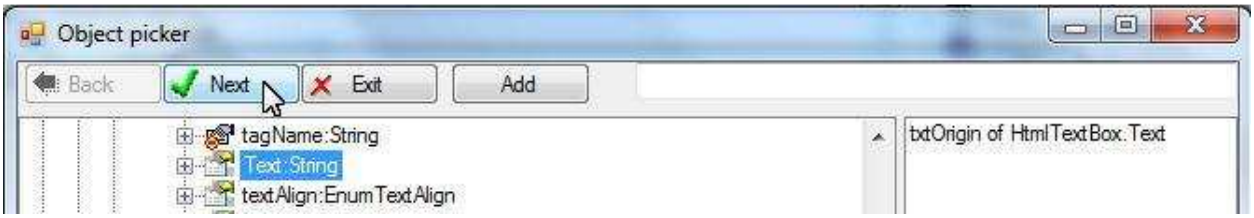




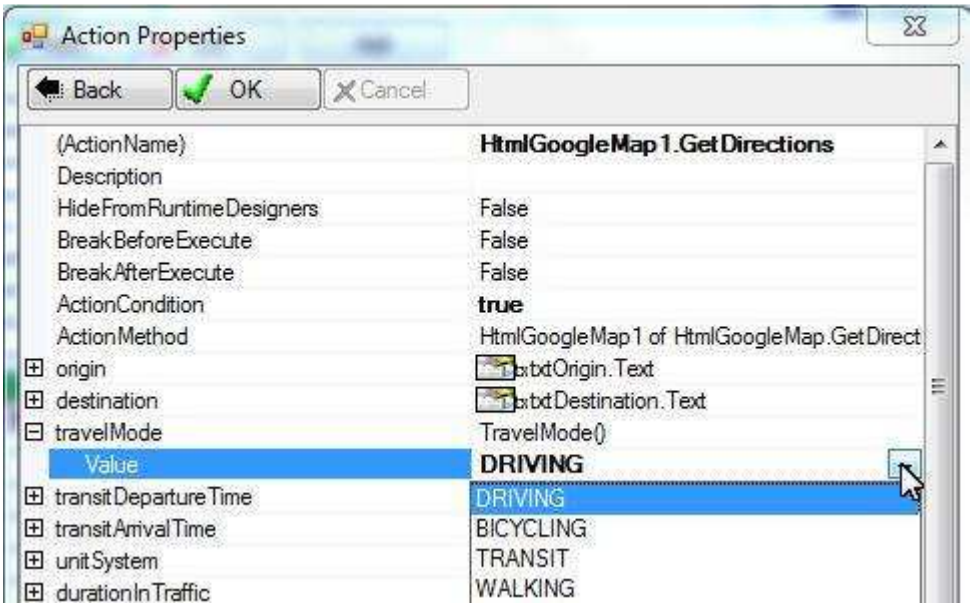




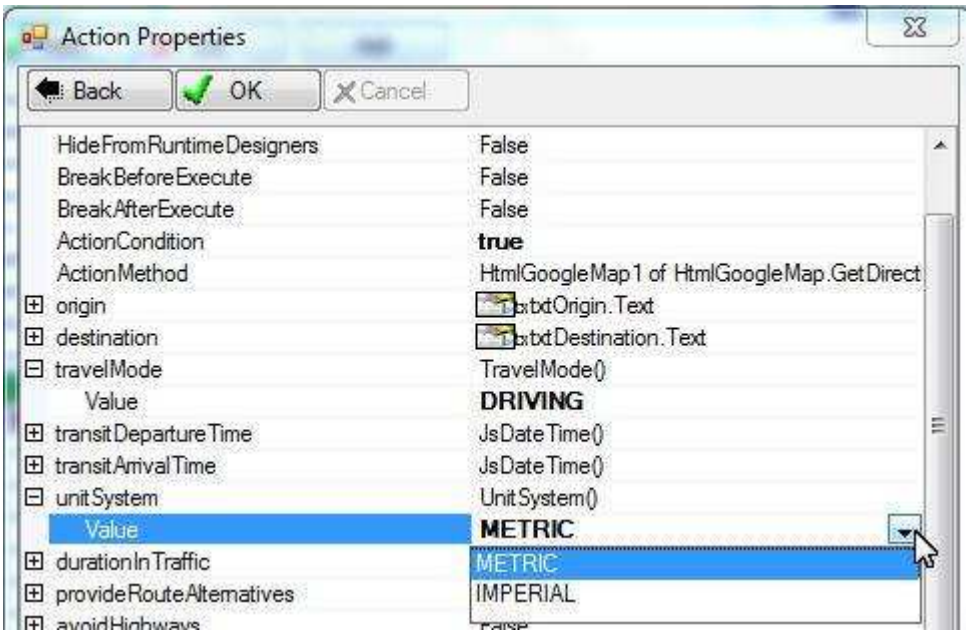




Choose travel mode:

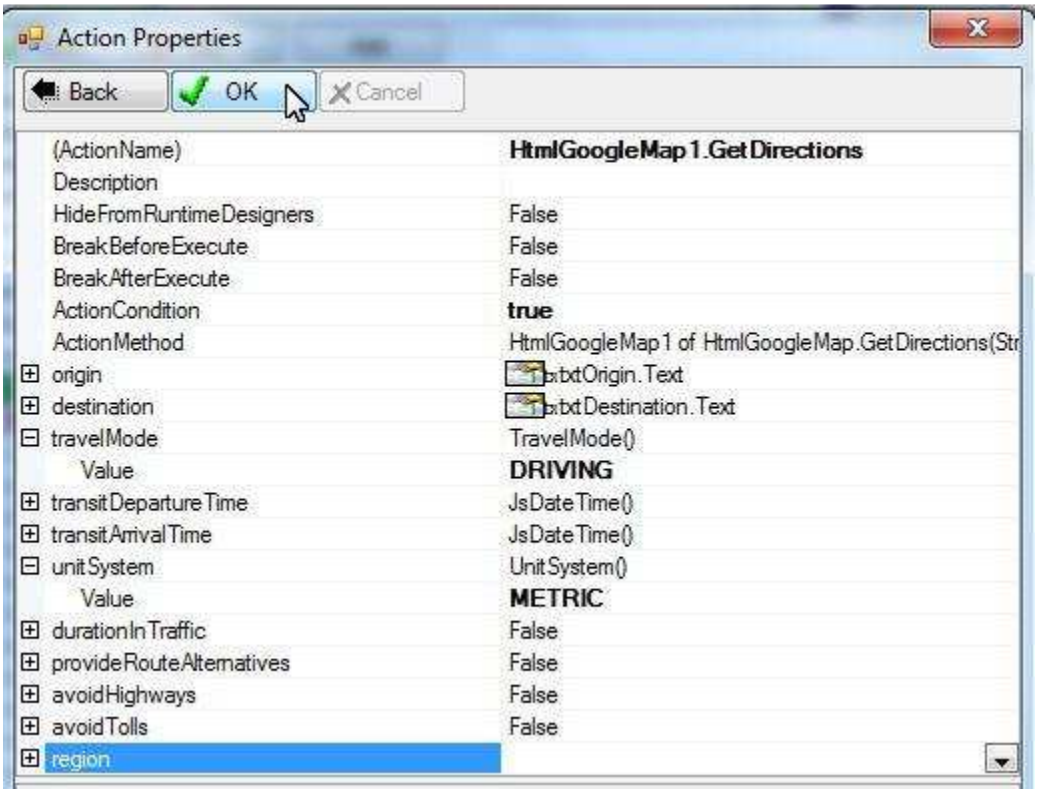


Choose unit system:

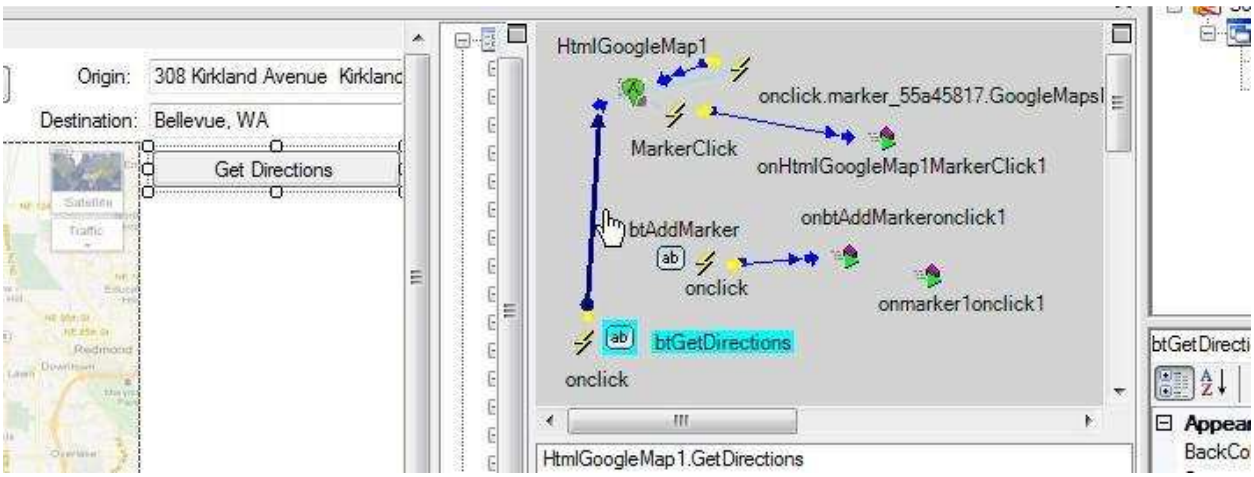


We may specify a region. For more information, see
https://developers.google.com/maps/documentation/javascript/directions#DirectionsRegionBiasing

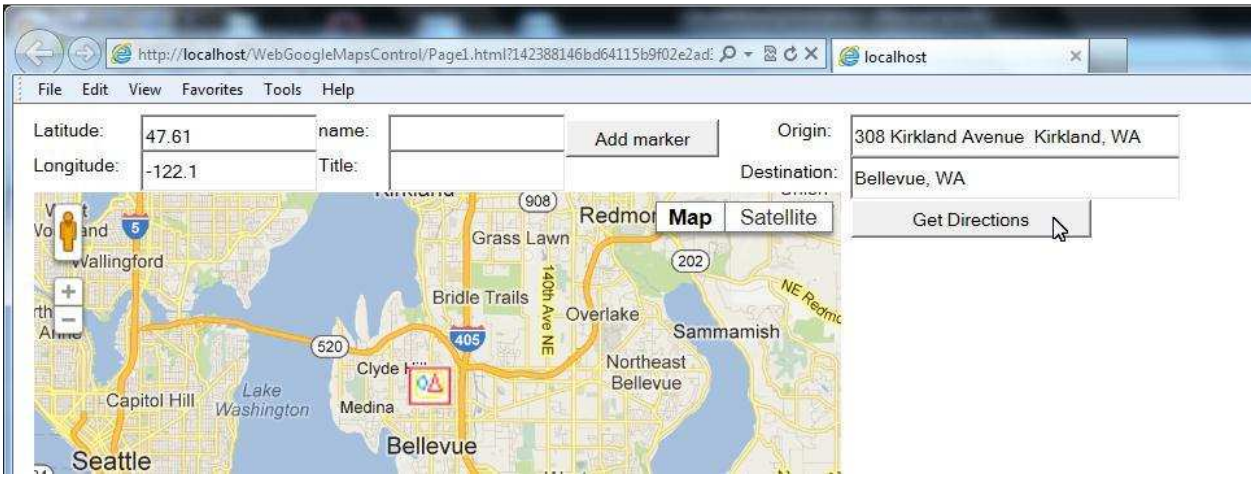
Click OK to finish creating the action:



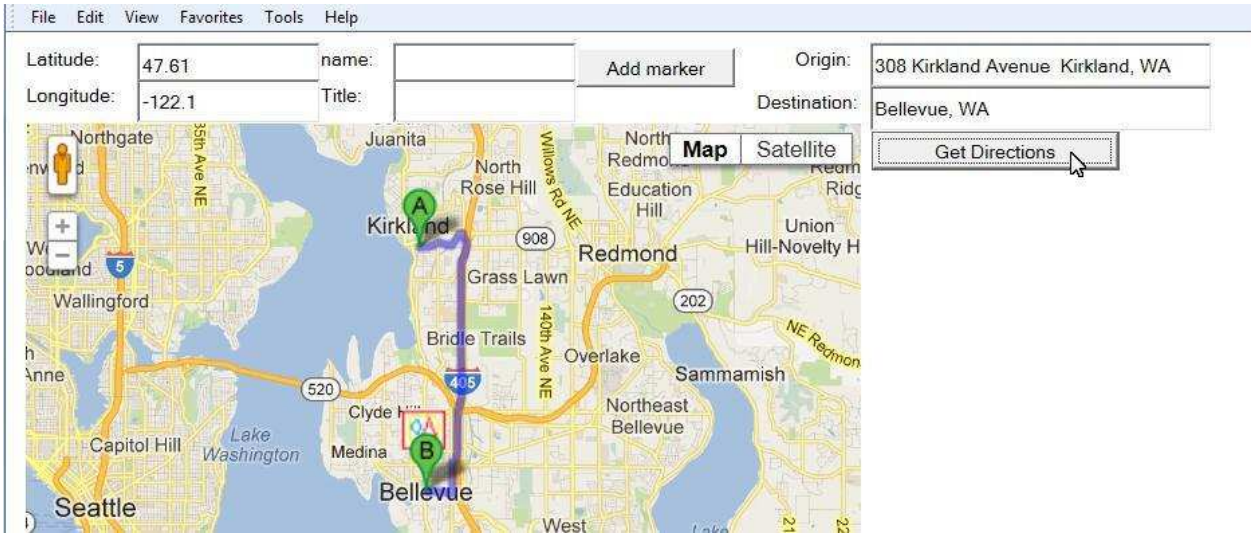
The action is created and assigned to the button:



We may launch the web page and try it:



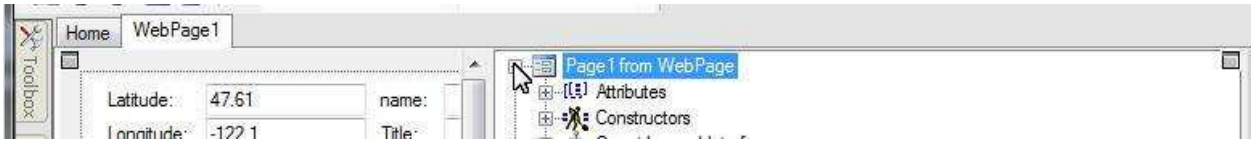
On clicking "Get Directions", we can see directions appear on the map:



Display Directions

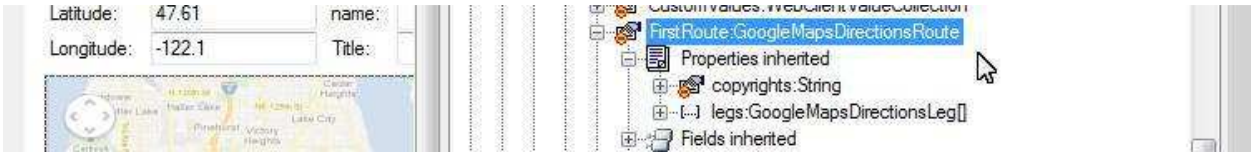
We may display property `GoogleMapsDirectionsResult` or `FirstRoute` on the web page after a successful directions query.

You may examine the properties contained in property `GoogleMapsDirectionsResult` or `FirstRoute` in Object Explorer to see what information they contain. Examine properties of HtmlGoogleMap:

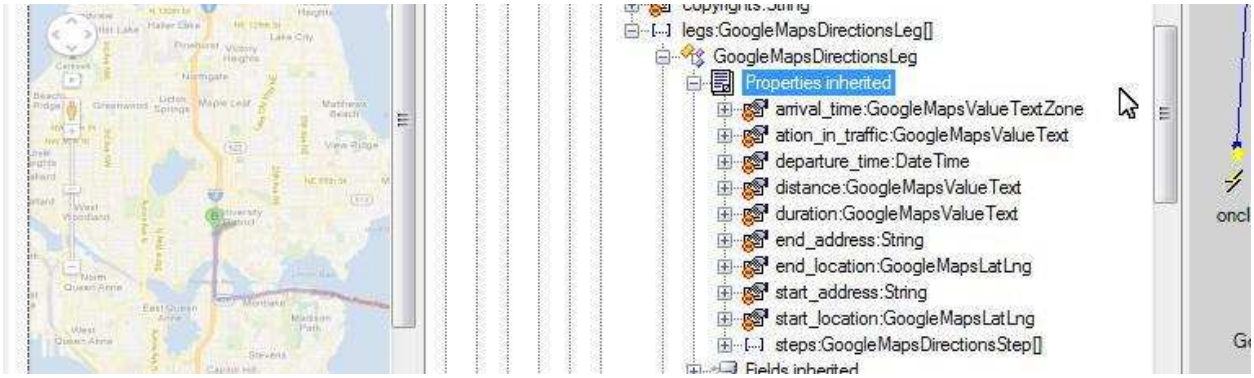


All information can be found under GoogleMapsDirectionsResult:

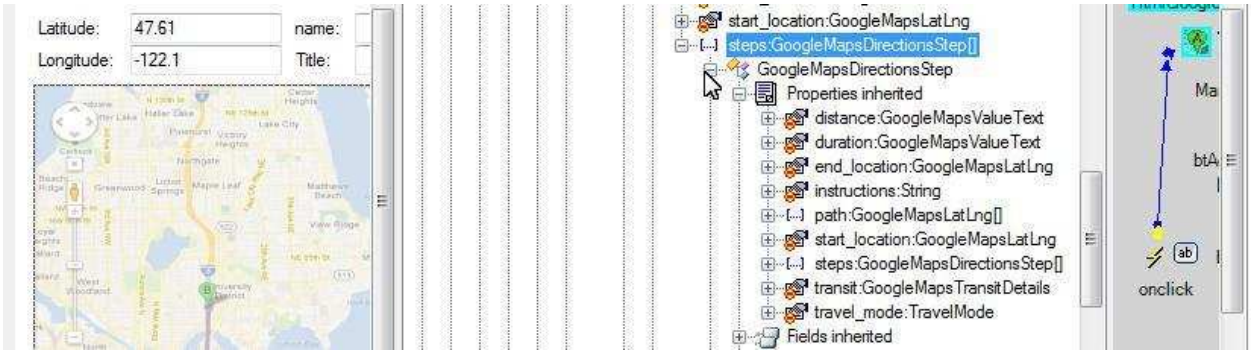
For convenience, the first route contained in routes can be found at FirstRoute property:



We can see that each route has an array named legs. Let's expand legs node. We can see a GoogleMapsDirectionsLeg node. Expand GoogleMapsDirectionsLeg's properties node:



You may expand each property to examine what information it contains. A leg contains an array of steps. Expand it to see what it contains:

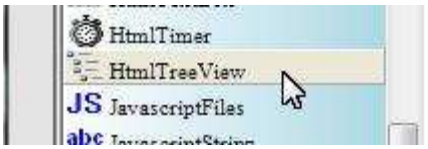


Note that a step also contains an array of steps.

Google web site provides detailed information on these elements. See https://developers.google.com/maps/documentation/directions/

You may display directions result based on your business needs.

Here we use a tree view as an example. Add an HtmlTreeView control to the web page:



For information on using HtmlTreeView, see http://www.limnor.com/support/WebTreeView.pdf

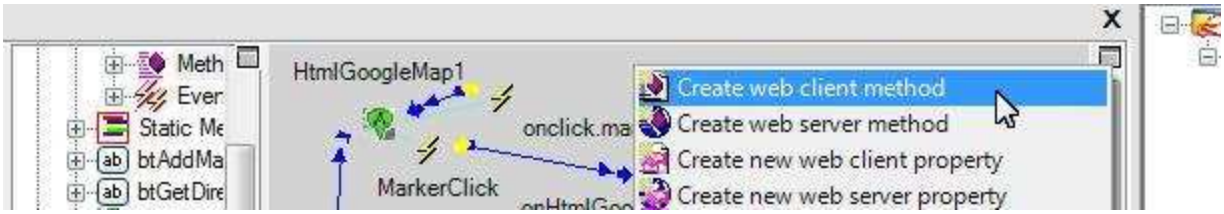
We create a root tree node to represent a route. For each leg in a route, create a child node for the route node. For each step in a leg create a child node for the leg node. For each step in a step create a child node for the step node. The above programming is done in the following way.

- We create a method, **showRoute**, which takes a route as its parameter.
- We create a method, **showLeg**, which takes two parameters: a parent tree node representing a route node; a leg representing the leg to be displayed.
- We create a method, **showStep**, which takes two parameters: a parent tree node representing a leg node or a step node; a step representing the step to be displayed.

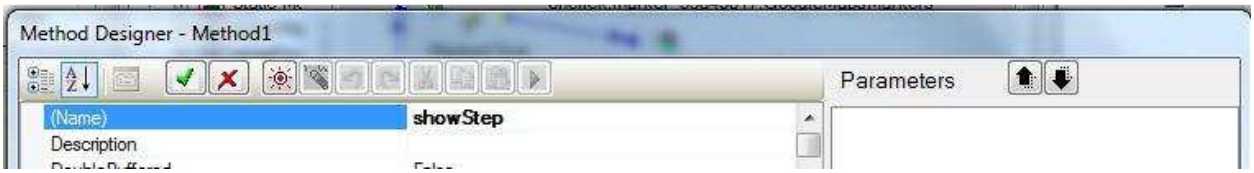
Show Step

We create **showStep** first.

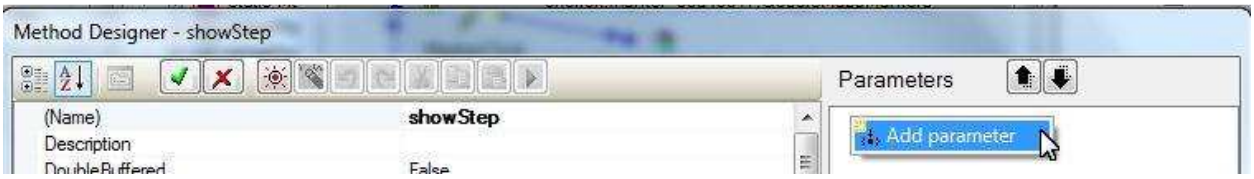
Add Parameters



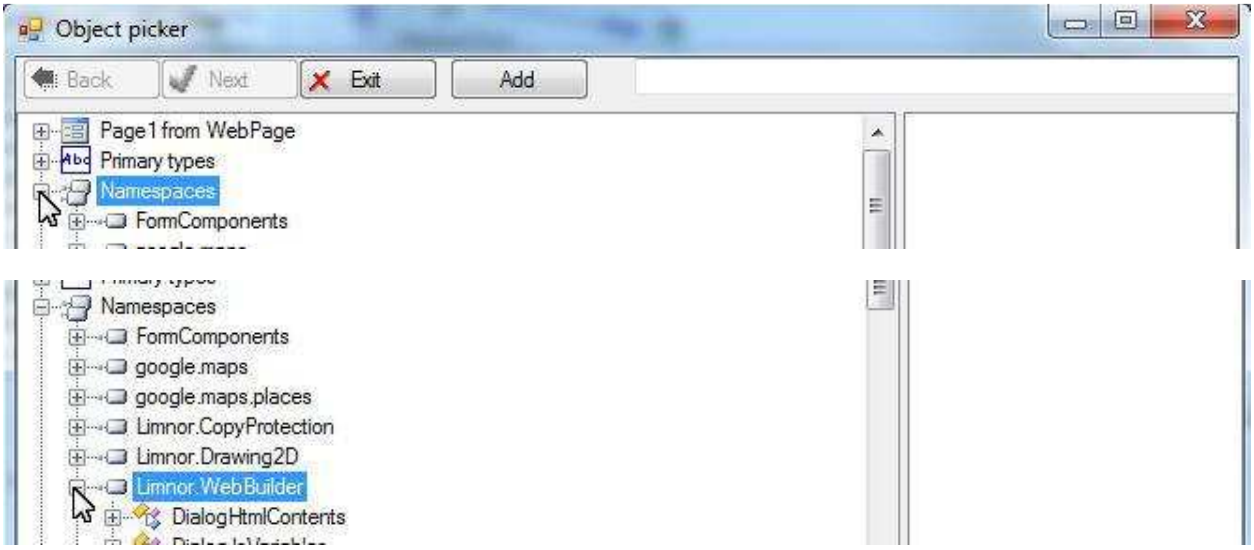
Rename the method:

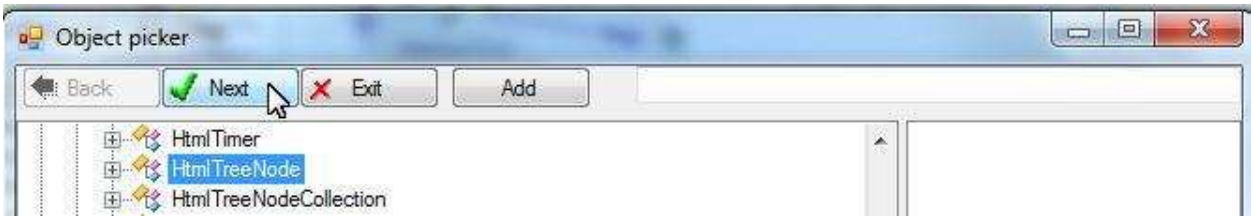


Add parent tree node parameter:

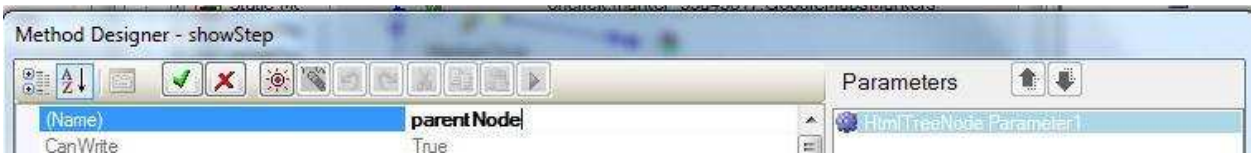


Select tree node:

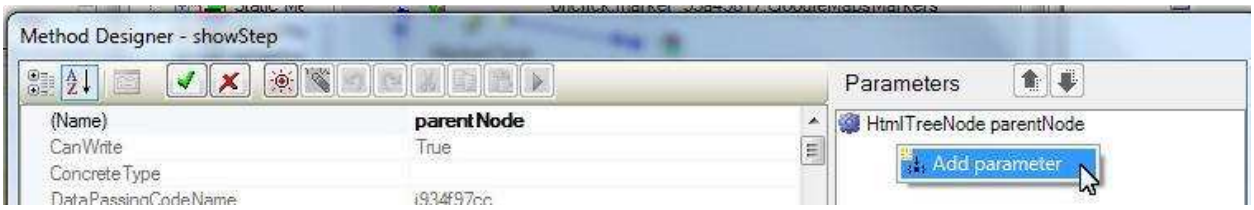




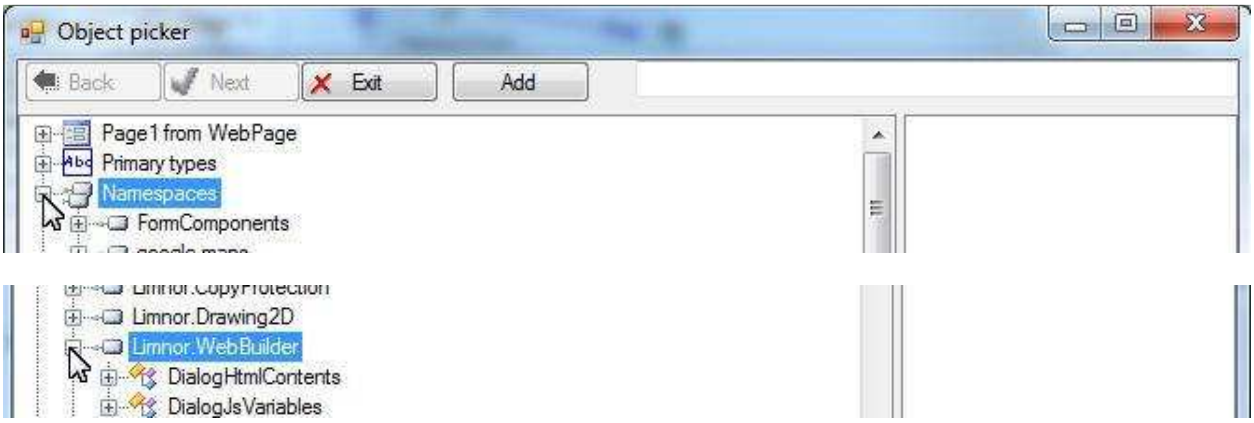
Rename the parameter:

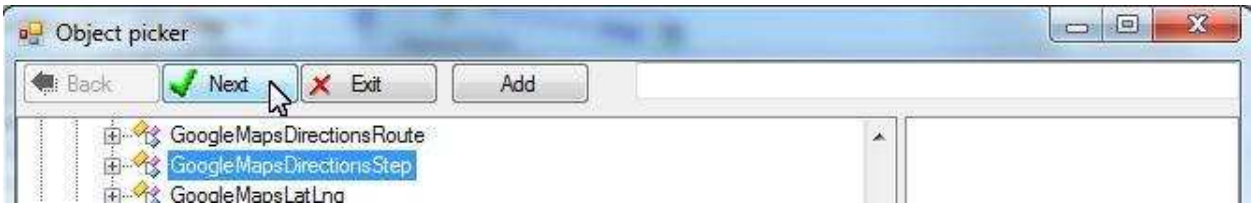


Add a parameter for step:

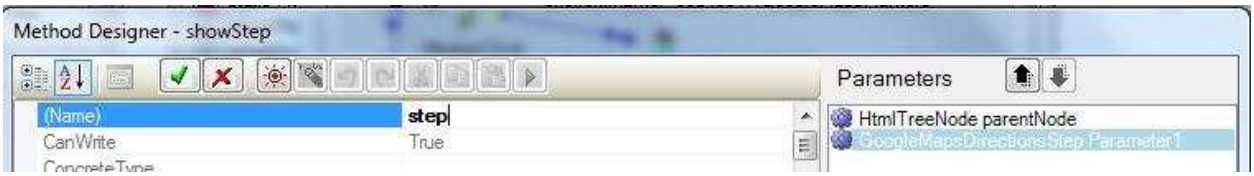


Select step object:

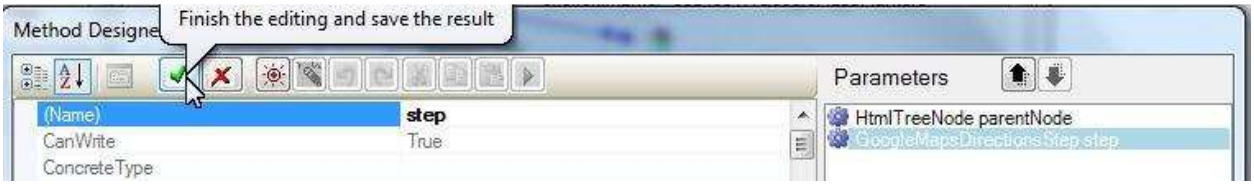


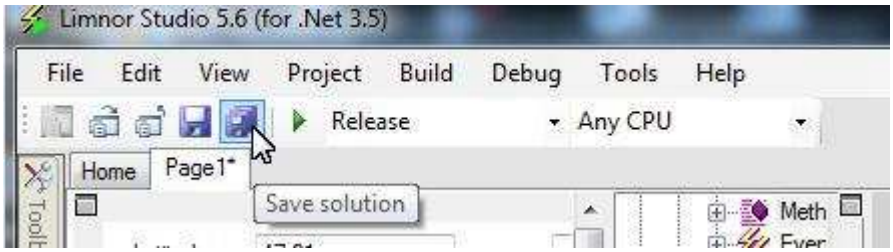


Rename the parameter:



We need to close the method editor and save the project at this time so that this method will appear in the web page. This method will be a recursive method because it will call itself.



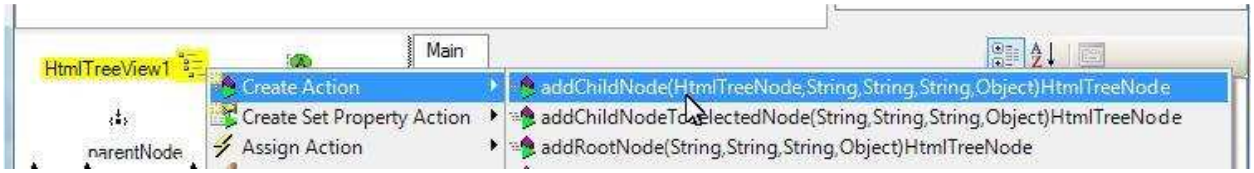


The method is created:

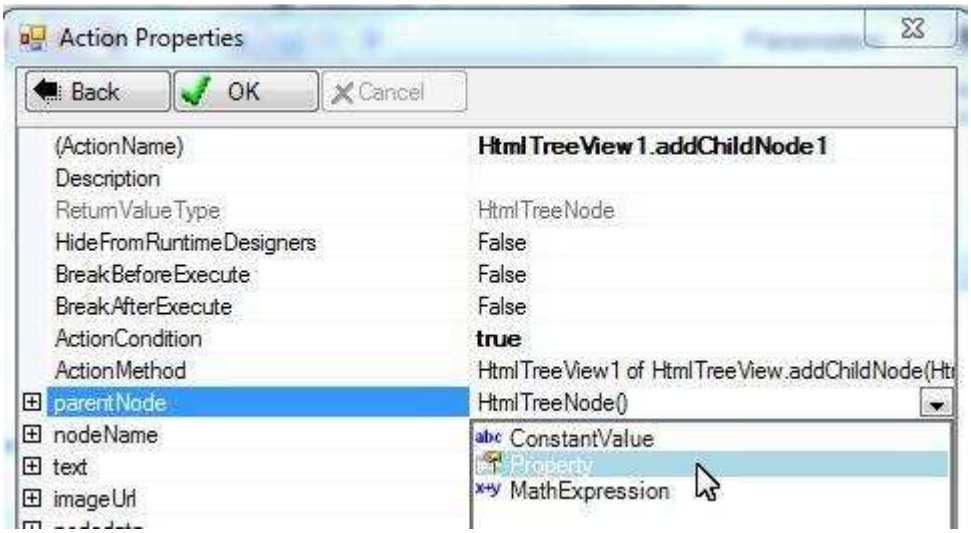
Display step

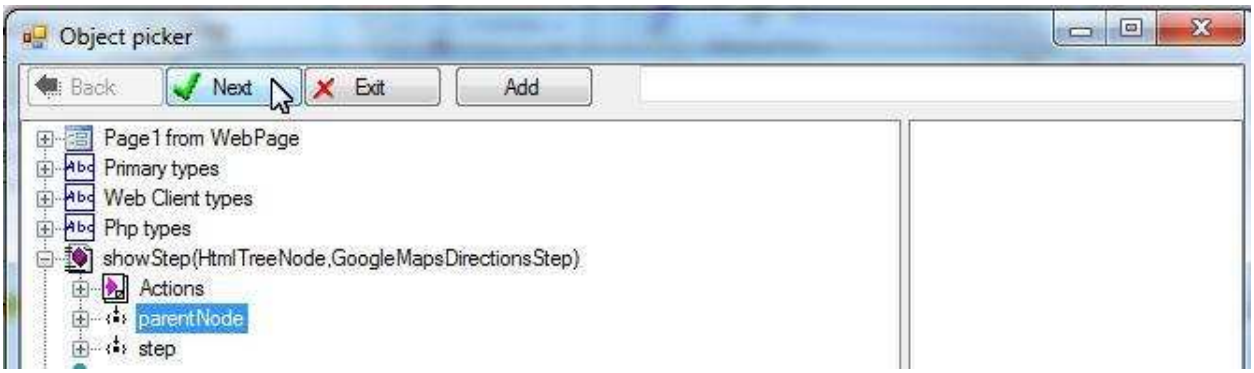
We may edit showStep to display the step:

Create an “addChildNode” action to display the step:

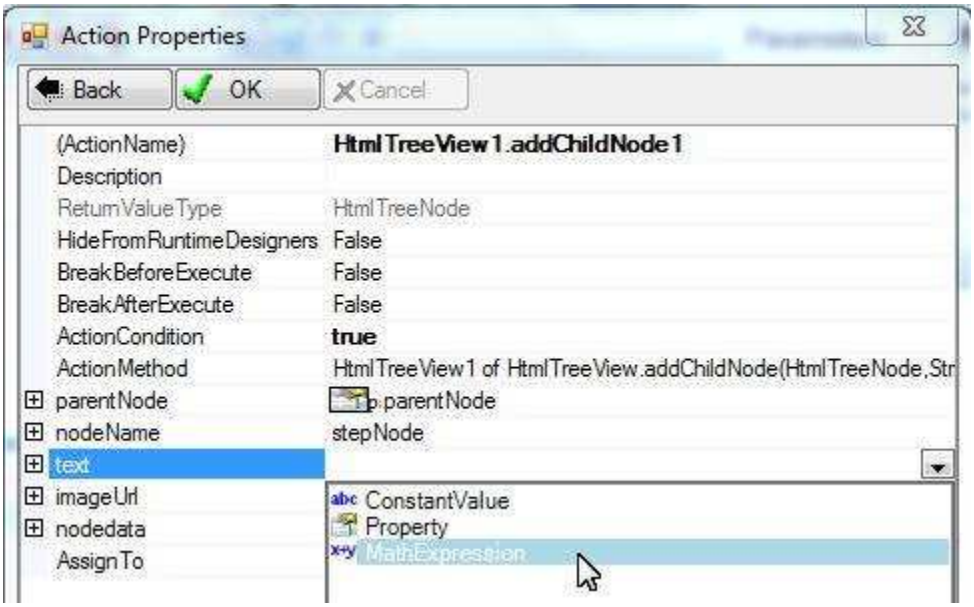


Use the method parameter for the parent node:

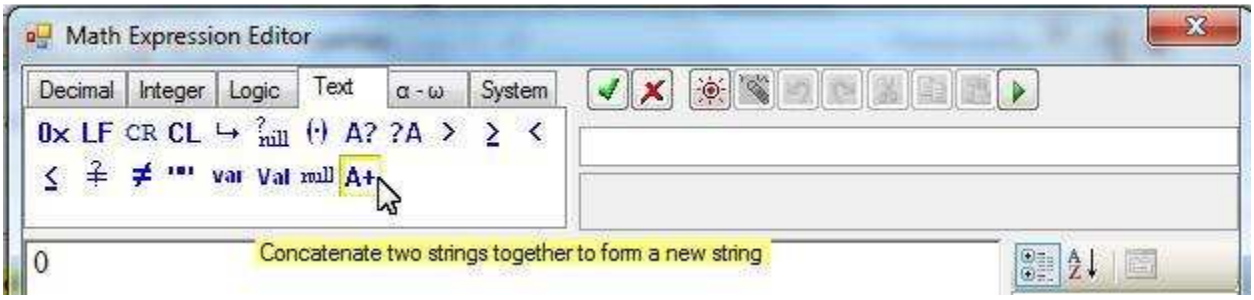


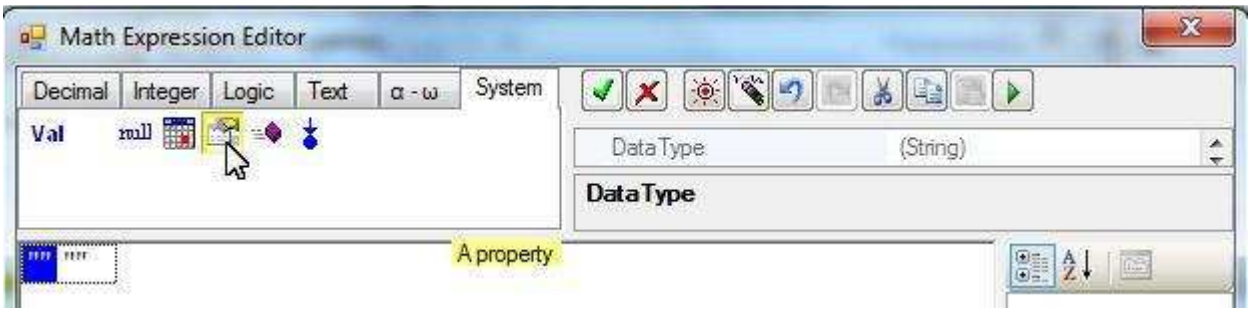


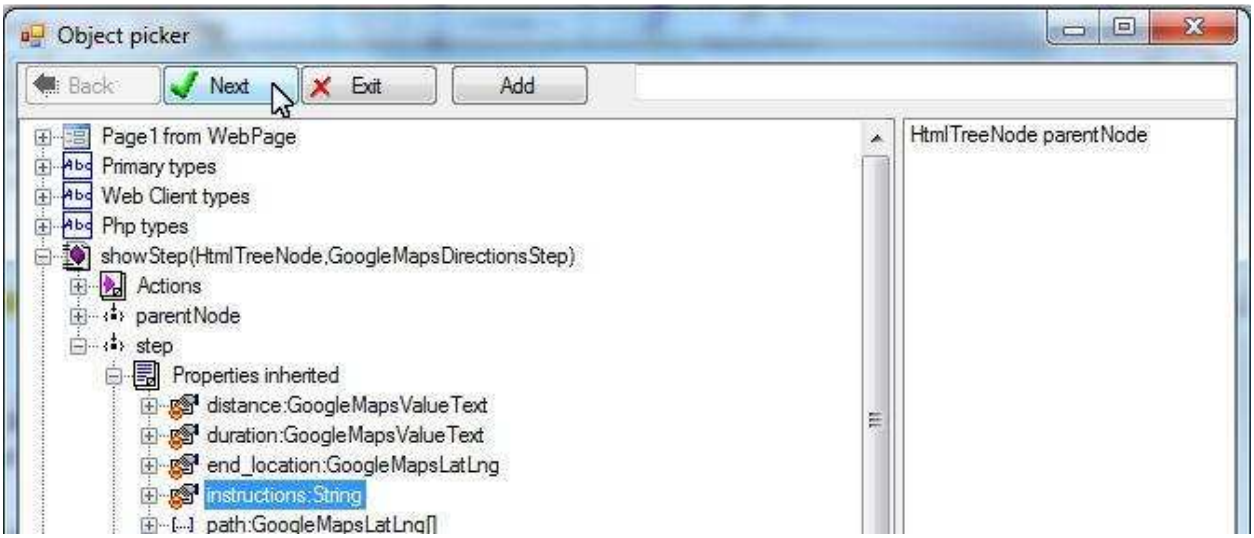
Use an expression to form step display:



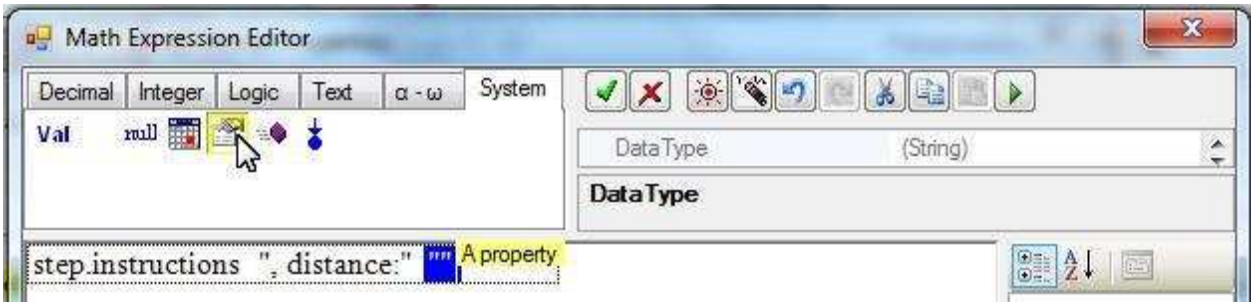
Use "A+" to concatenate step properties we want to display:


Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Concatenate two strings together to form a new string

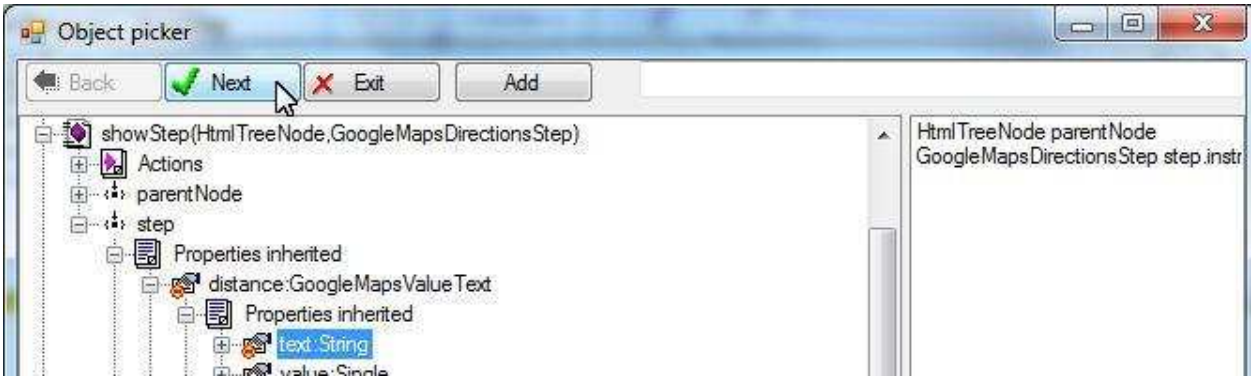

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
DataType
(String)
DataType
A property

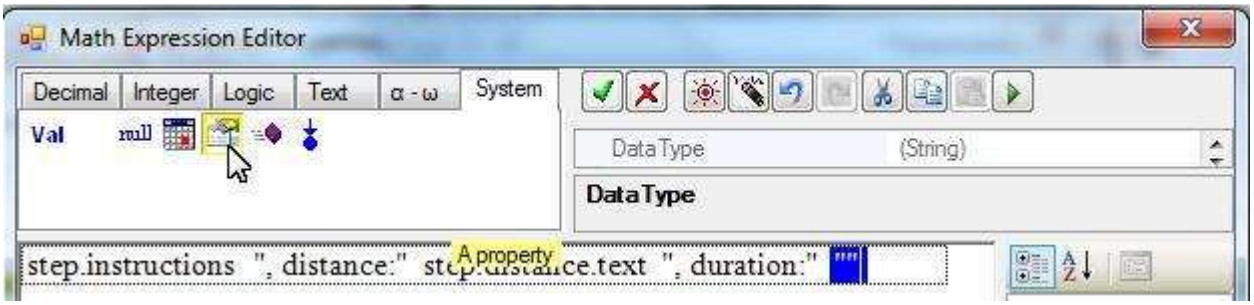

Object picker
Back
Next
Exit
Add
Page1 from WebPage
Primary types
Web Client types
Php types
showStep(HtmlTreeNode,GoogleMapsDirectionsStep)
Actions
parentNode
step
Properties inherited
distance:GoogleMapsValueText
duration:GoogleMapsValueText
end_location:GoogleMapsLatLng
instructions:String
path:GoogleMapsLatLng[]
HtmlTreeNode parentNode

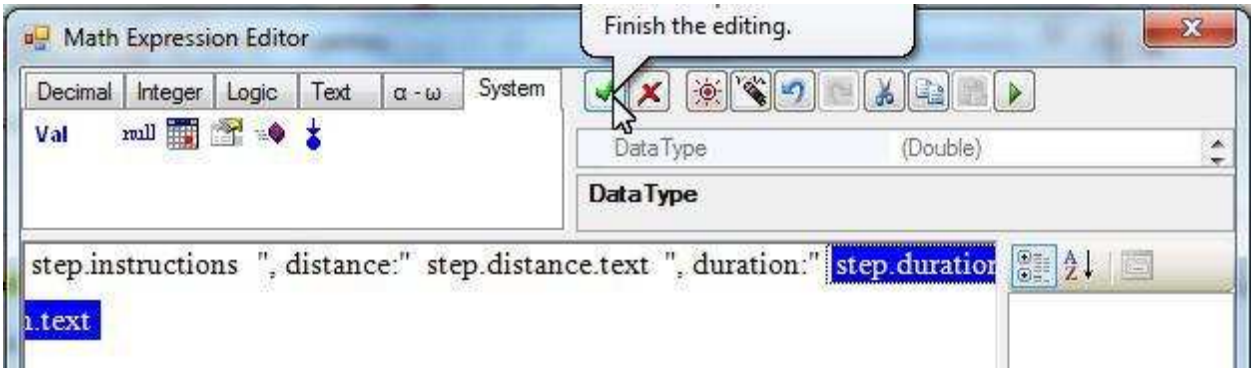

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
DataType
(String)
DataType
step.instructions
Concatenate two strings together to form a new string



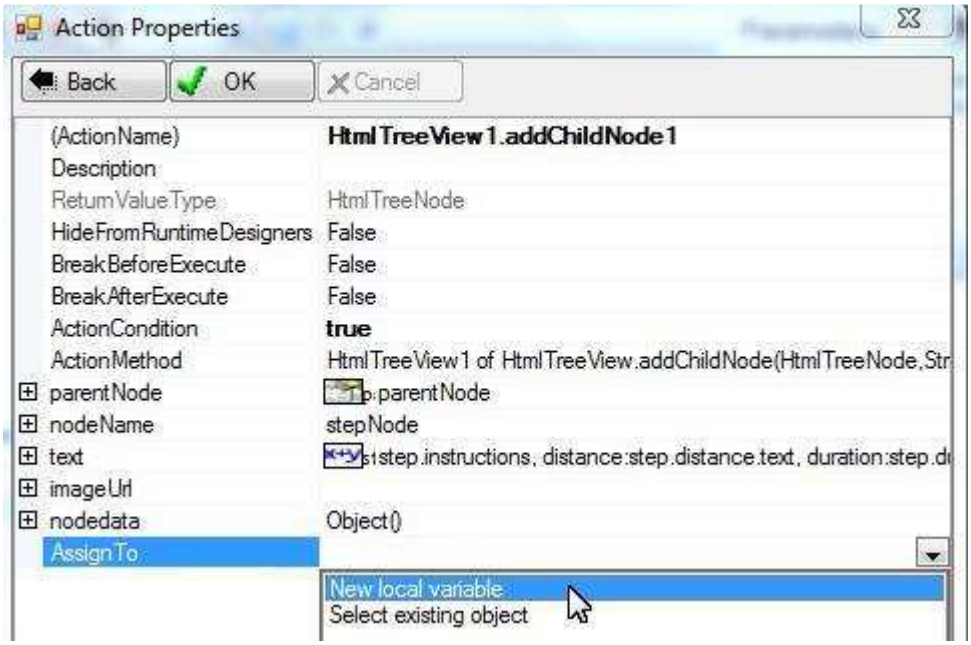
Note that we use “text” property of “distance”:

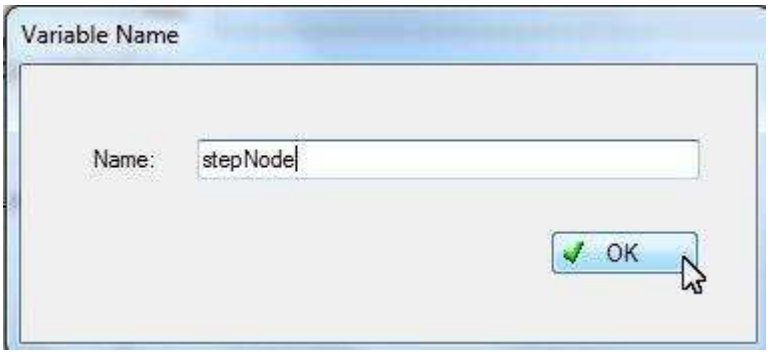




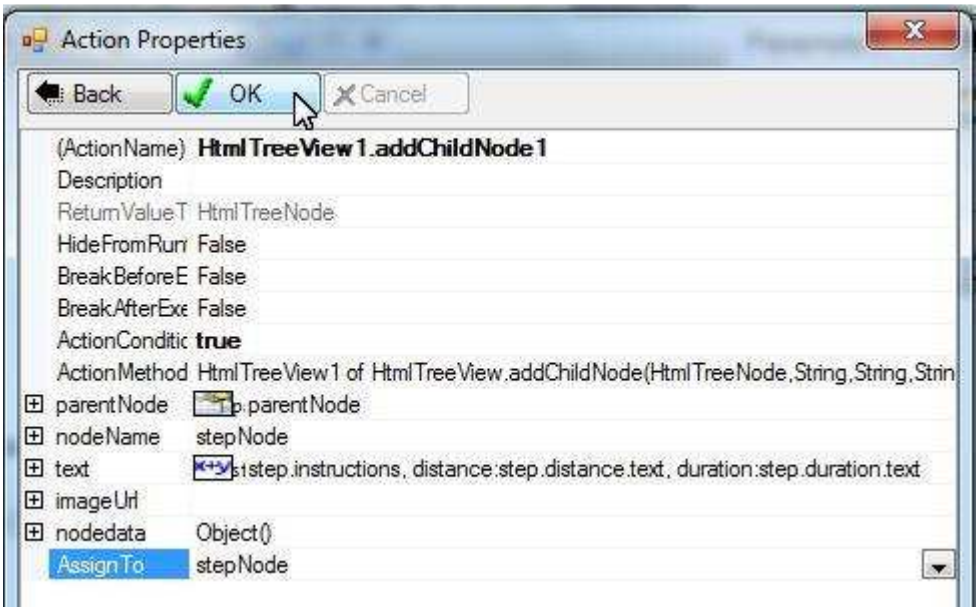


Use a new variable for the new tree node:

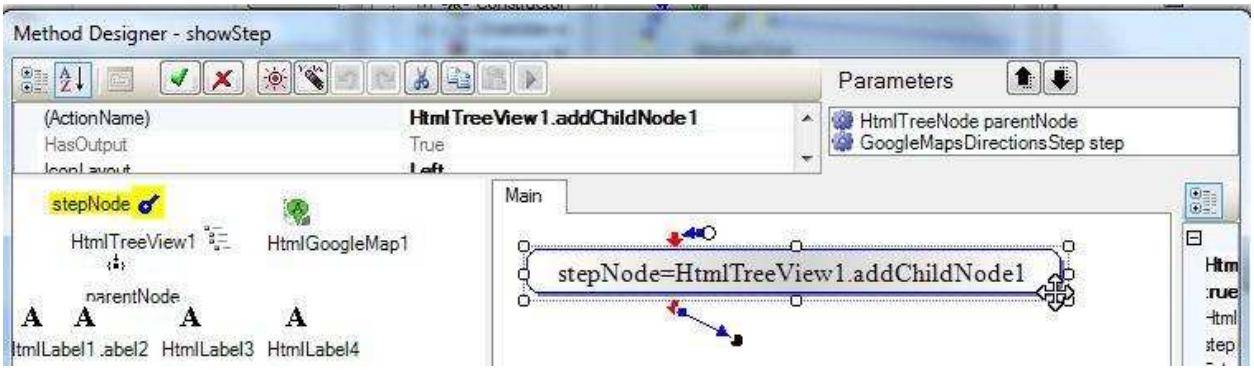




Click OK to finish creating the action:

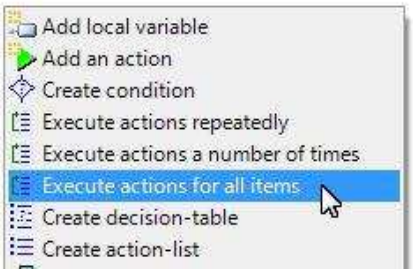


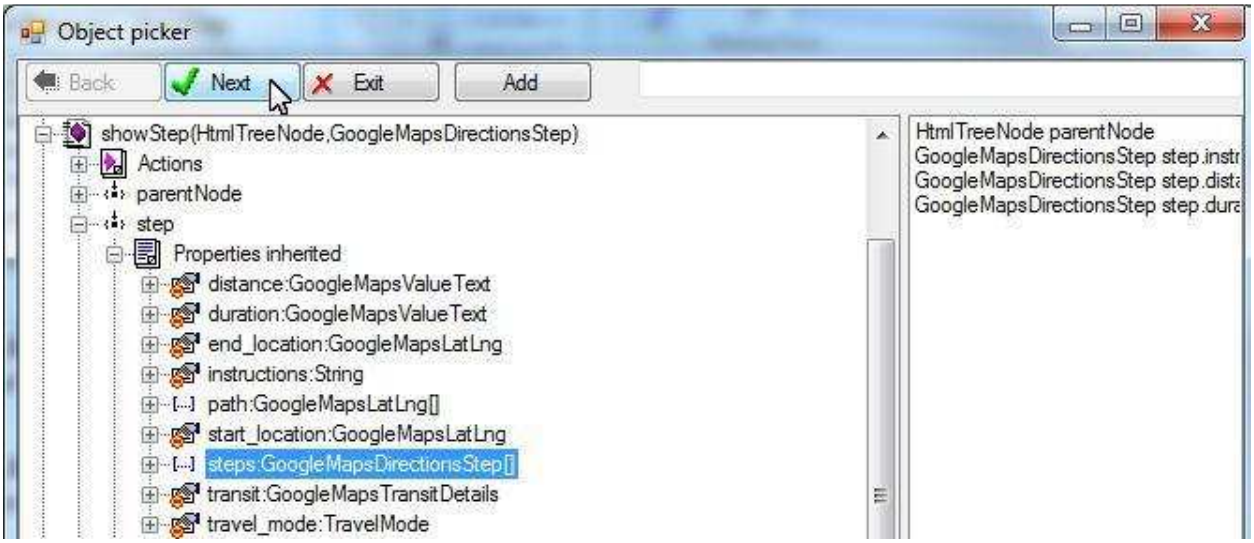
The new step node and the new action appear:



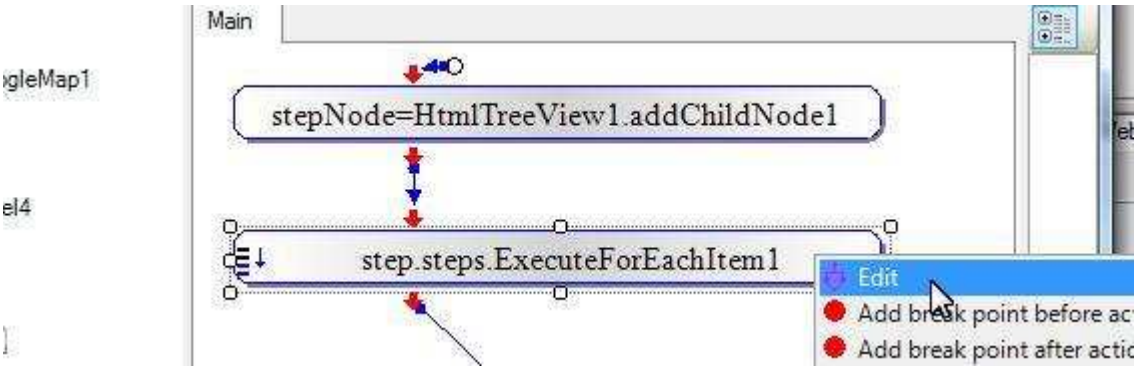
Display sub-steps

The step has its own steps array. We may use an "execute actions for all items" action to display sub-steps:

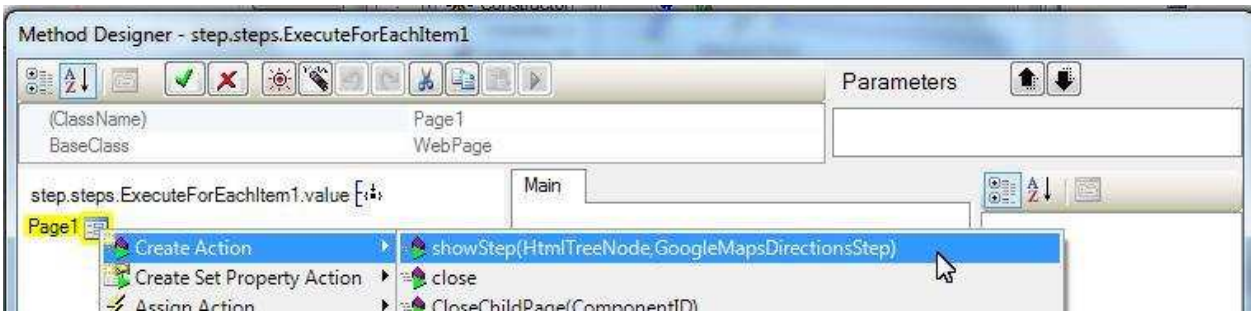




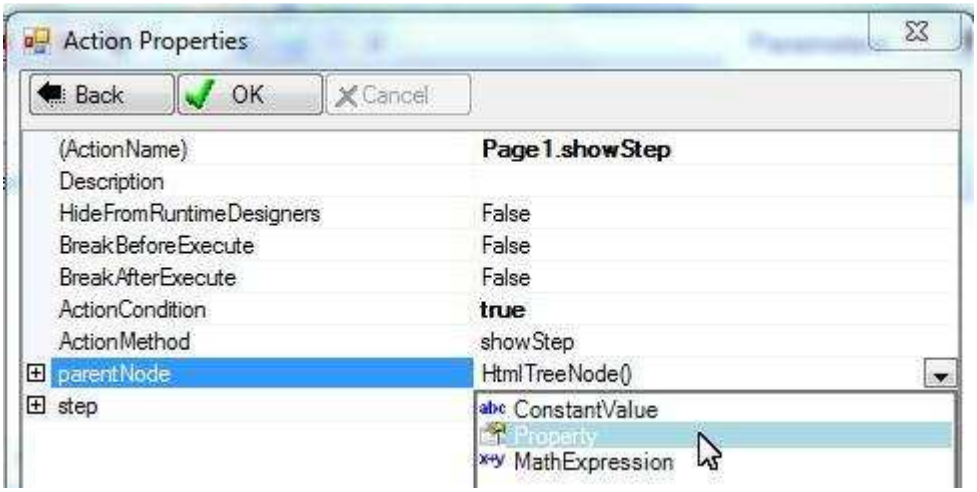
Edit the action to perform actions on each step:



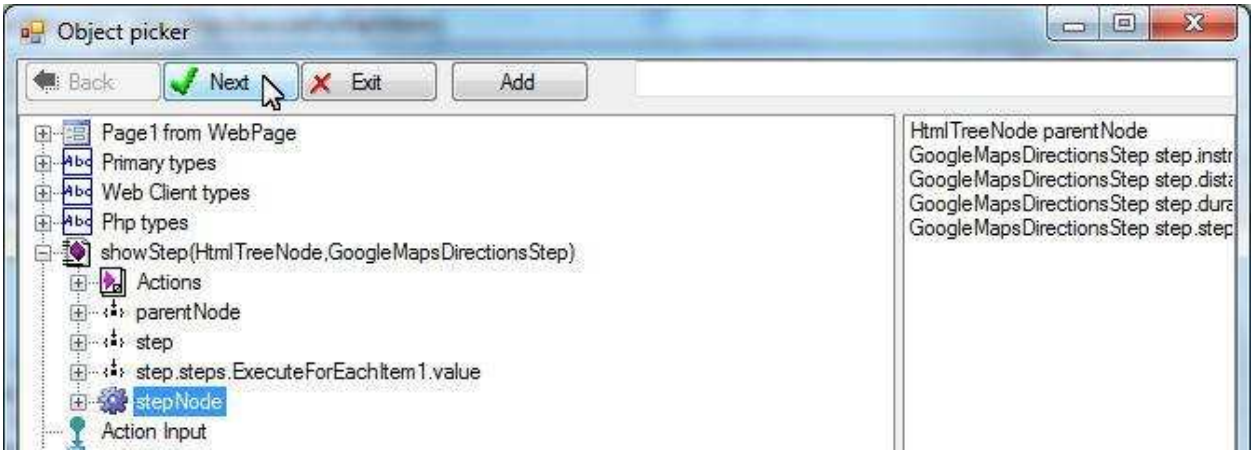
A method editor appears for adding actions. Note that “value” icon is the step to be displayed. Create a “showStep” action to display the step:



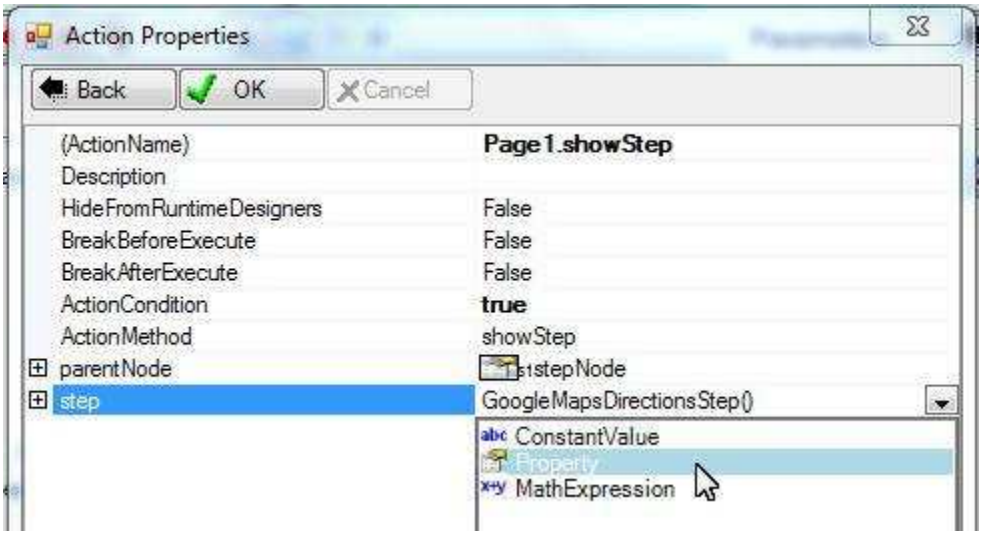
Select parent node:

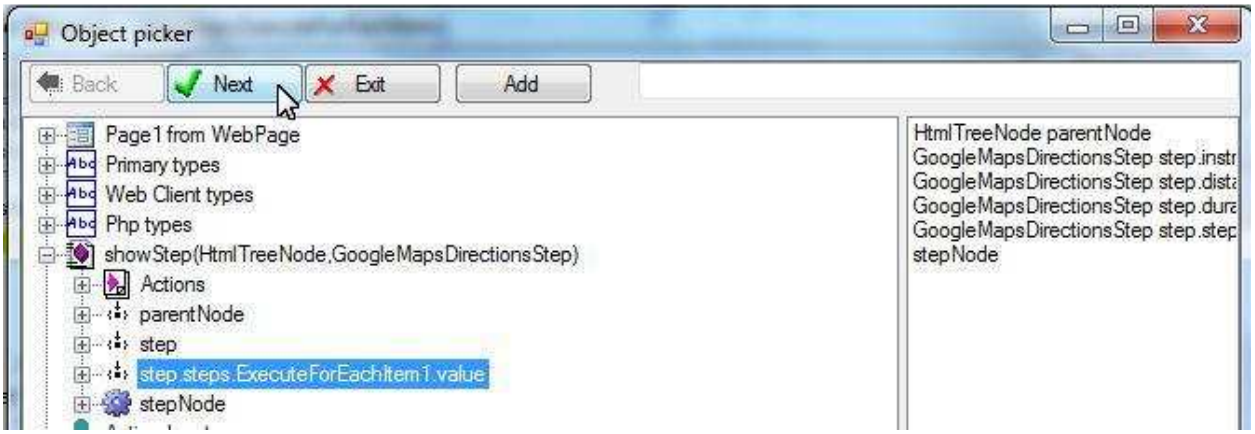


Note that we must use the stepNode variable as the parent node:

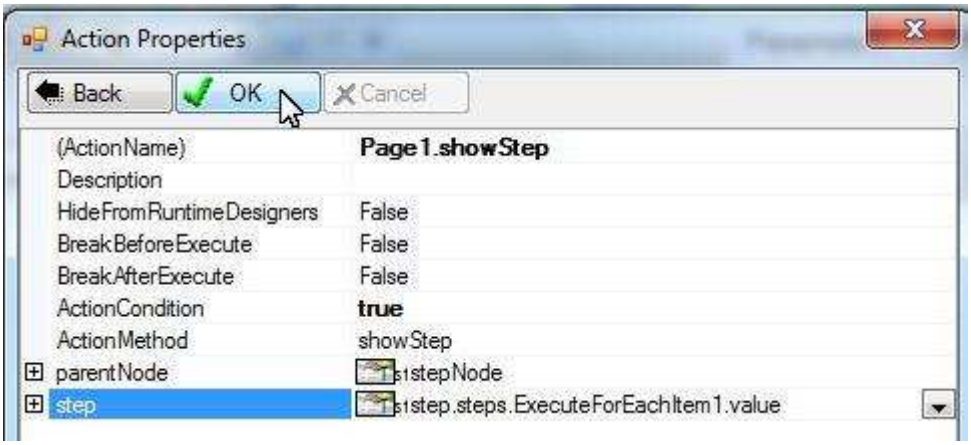


Select “value” for step:

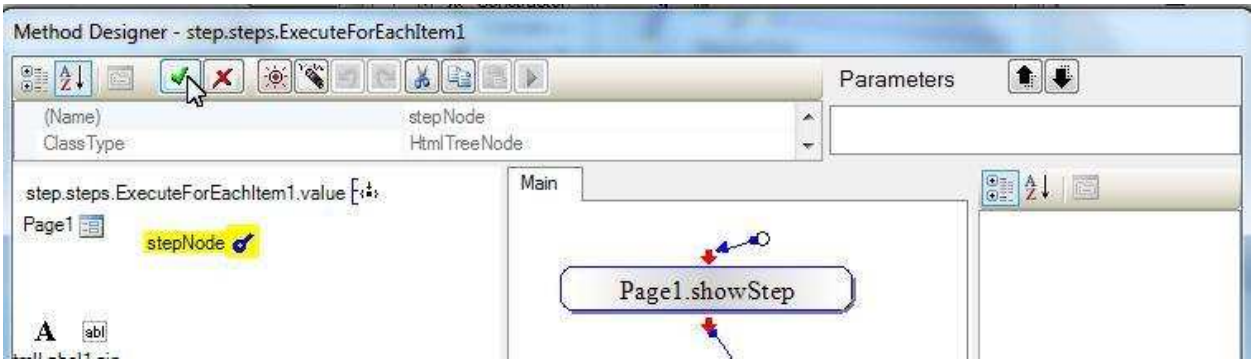




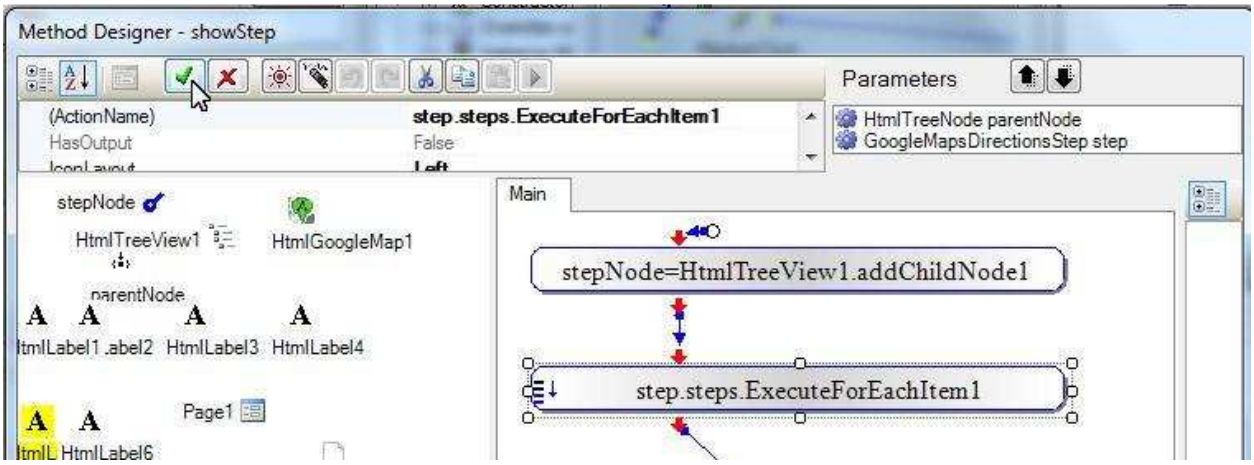
Click OK:



That is all we need:



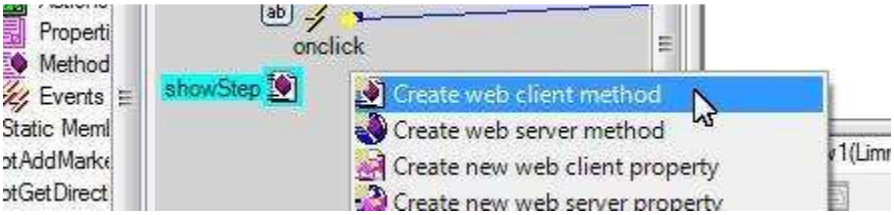
We are done creating showStep:



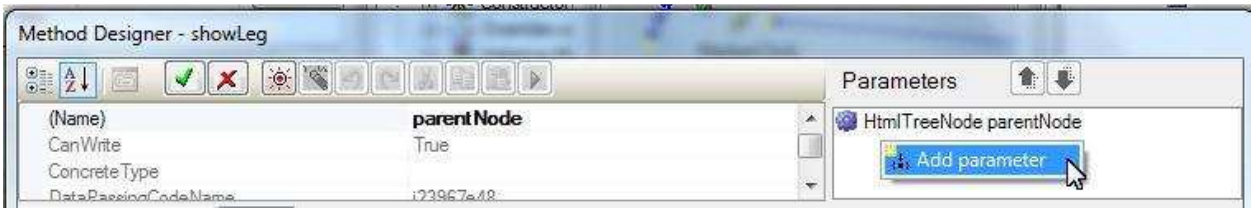
Show Leg

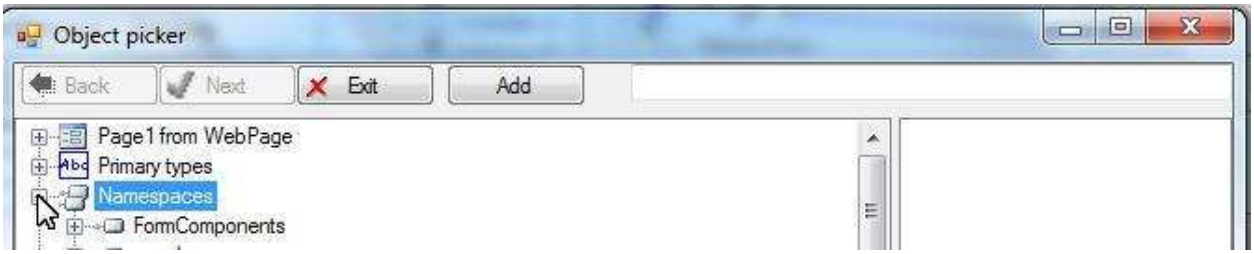
We may create a showLeg method.

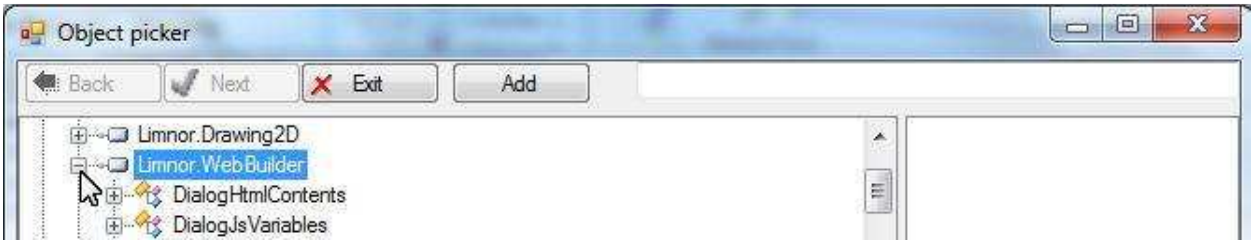
Add parameters

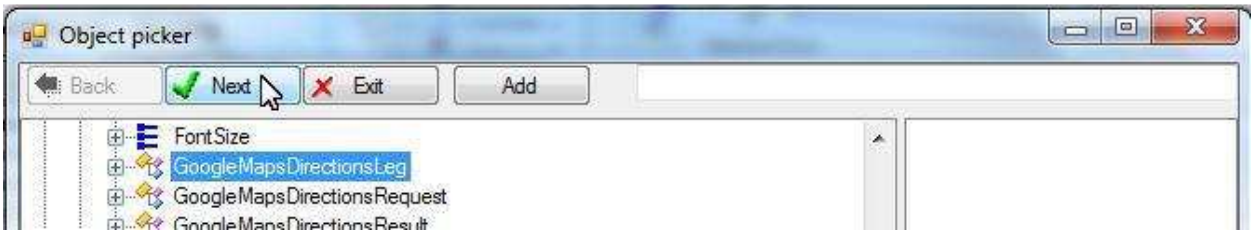


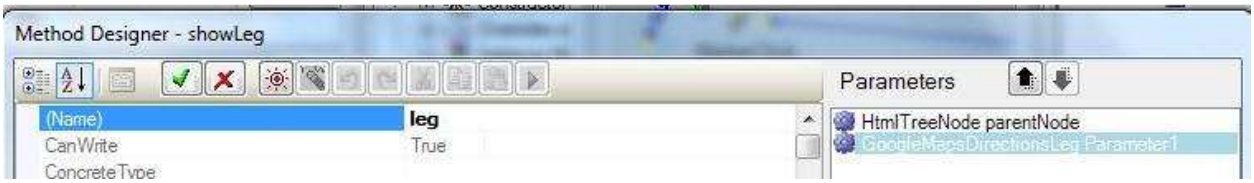
Rename the method and add a parent tree node parameter. Add a parameter for leg:





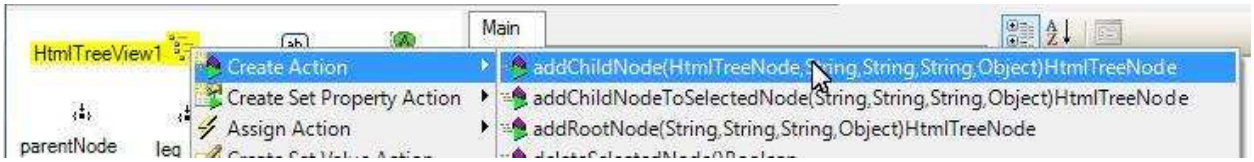




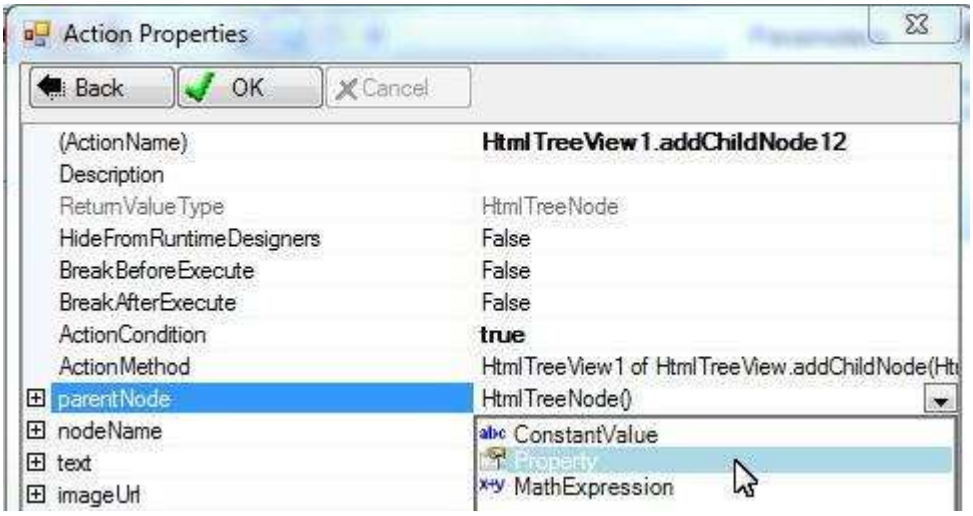


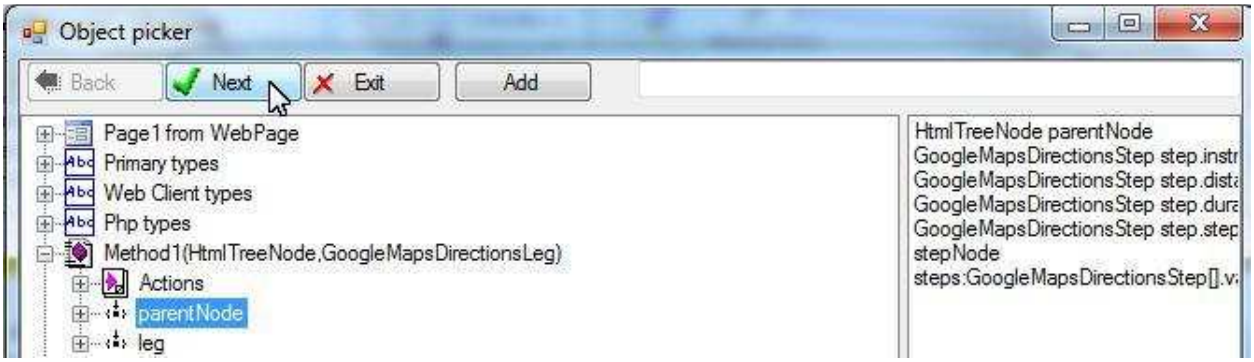
Display leg

Add a child node to display the leg.

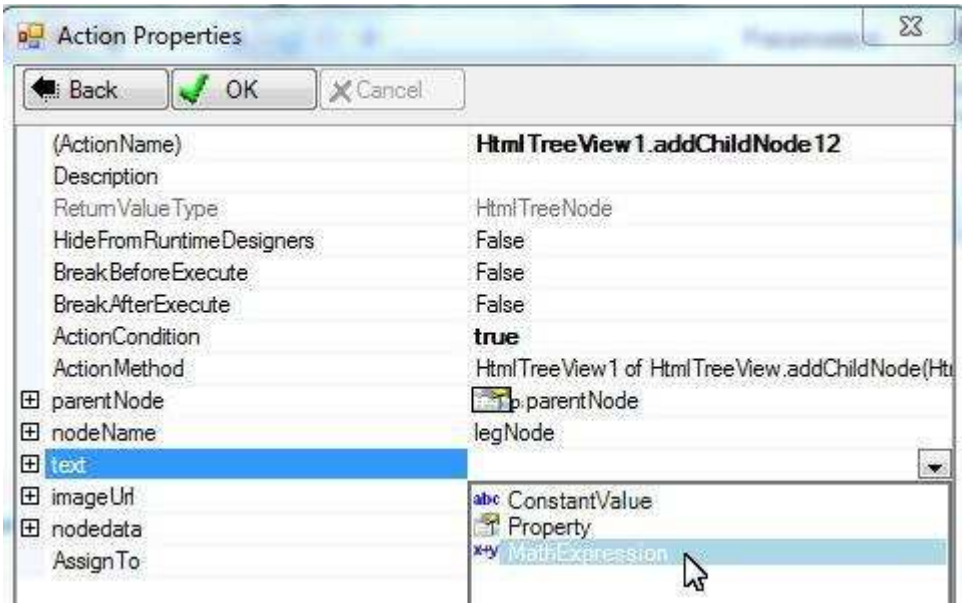


Set parent node:

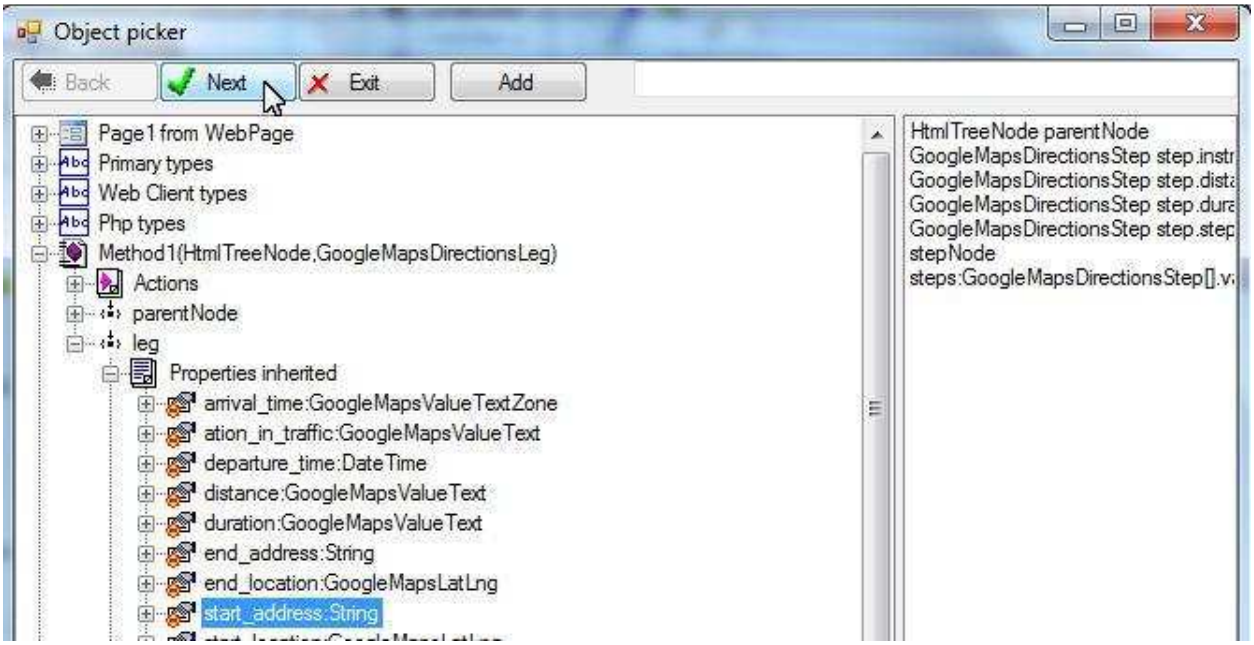


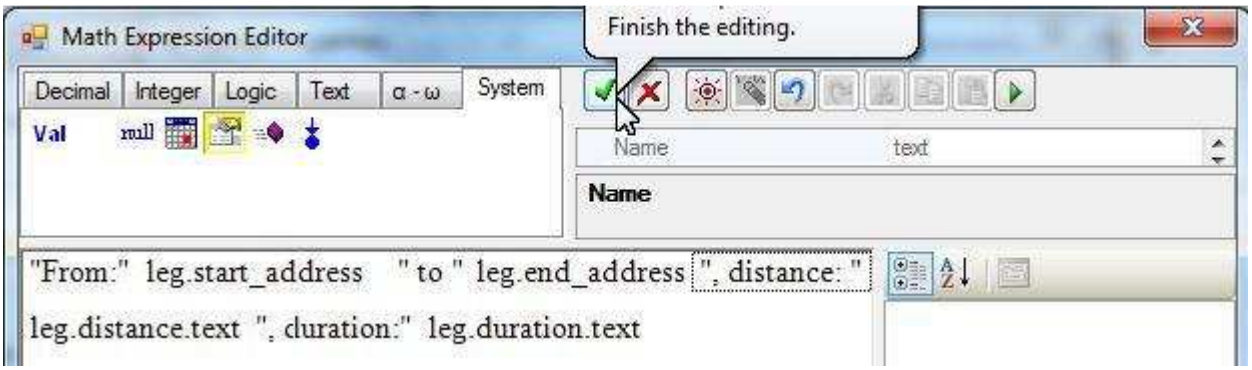


Set node text to display the leg:

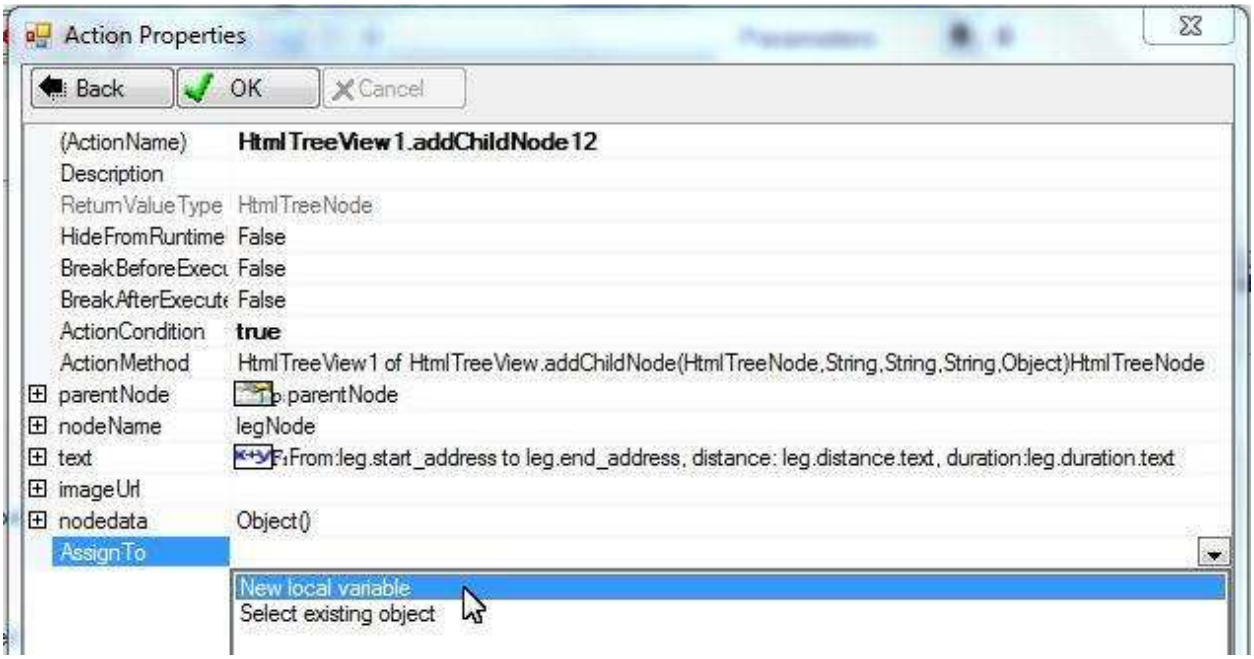


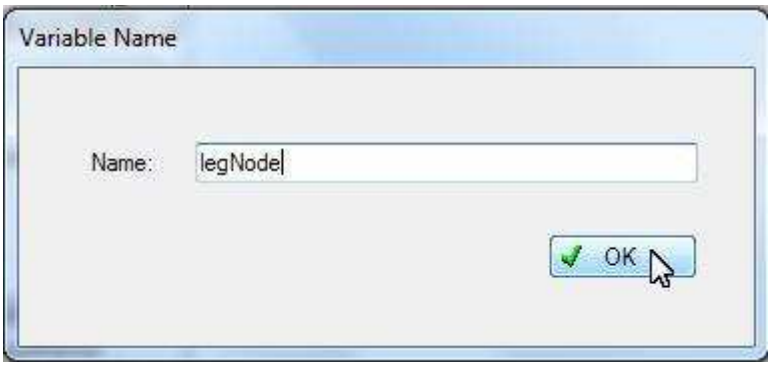
Form display expression using “A+” with properties of the leg.



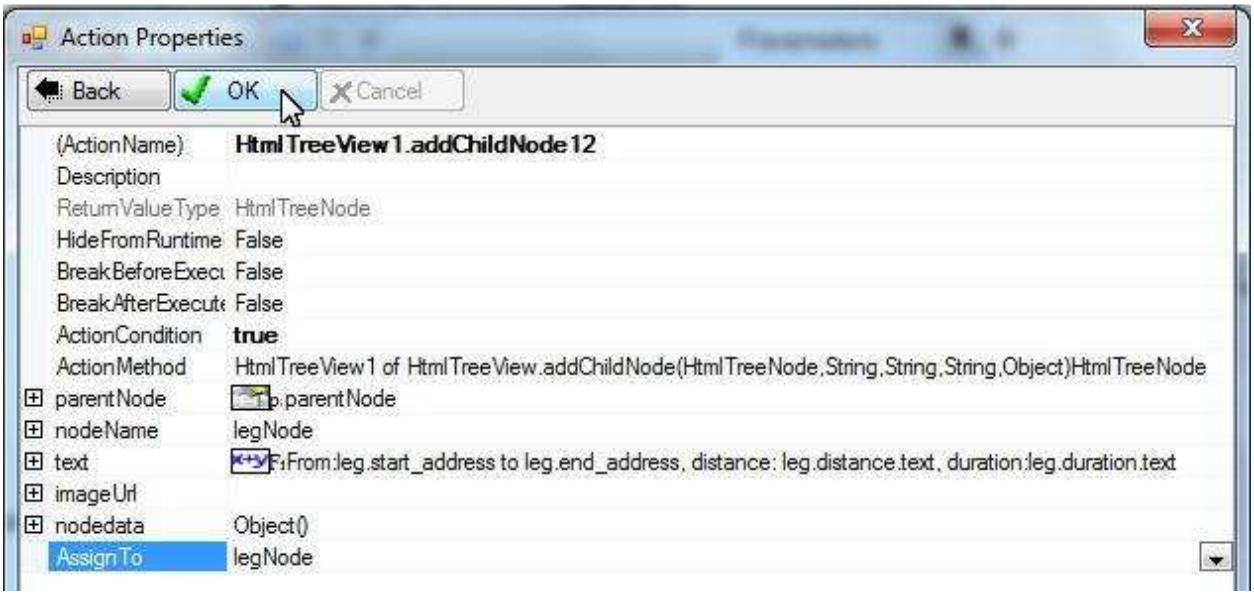


Use a variable for the new node:

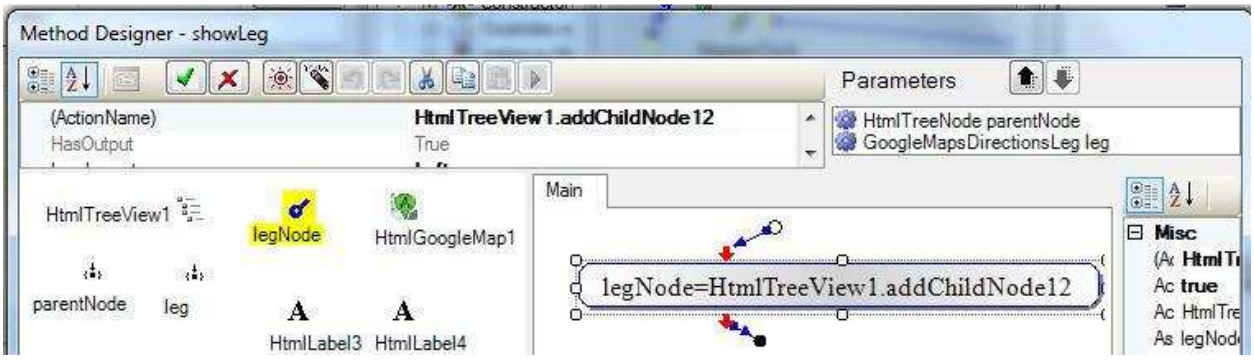

Action Properties
Back
OK
Cancel
(ActionName) HtmlTreeView1.addChildNode12
Description
ReturnValueType HtmlTreeNode
HideFromRuntime False
BreakBeforeExecu False
BreakAfterExecute False
ActionCondition true
ActionMethod HtmlTreeView1 of HtmlTreeView.addChildNode(HtmlTreeNode,String,String,String,Object)HtmlTreeNode
parentNode parentNode
nodeName legNode
text From:leg.start_address to leg.end_address, distance: leg.distance.text, duration:leg.duration.text
imageUrl
nodedata Object()
AssignTo
New local variable
Select existing object


Variable Name
Name:
legNode
OK

Click OK:

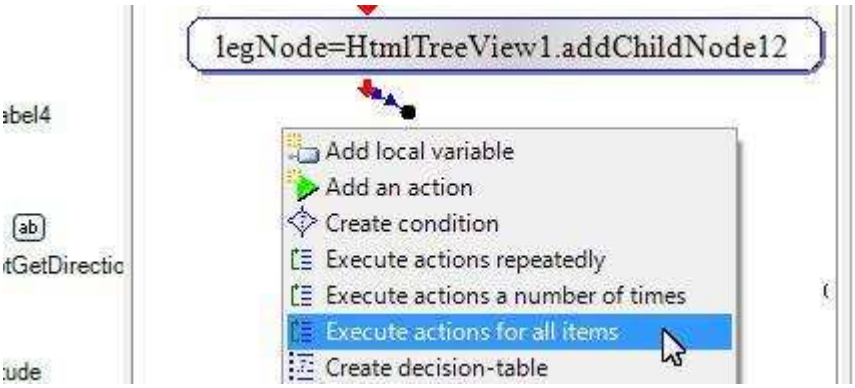


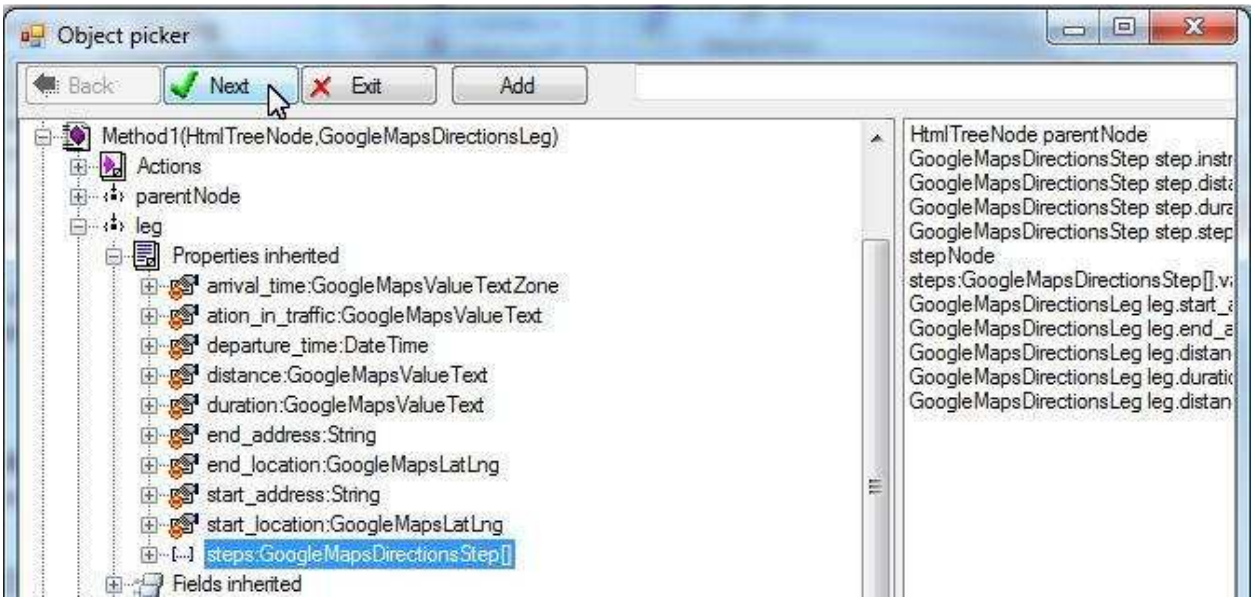
The action and variable legNode appear:



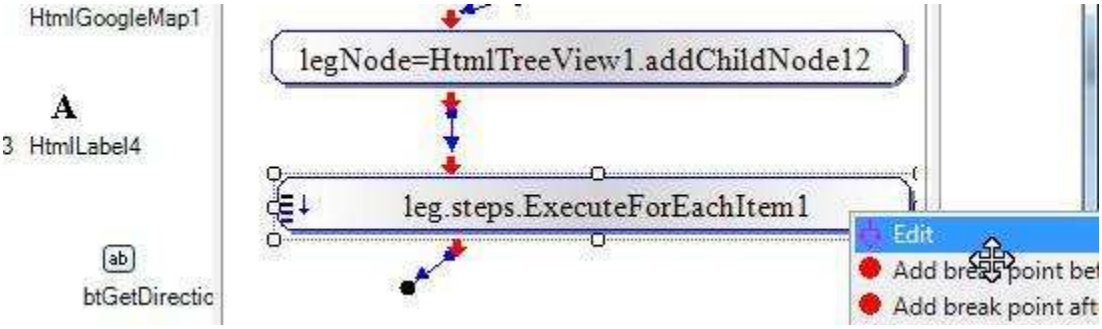
Display steps

A leg has a steps property, which is an array of steps. We may create an "Execute actions for all items" action to display all steps.

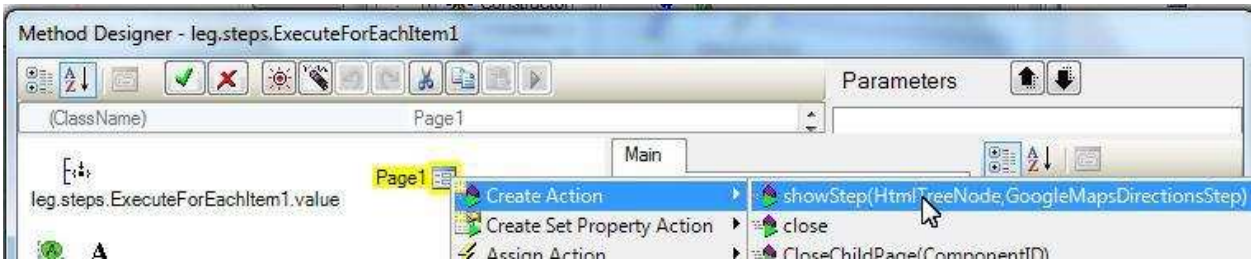




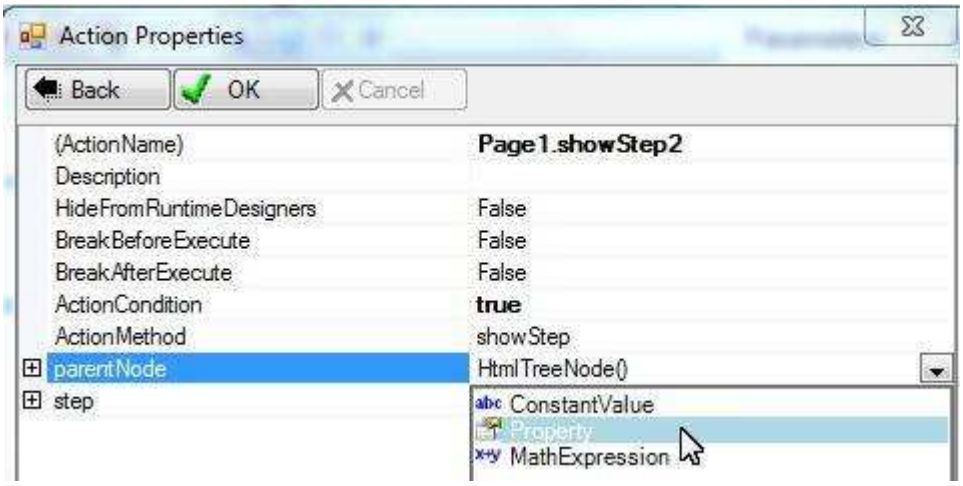
Edit the action to process each step:

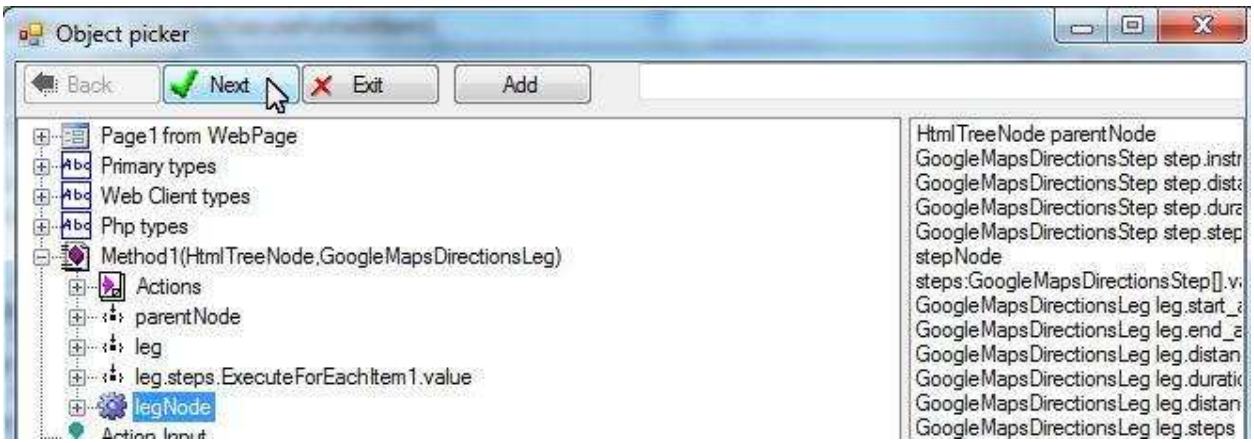


A method editor appears for adding actions. Add a “showStep” action to display the step:

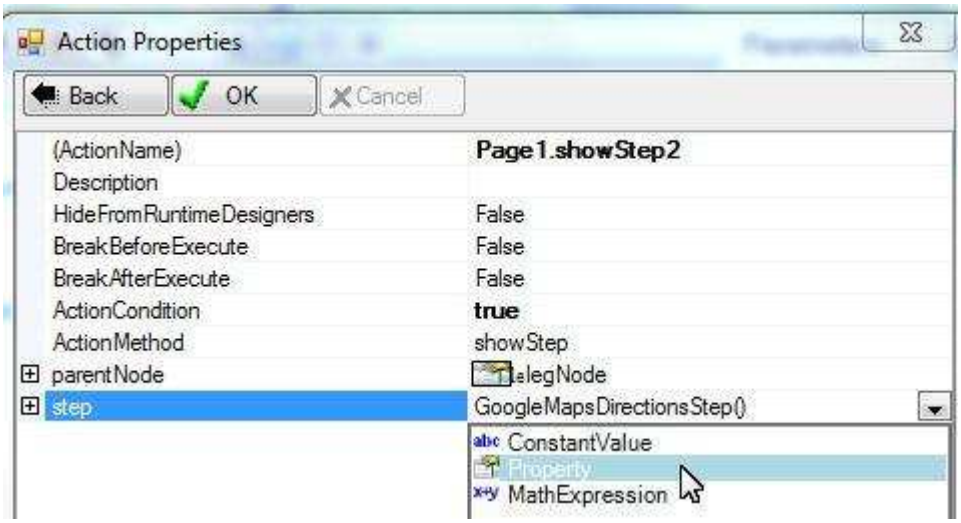


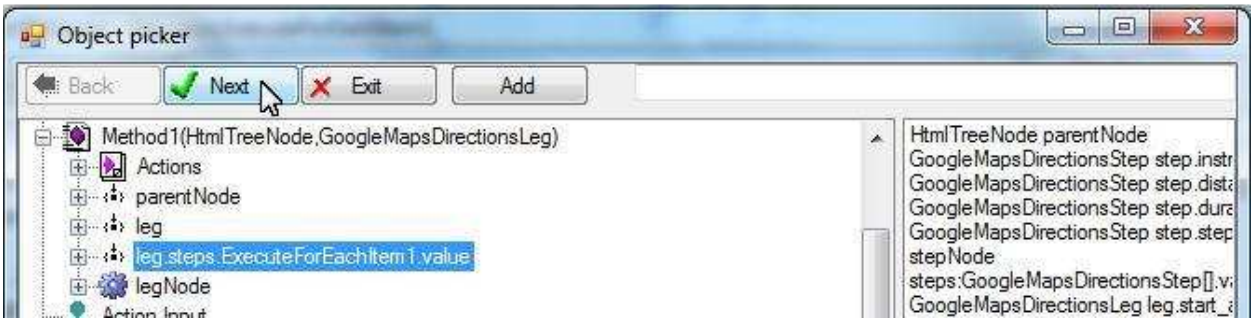
Use the legNode variable as the parentNode:

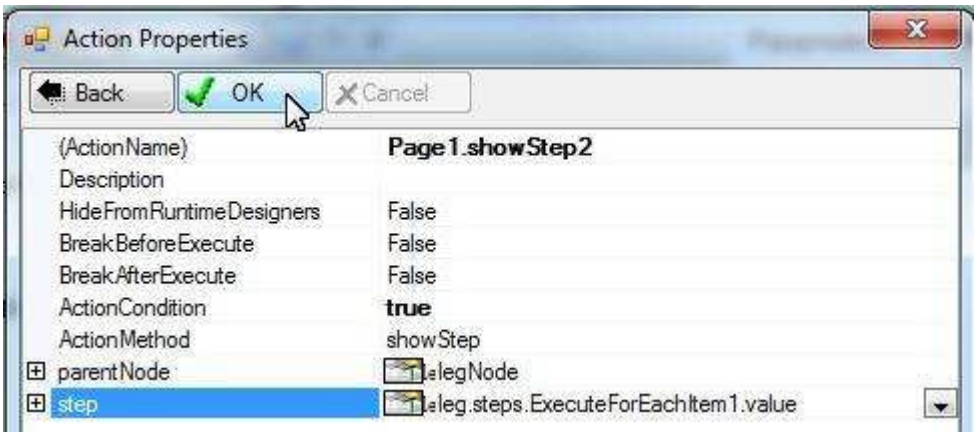




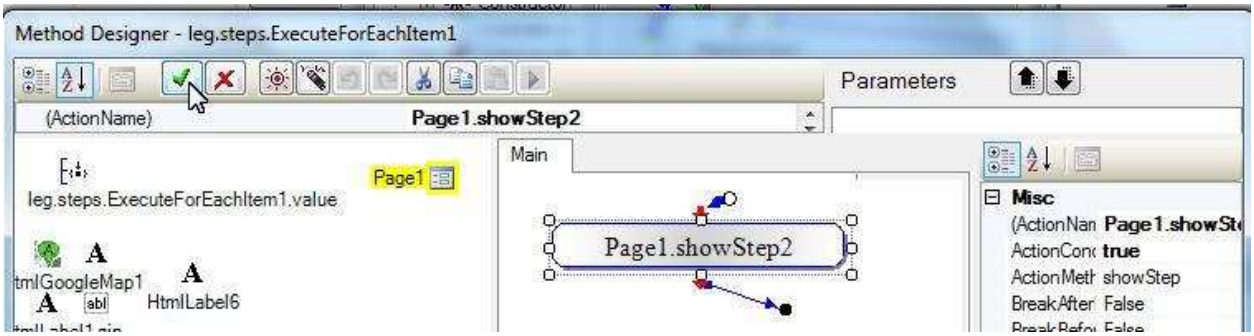
Select the item value for step:

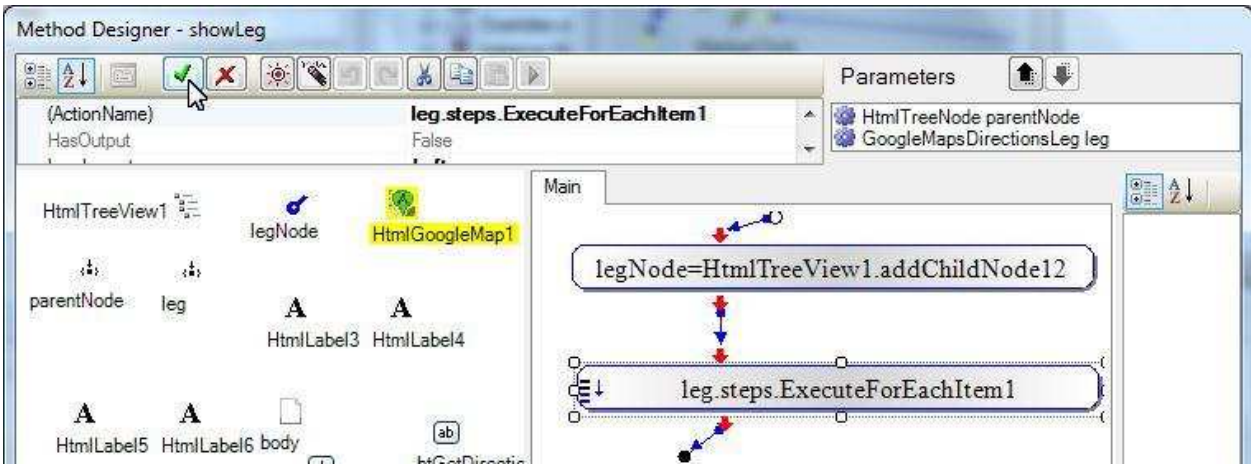






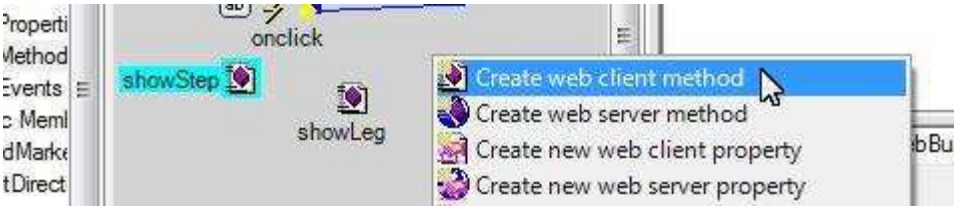
That is all for this sample.



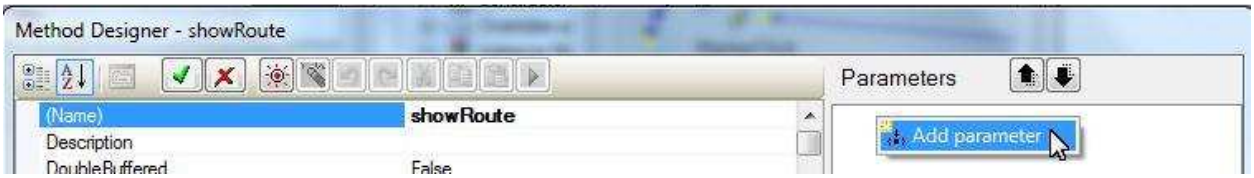


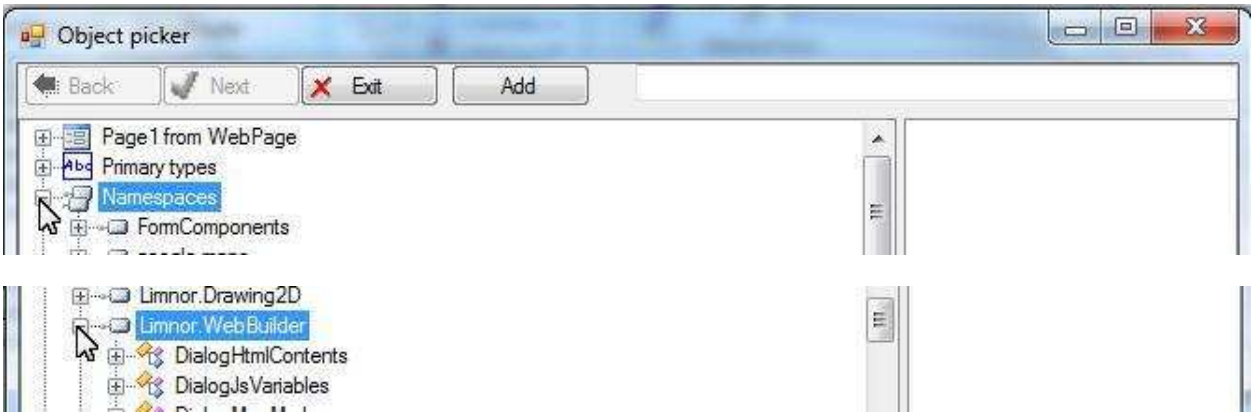
Show route

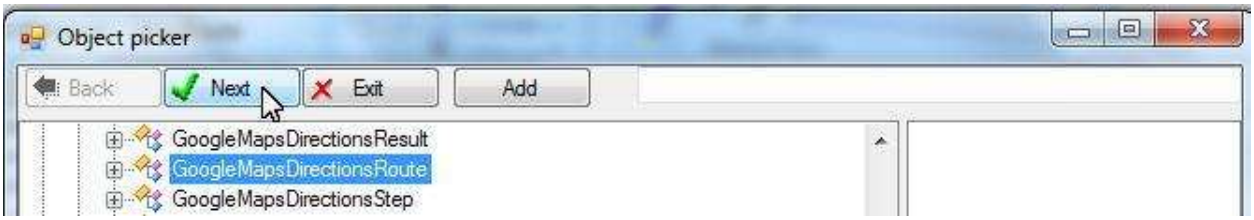
Add a method to display a route.



Add route parameter

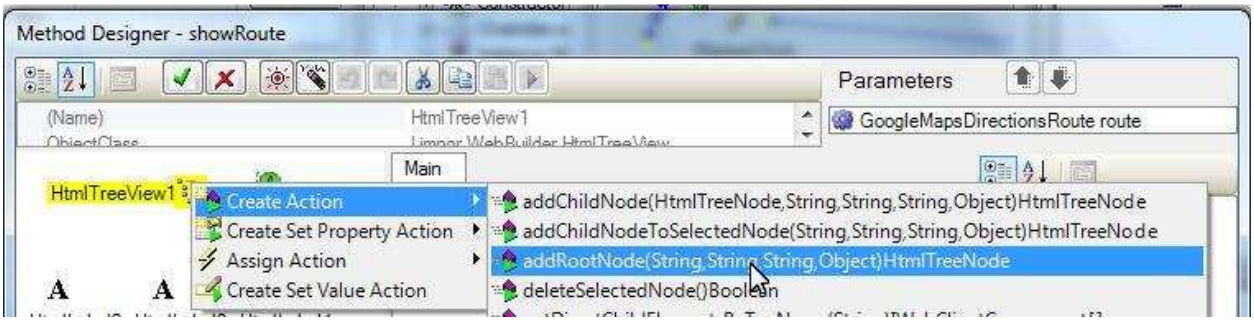




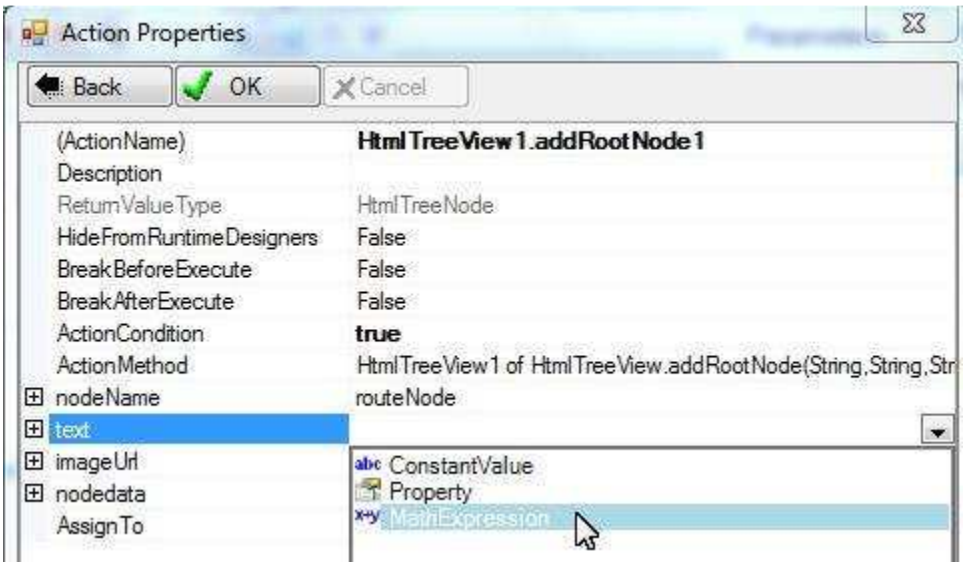


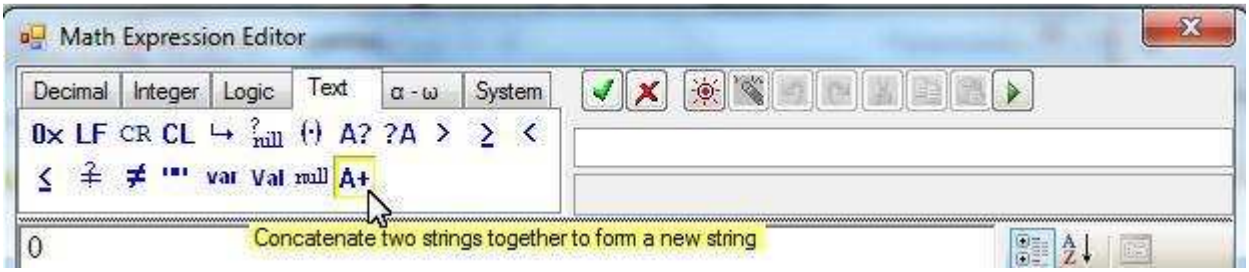
Create a root node

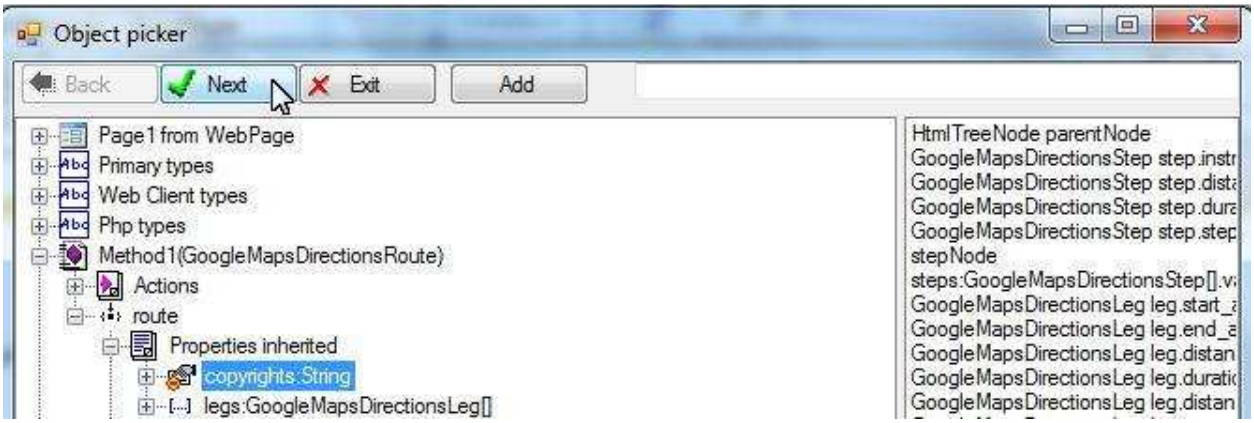
We may create a root node for the route:



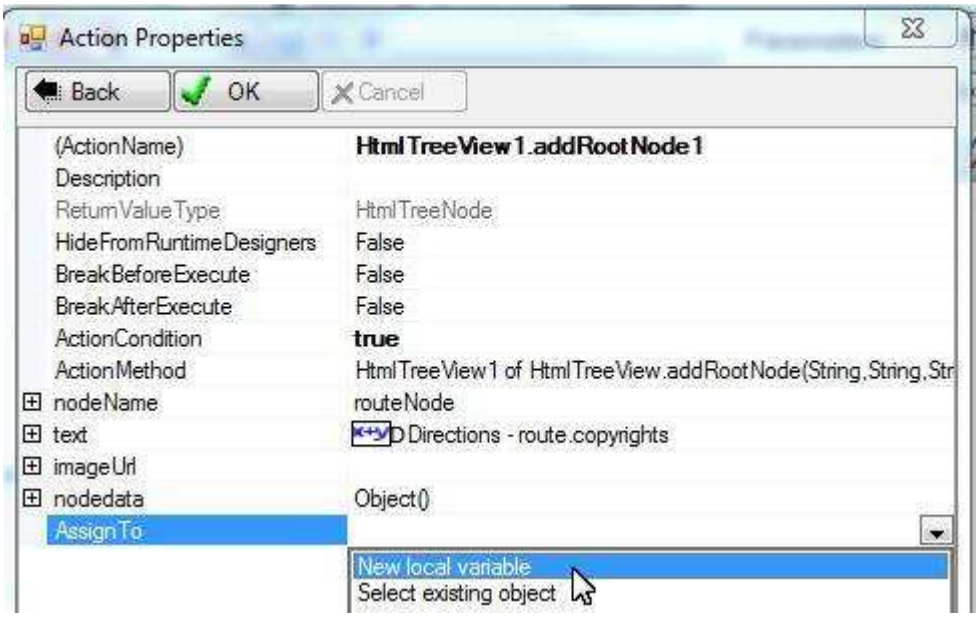
Form display text using an expression:

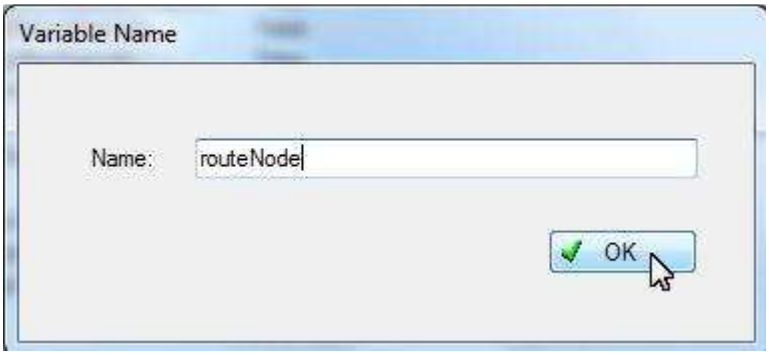


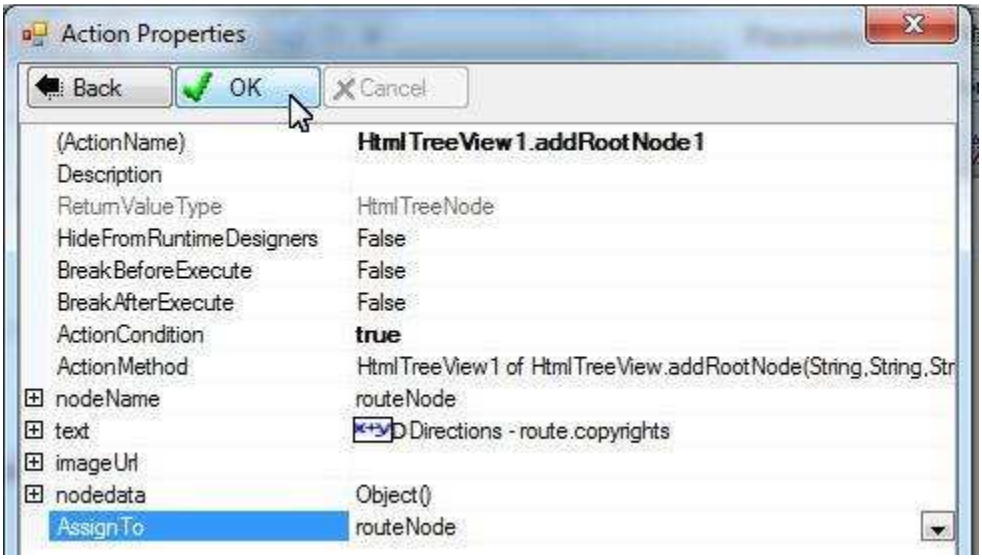




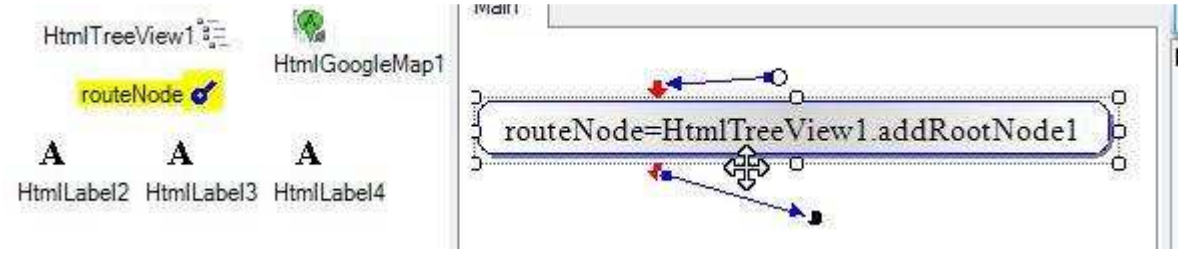
Use a variable for the new route node:





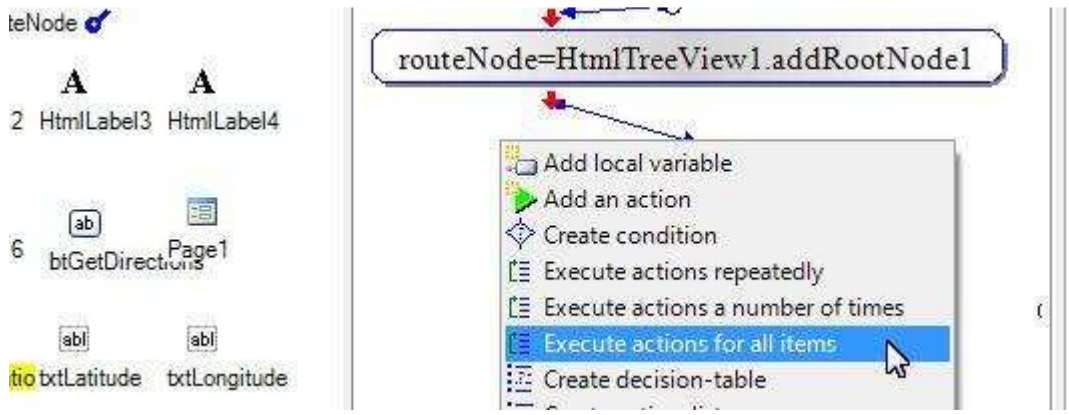


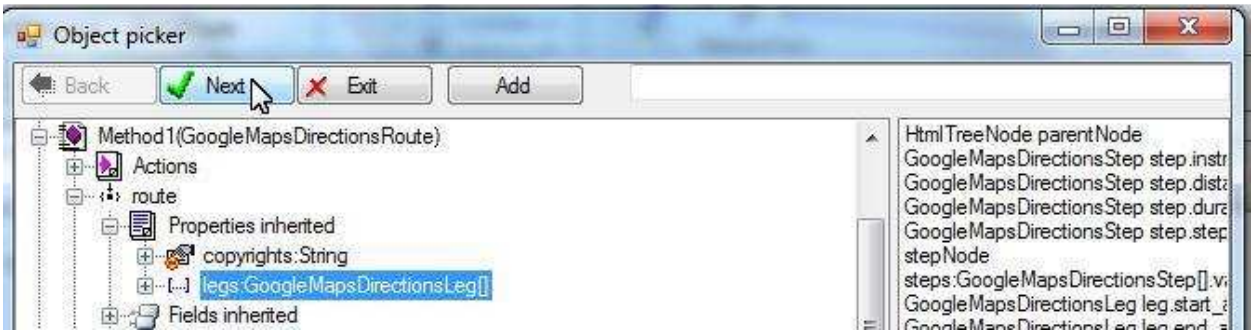
The action and the variable appear.



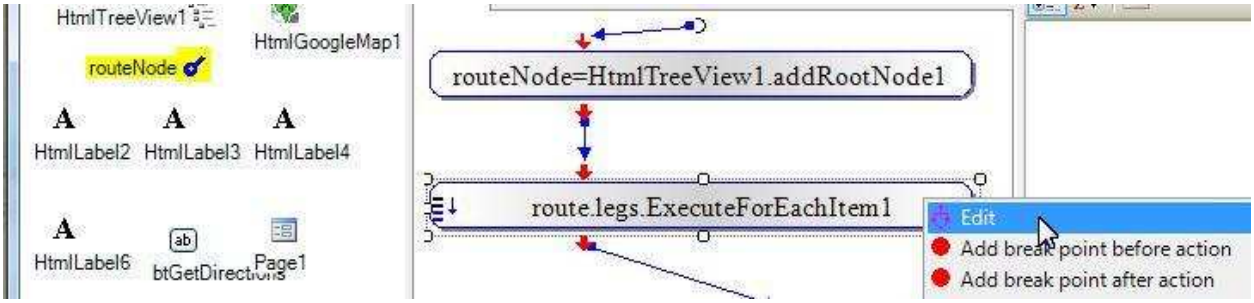
Display legs

Property “legs” is an array of legs. We may use an “Execute actions for each item” to display all legs.

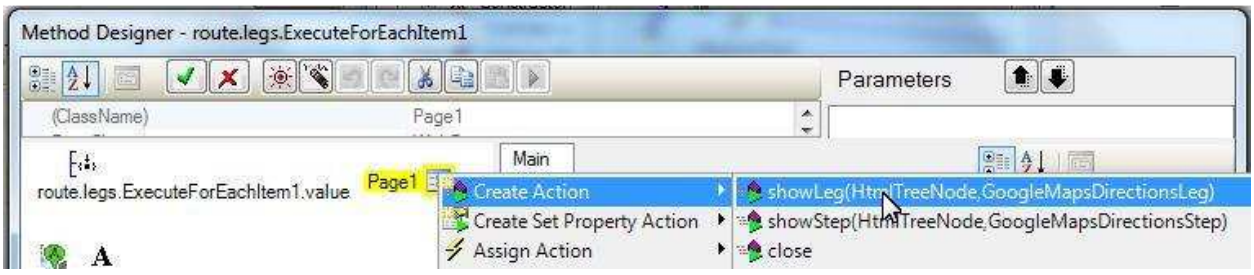




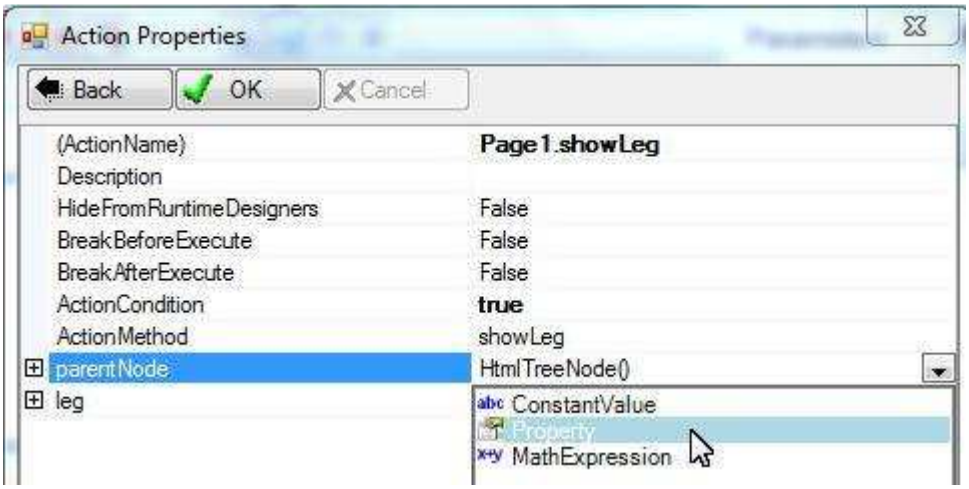
Edit the action to process each leg:

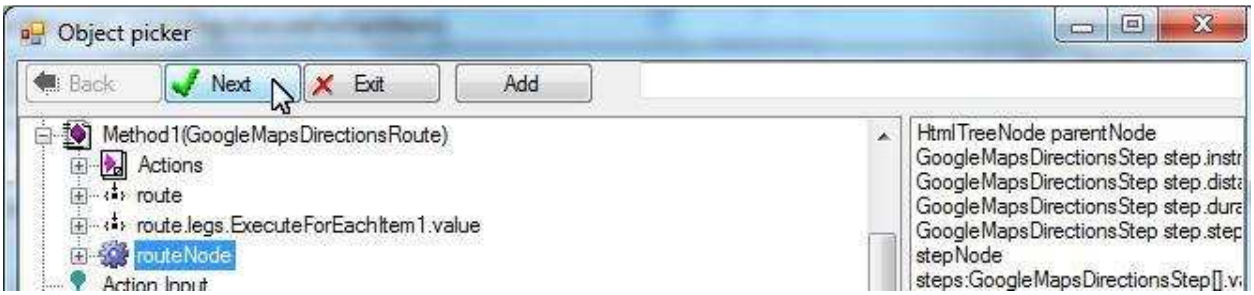


A method editor appears for adding actions. Create a showLeg action to display the leg:

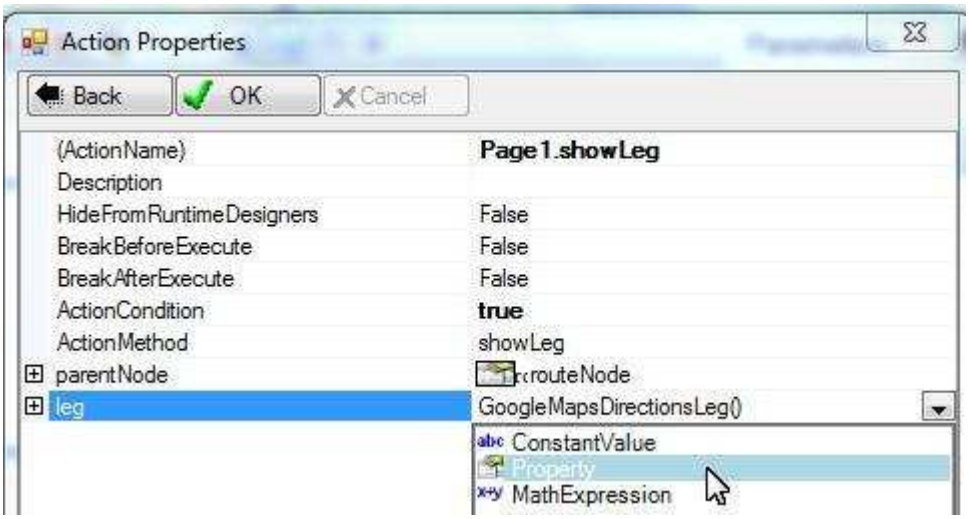


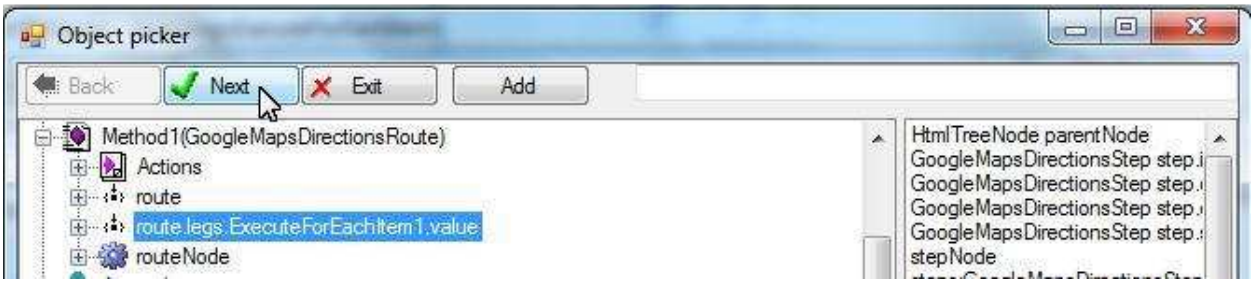
Use routeNode variable as the parent node:



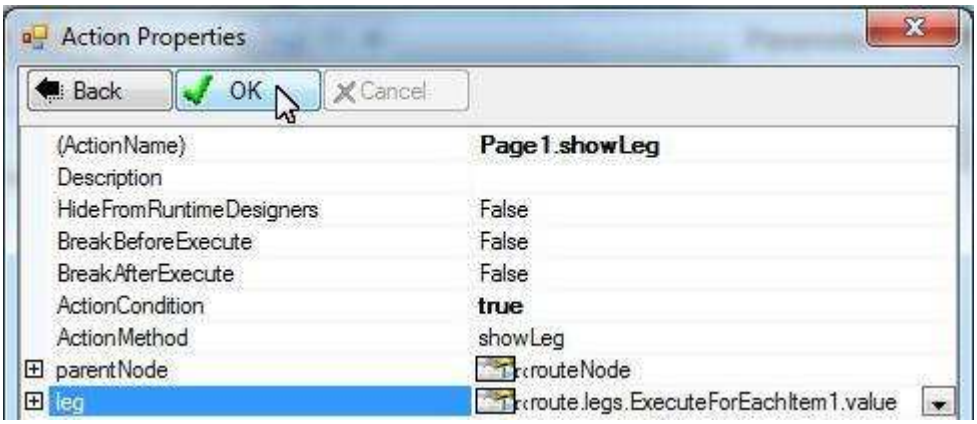


Select “value” for the leg:

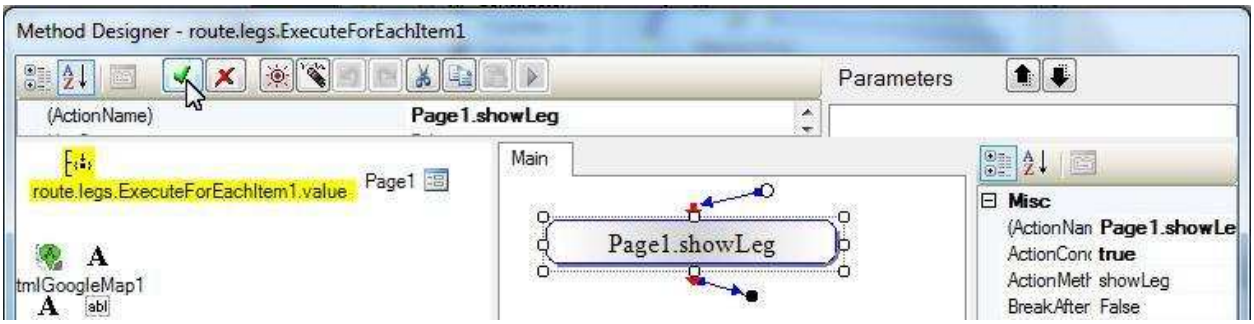


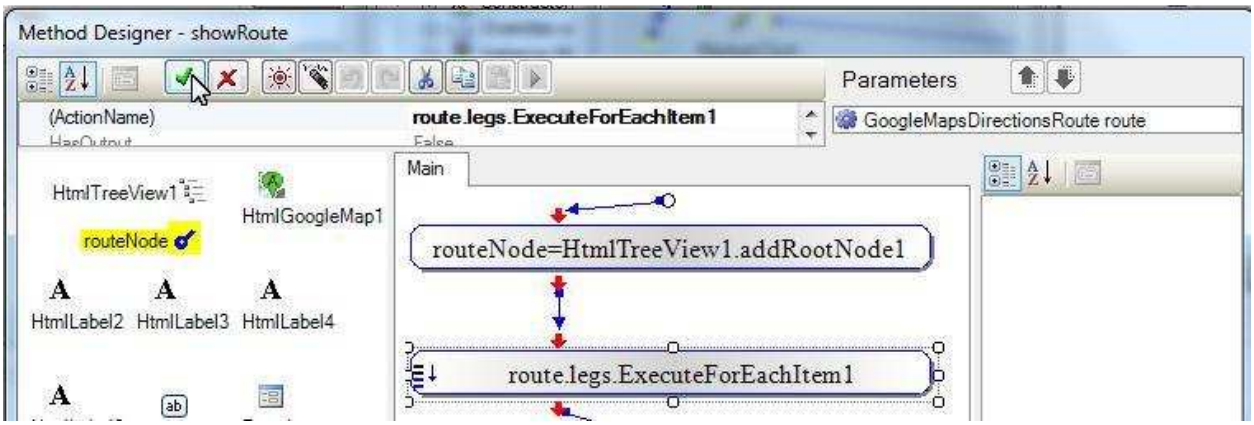


Click OK:



That is all for our sample:



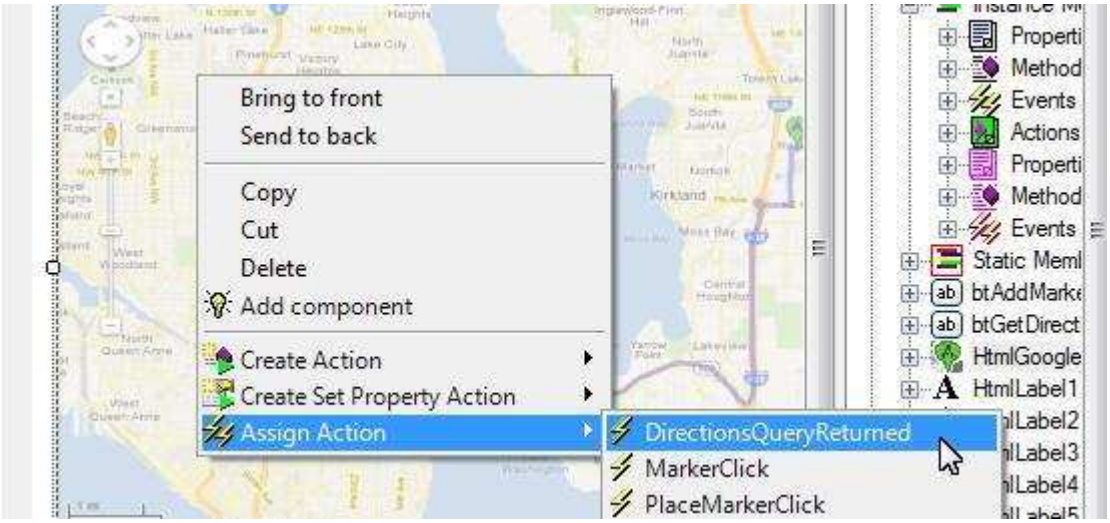


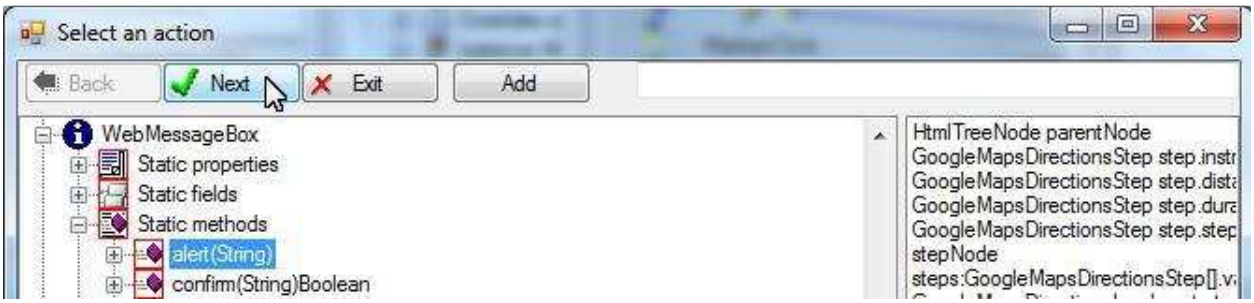
Handle Directions Query Event

Event `DirectionsQueryReturned` occurs when Google Maps service responds to a `GetDirections` action. At this event we may display route.

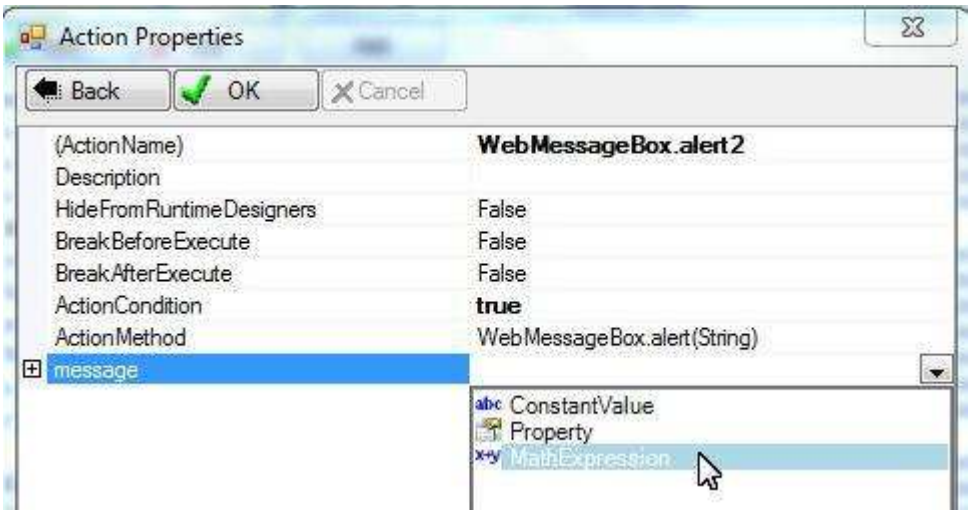
Show status

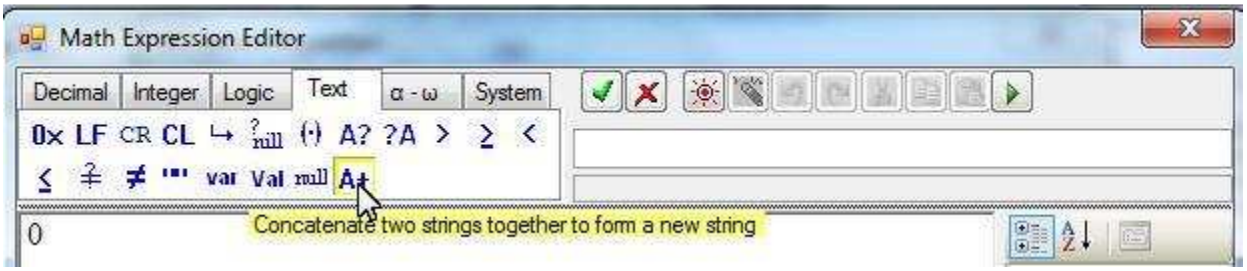
As a sample, we first use a message box to display directions service status:



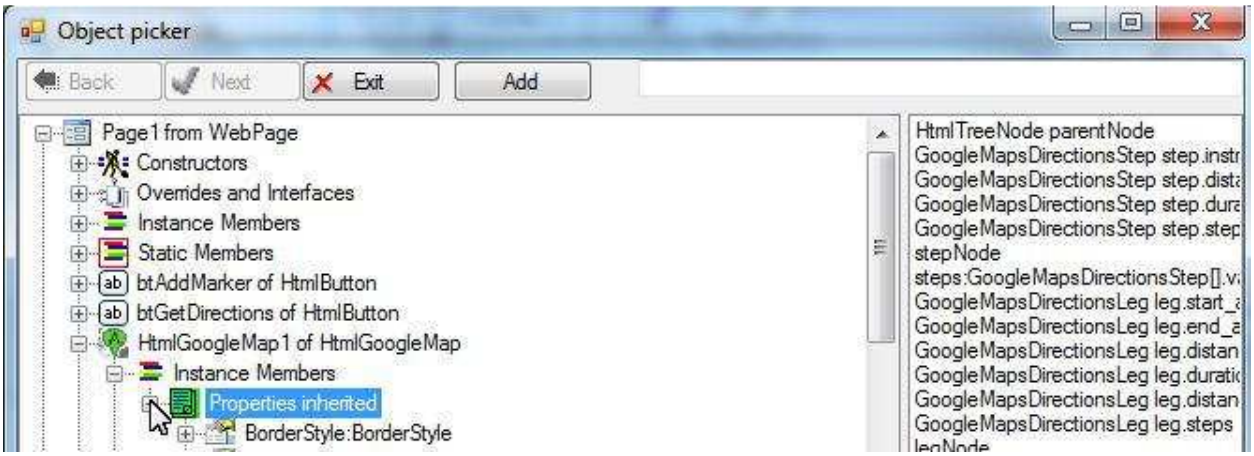


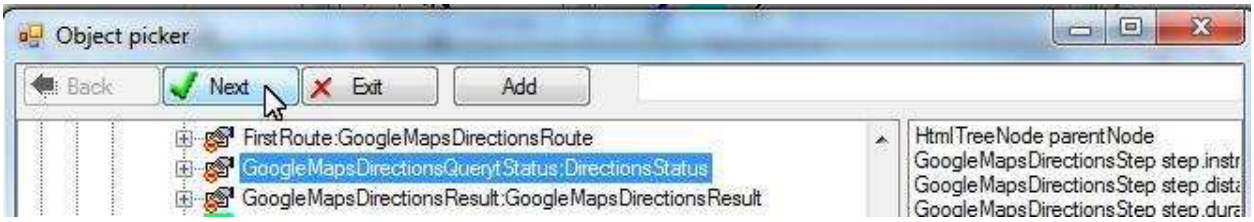
Form a display expression:

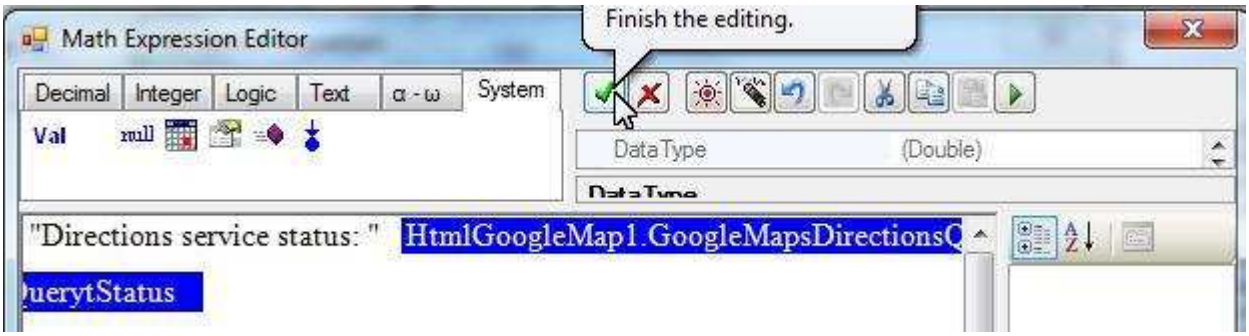


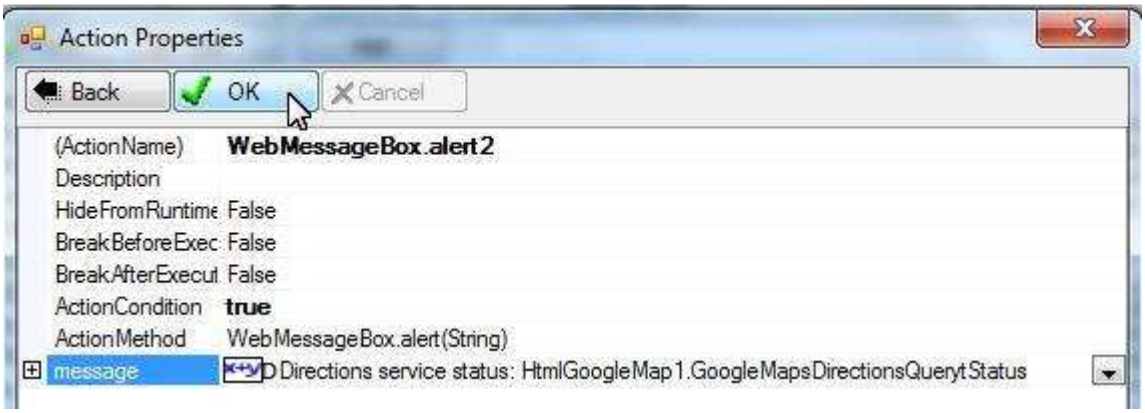

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Concatenate two strings together to form a new string
0


Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(String)
"Directions service status: "
A property

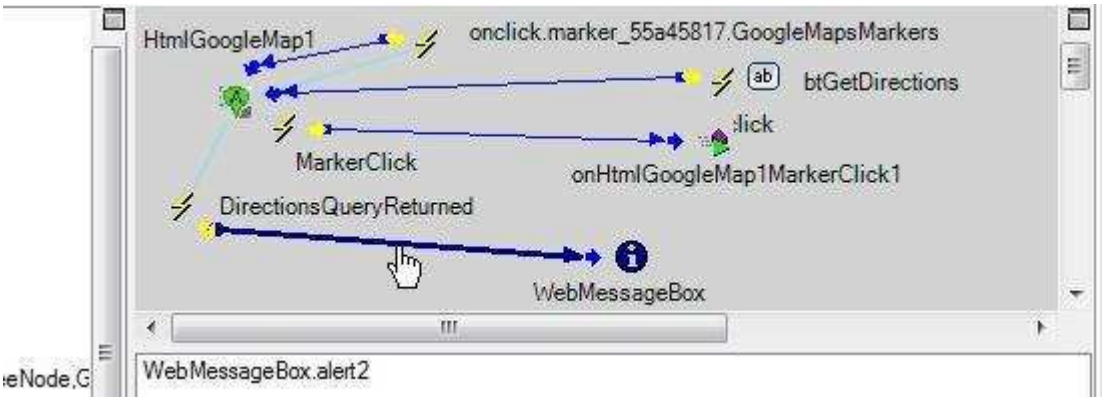

Object picker
Back
Next
Exit
Add
Page1 from WebPage
Constructors
Overrides and Interfaces
Instance Members
Static Members
btAddMarker of HtmlButton
btGetDirections of HtmlButton
HtmlGoogleMap1 of HtmlGoogleMap
Instance Members
Properties inherited
BorderStyle:BorderStyle
HtmlTreeNode parentNode
stepNode


Object picker
Back
Next
Exit
Add
FirstRoute:GoogleMapsDirectionsRoute
GoogleMapsDirectionsQueryStatus:DirectionsStatus
GoogleMapsDirectionsResult:GoogleMapsDirectionsResult
HtmlTreeNode parentNode


Finish the editing.
Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(Double)
"Directions service status: "
HtmlGoogleMap1.GoogleMapsDirectionsQ
uerytStatus

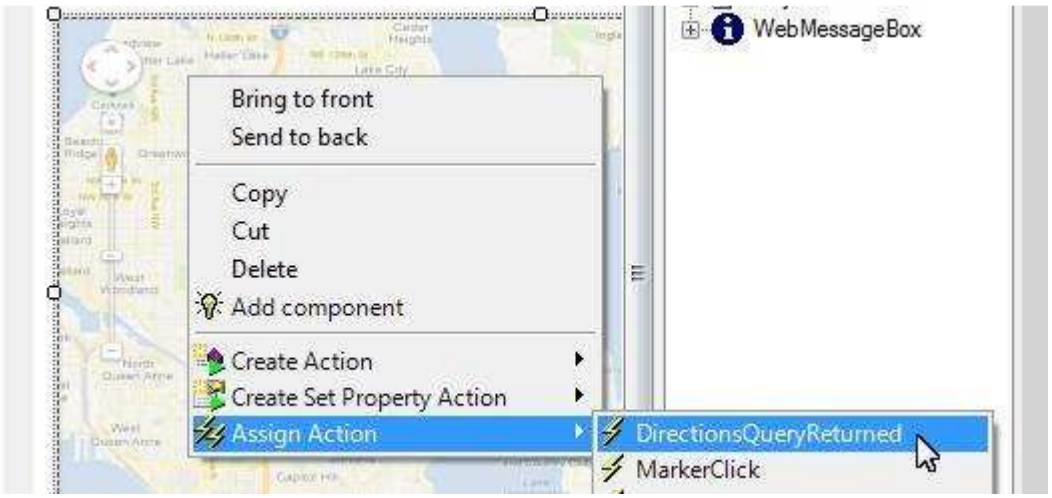


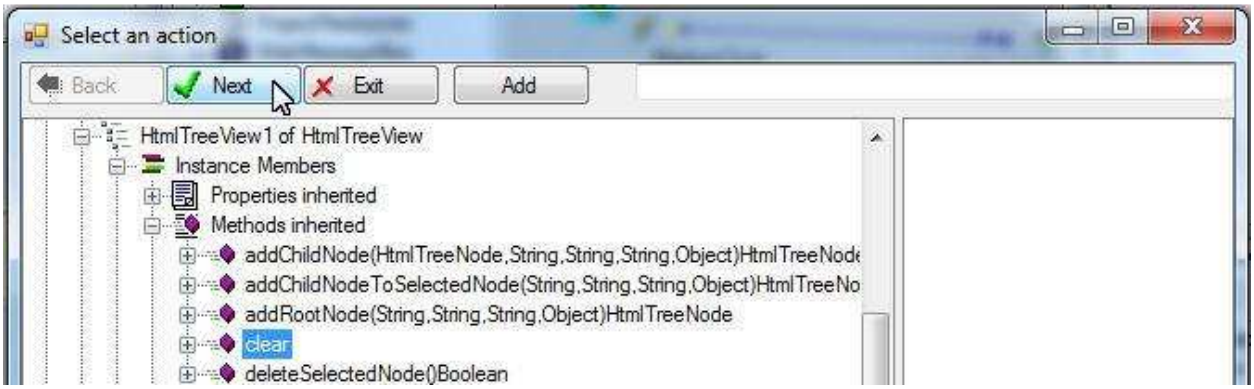
The action is created and assigned to the event:

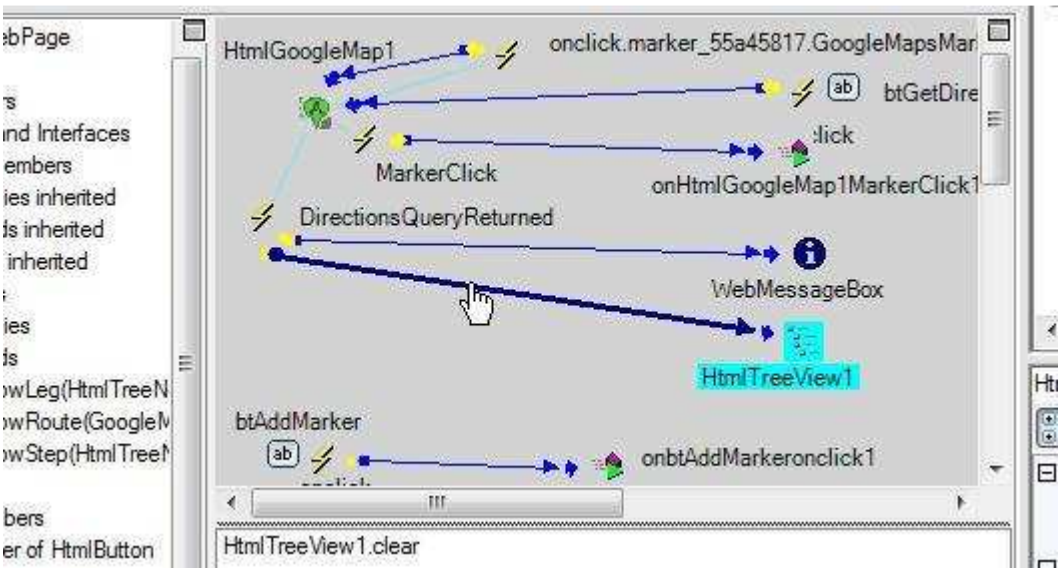


Clear tree view

Before displaying directions we need to remove all existing tree nodes at this event:

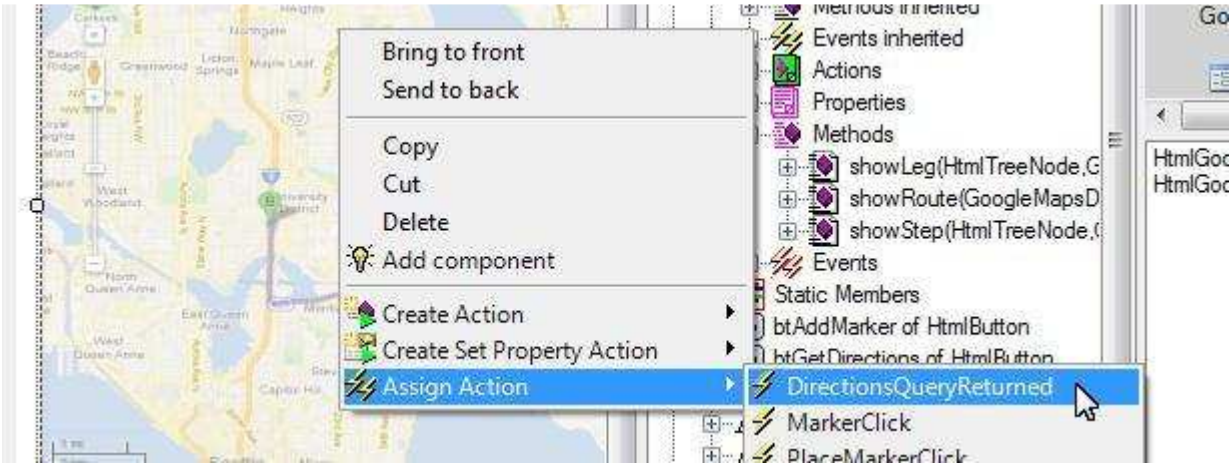


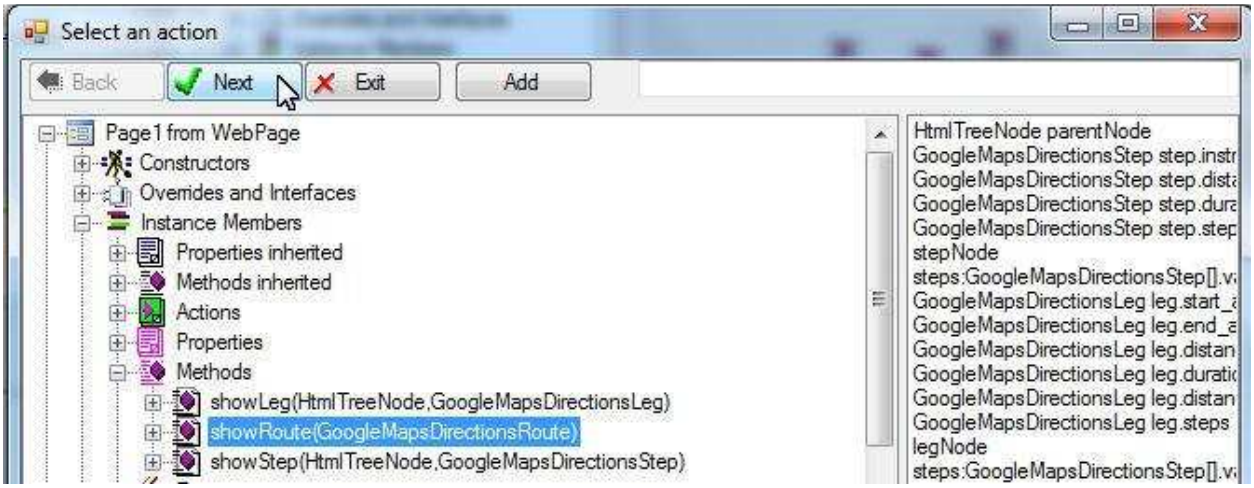




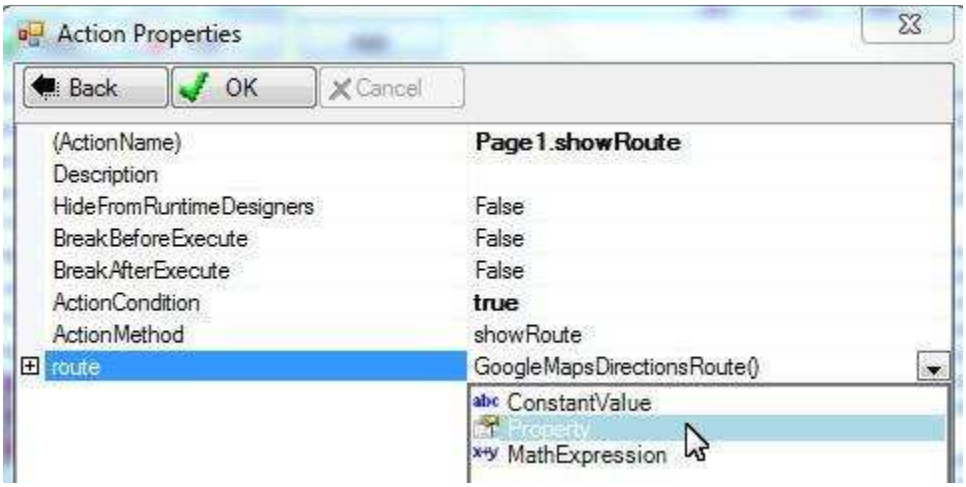
Show route in tree view

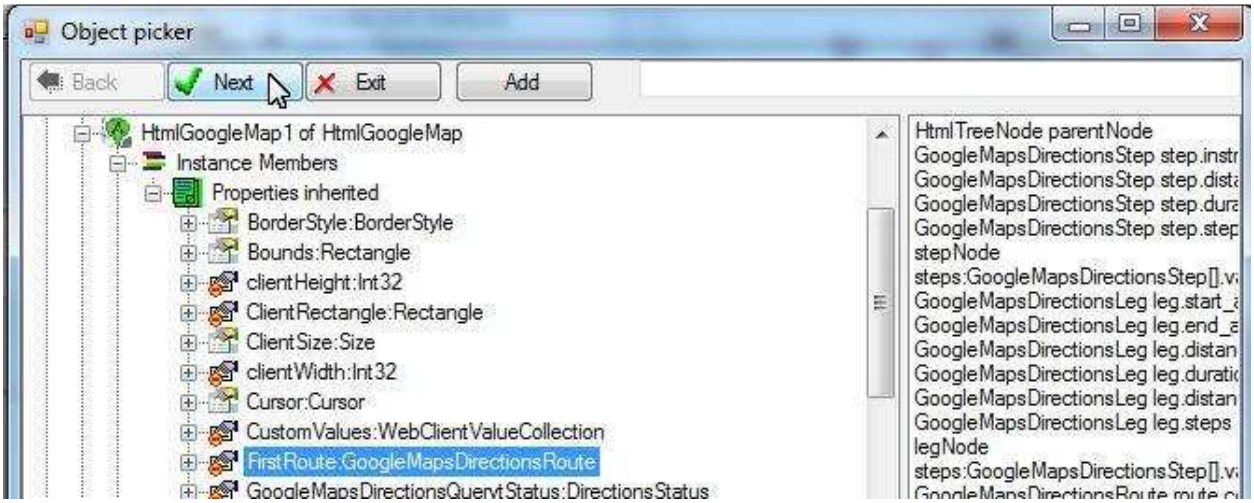
We create a showRoute action to display the first route:



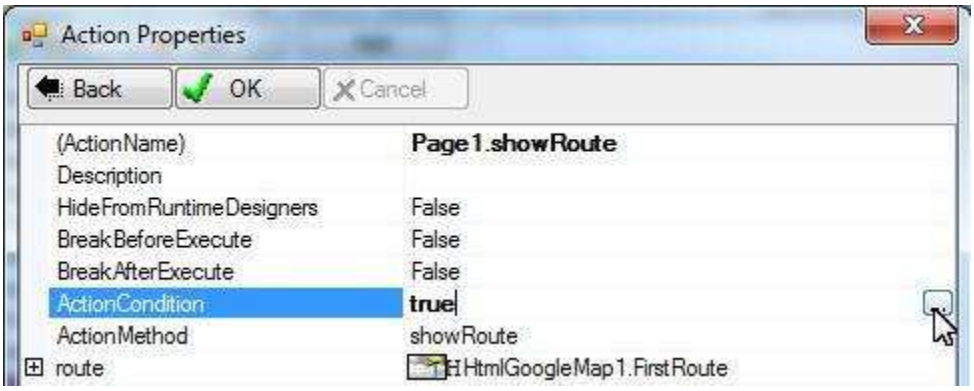


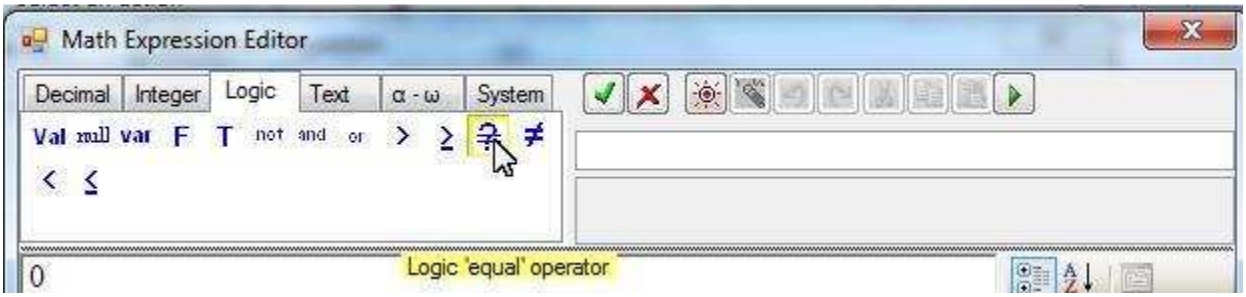
Select the first route:

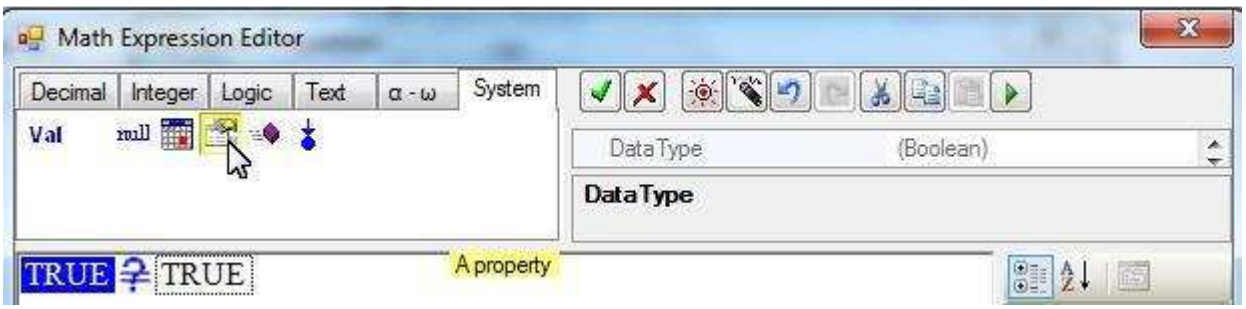


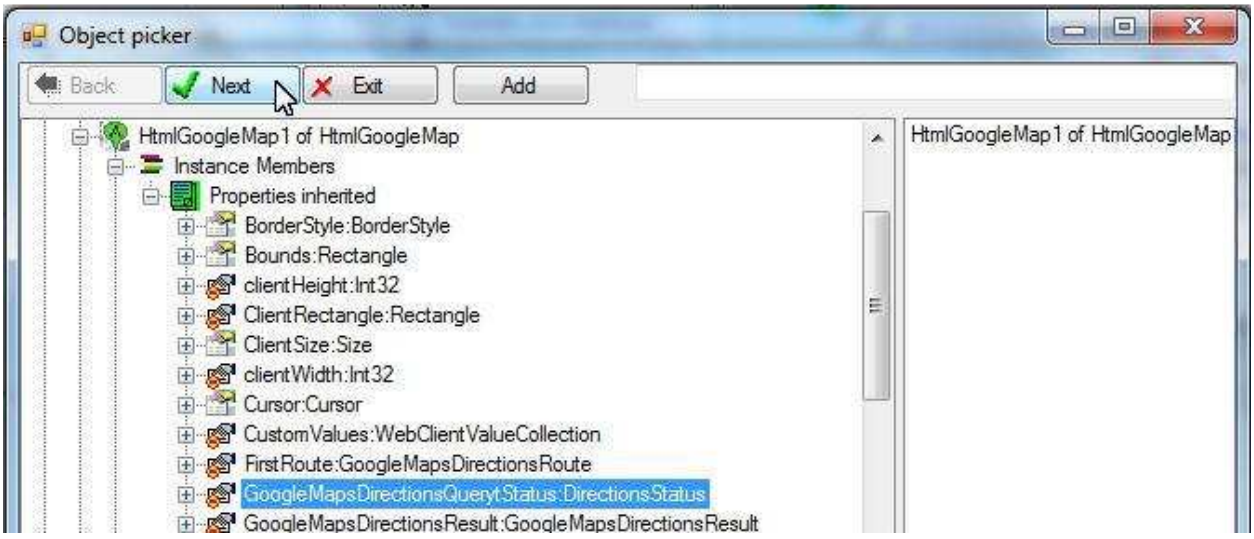


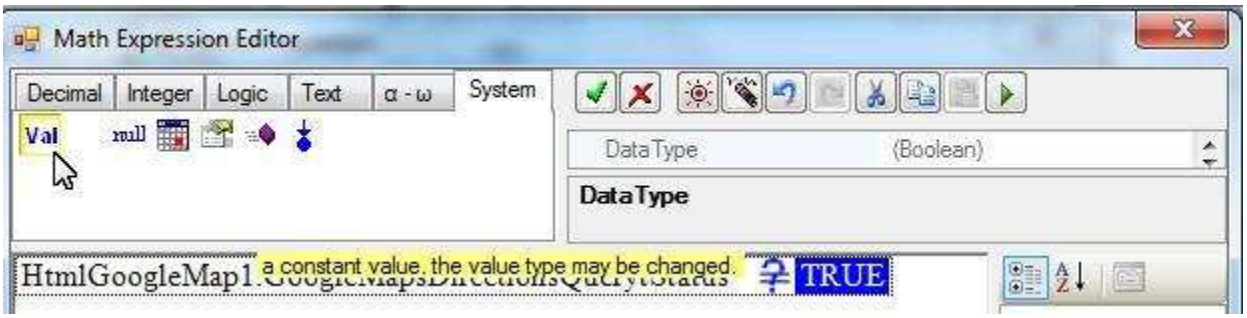
Set action condition so that the action is executed only if directions status is OK:

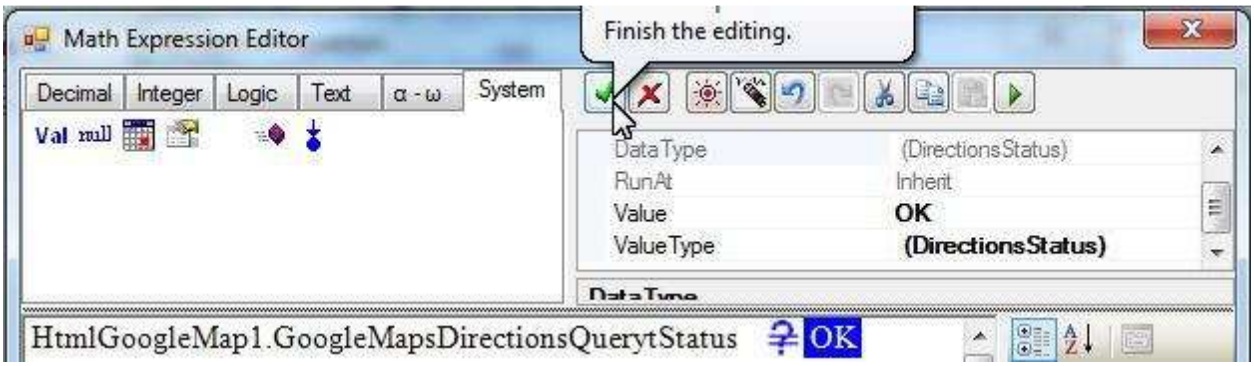


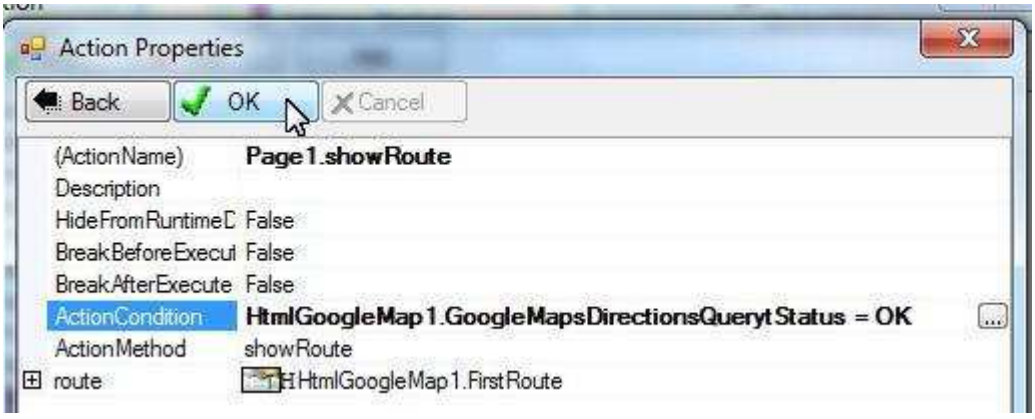




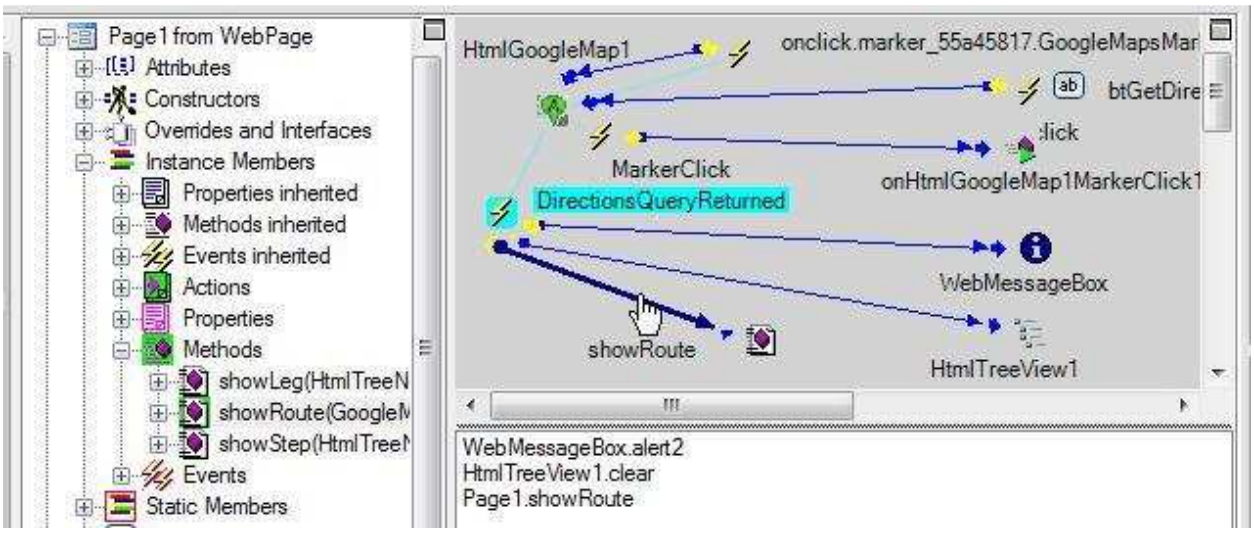

Object picker
Back
Next
Exit
Add
HtmlGoogleMap1 of HtmlGoogleMap
Instance Members
Properties inherited
BorderStyle:BorderStyle
Bounds:Rectangle
clientHeight:Int32
ClientRectangle:Rectangle
ClientSize:Size
clientWidth:Int32
Cursor:Cursor
CustomValues:WebClientValueCollection
FirstRoute:GoogleMapsDirectionsRoute
GoogleMapsDirectionsQueryt Status:DirectionsStatus
GoogleMapsDirectionsResult:GoogleMapsDirectionsResult
HtmlGoogleMap1 of HtmlGoogleMap


Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(Boolean)
DataType
HtmlGoogleMap1
a constant value, the value type may be changed.
TRUE


Math Expression Editor
Finish the editing.
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(DirectionsStatus)
RunAt
Inherit
Value
OK
ValueType
(DirectionsStatus)
HtmlGoogleMap1.GoogleMapsDirectionsQuerytStatus
OK

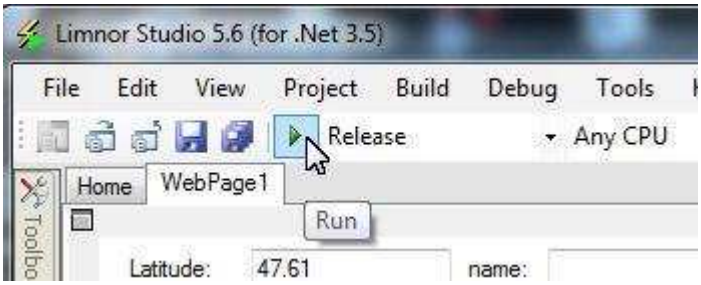

Action Properties
Back
OK
Cancel
(ActionName)
Page1.showRoute
Description
HideFromRuntimeC
False
BreakBeforeExecut
False
BreakAfterExecute
False
ActionCondition
HtmlGoogleMap1.GoogleMapsDirectionsQuerytStatus = OK
ActionMethod
showRoute
route
HtmlGoogleMap1.FirstRoute

The action is created and assigned to the event.

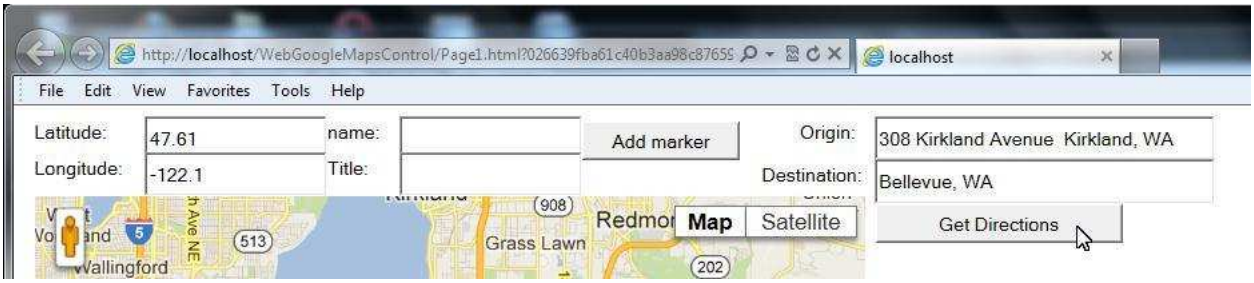


Test

We may test directions services now.

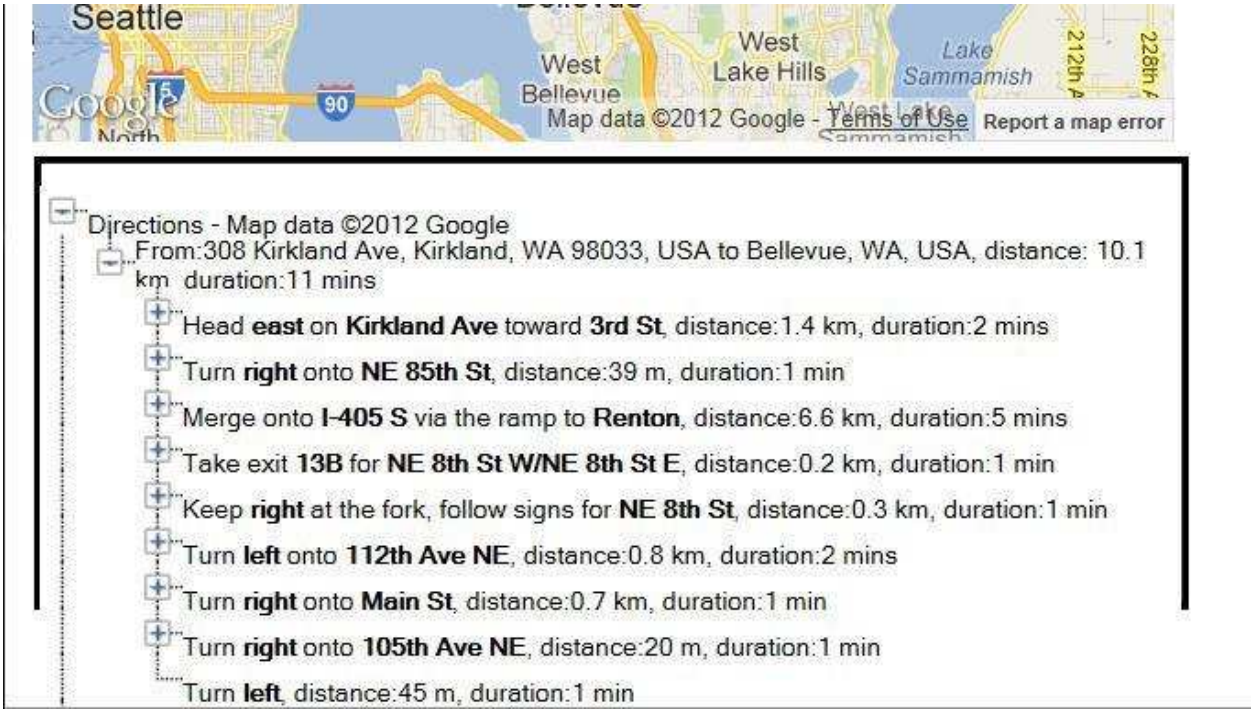


The web page appears. Enter origin and destination. Click “Get Directions”.



A message box appears:

Dismiss the message box. In addition to the markings on the map, the tree view is filled with data:



Places Search

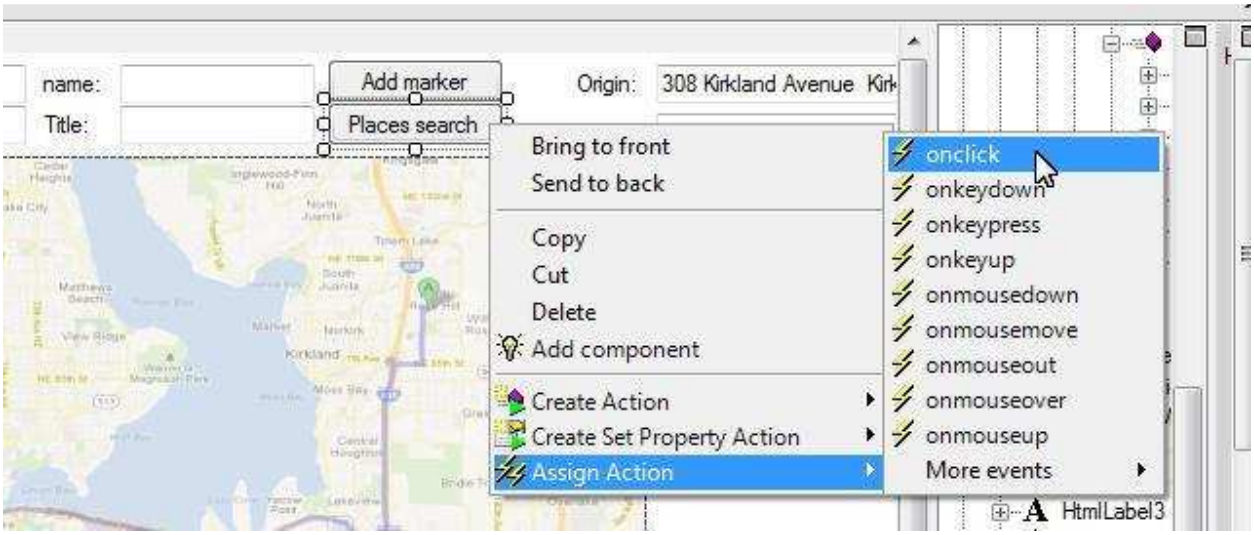
HtmlGoogleMap may be used to search places.

- A **`SearchPlacesByLocation`** action sends places search request to Google Maps services.

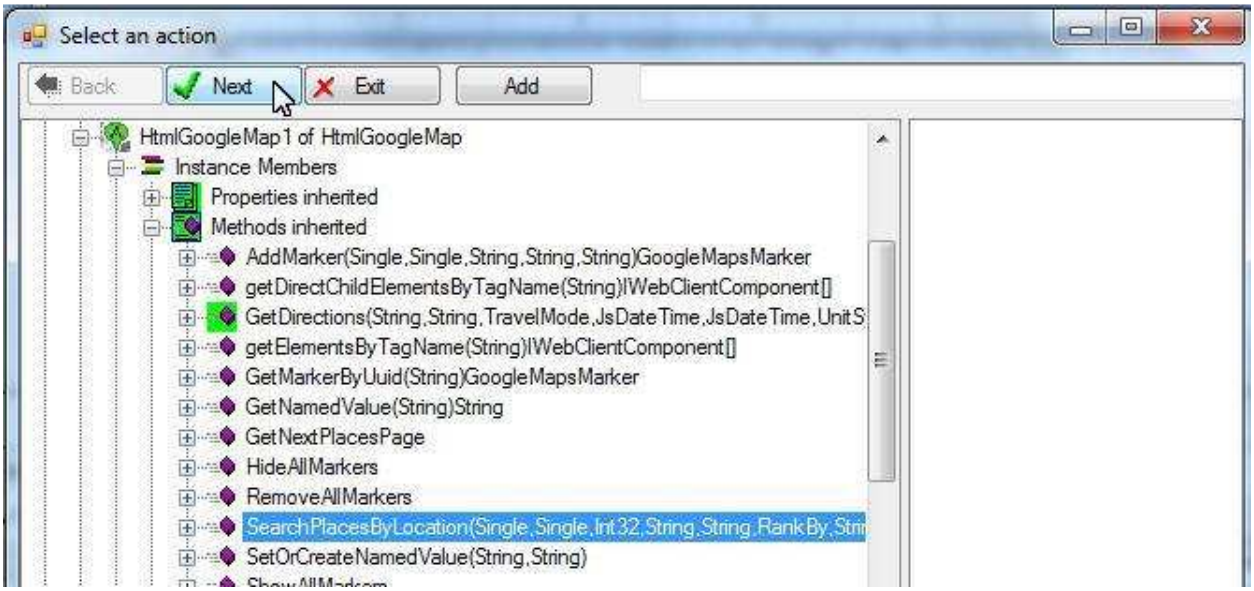
- Event **`PlacesQueryReturned`** occurs when Google Maps services respond to search request and get-next-page request.
- Property **`GoogleMapsPlacesQuerytStatus`** indicates status on Google Maps responding to search request and get-next-page request.
- Google Maps services return places one page at a time. Property **`placesOnLastPage`** is an array containing places on the last page Google Maps services return. Property **`placesCountLastPage`** indicates how many places contained on the last page.
- Property **`GoogleMapsPlacesHasNextPage`** indicates whether there is a next page available from Google Maps services.
- If **`GoogleMapsPlacesHasNextPage`** is True then a **`GetNextPlacesPage`** action fetches the next page. Event **`PlacesQueryReturned`** occurs when Google Maps services respond to the `GetNextPlacesPage` action. Property `placesOnLastPage and placesCountLastPage` are updated on this event.
- Property **`placesAll`** contains all places returned by a `SearchPlacesByLocation` action and all subsequent `GetNextPlacesPage` actions. Property `placesCount` indicates how many places `placesAll` contains.

Action to Search Places

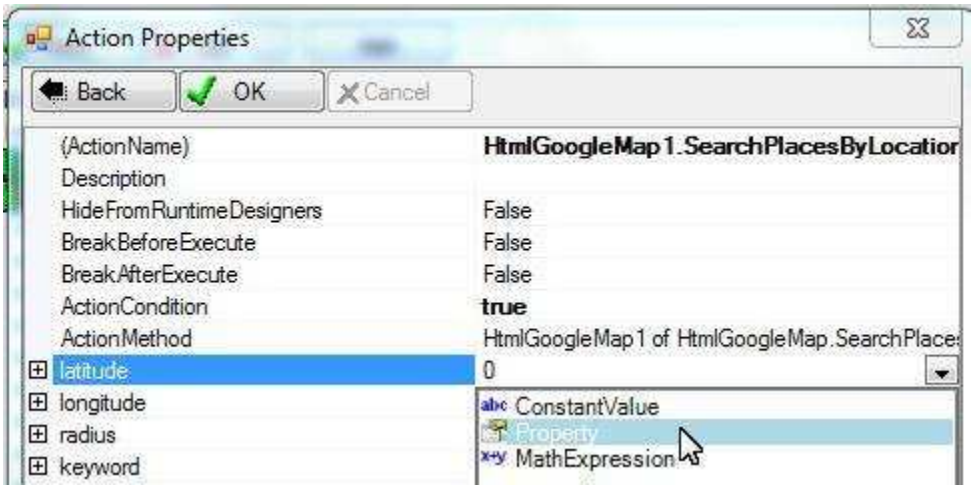
Suppose we use a button to initiate a places-search:

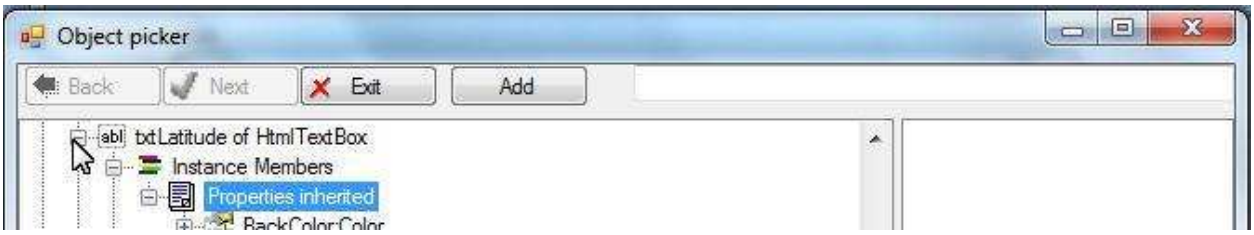


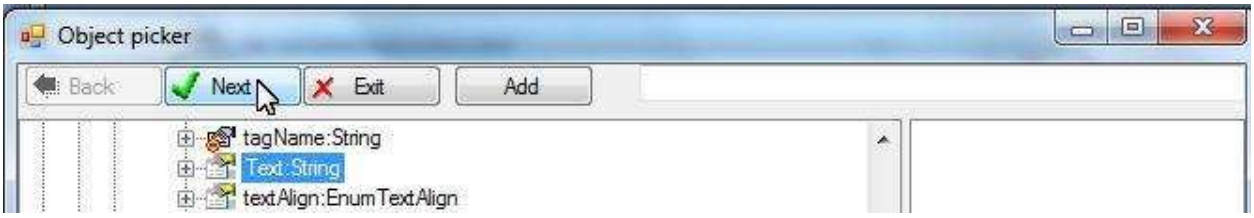
Select `SearchPlacesByLocation`:

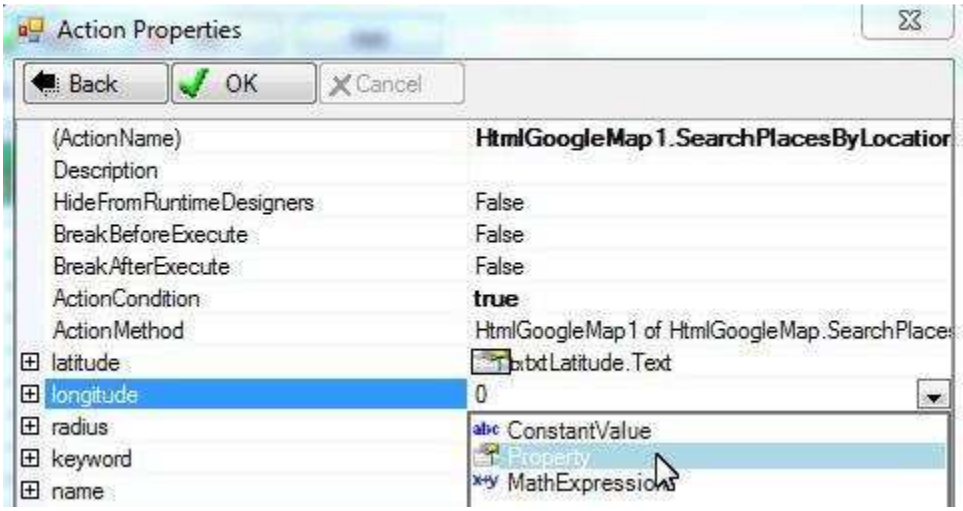


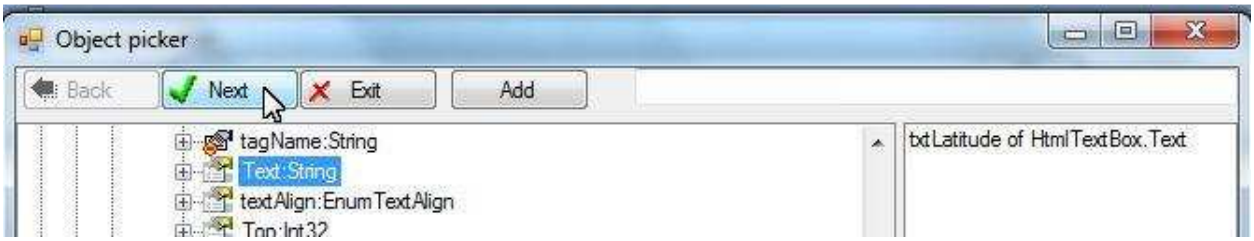
Set search center and radius:



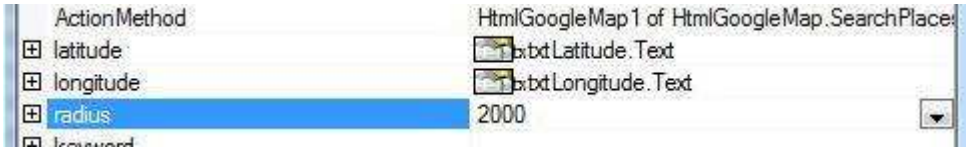




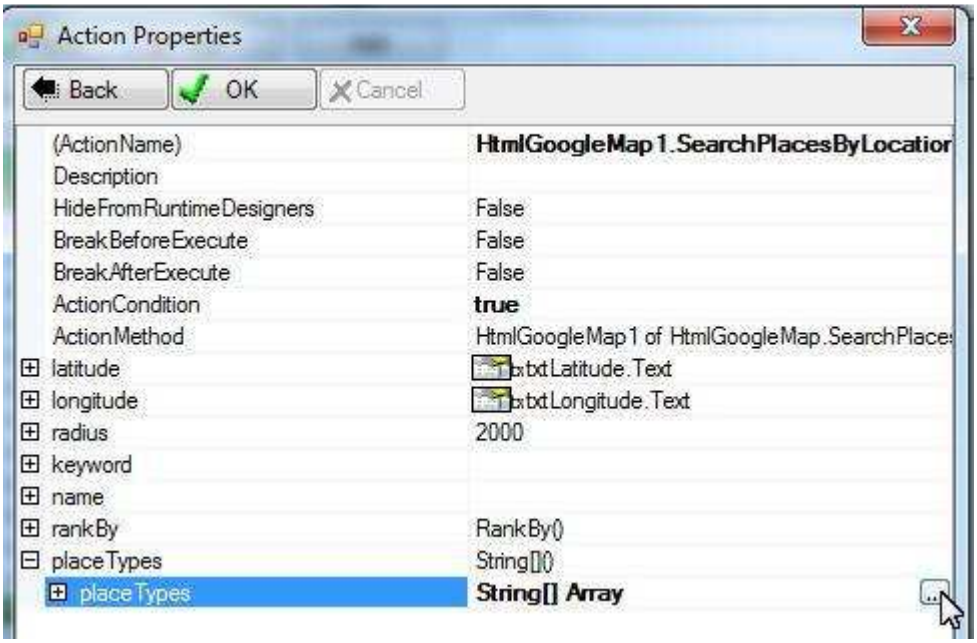




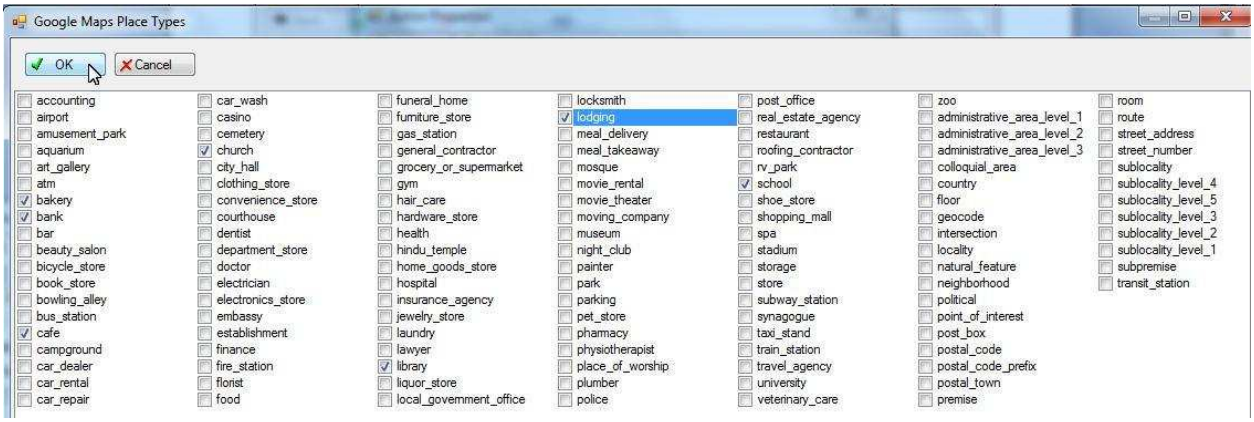
Radius is in meters.



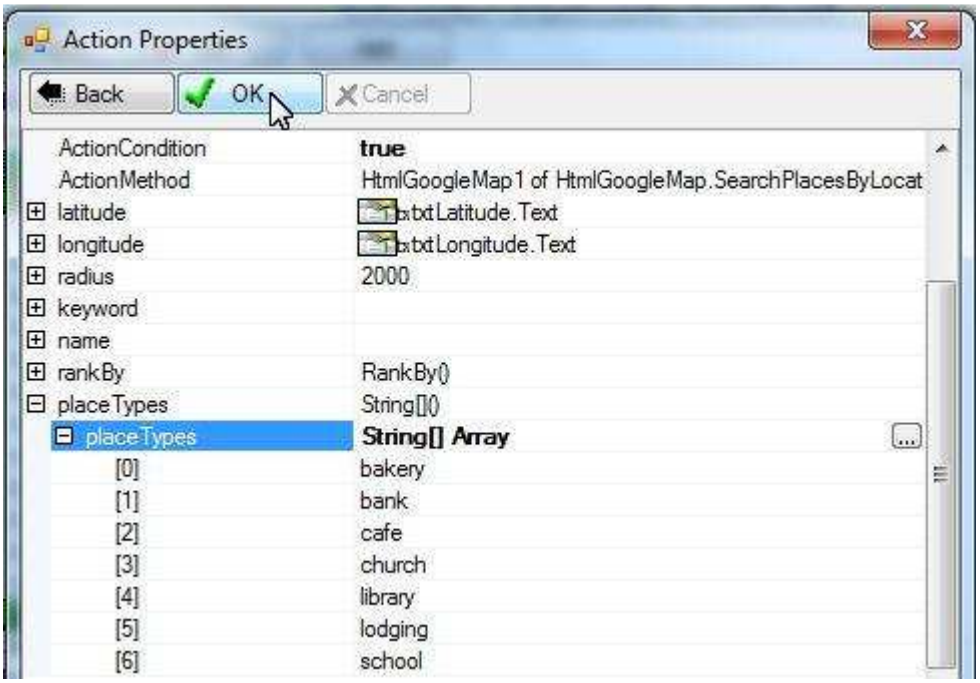
Select places to be searched:



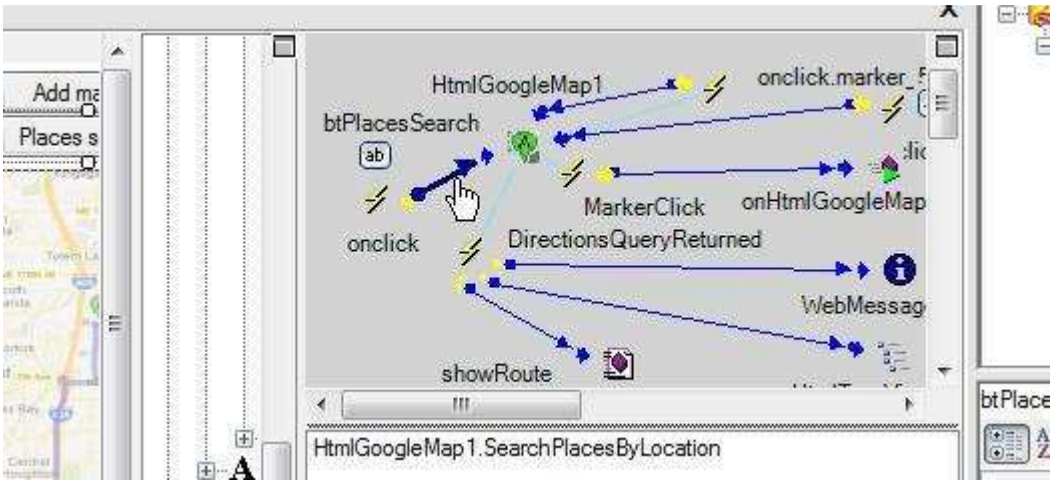
Check all places to be searched:



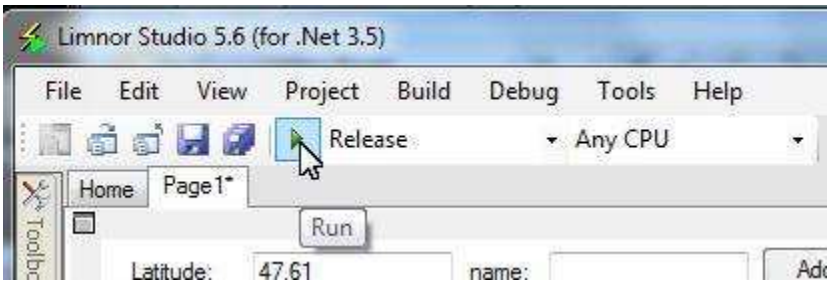
Click OK to create the action:



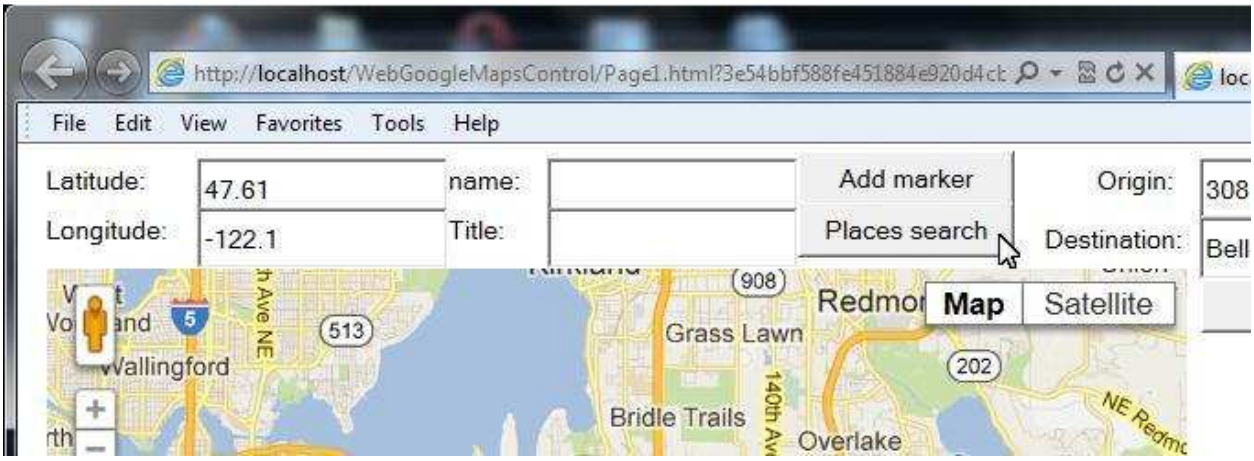
The action is created and assigned to the button:



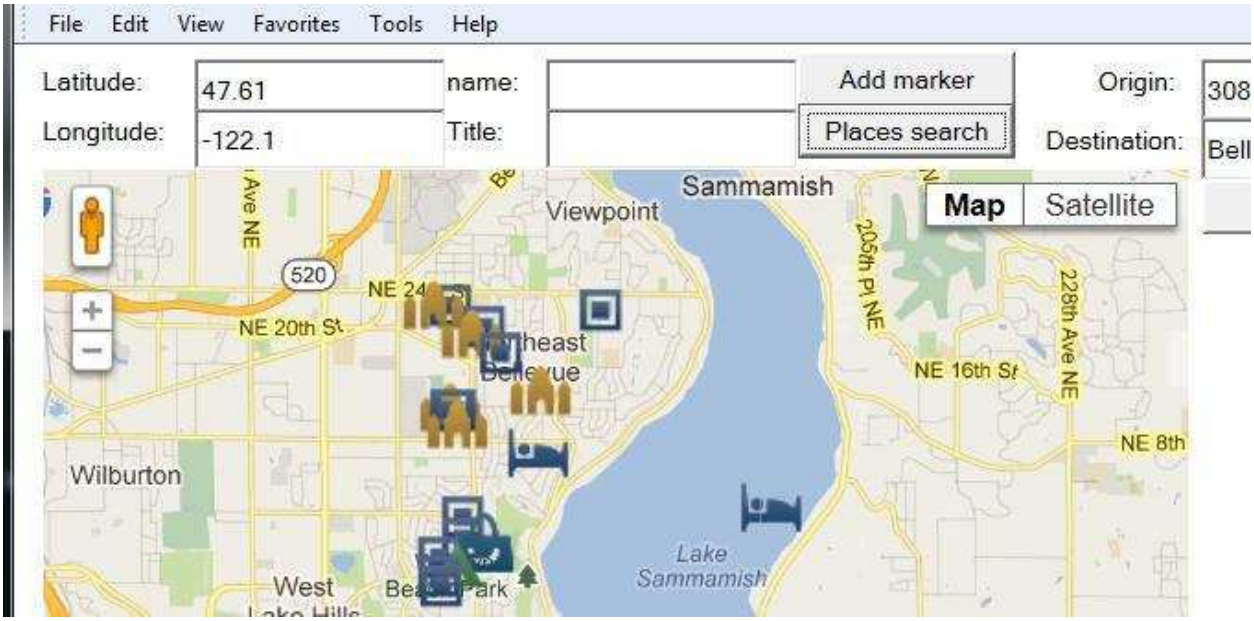
We can test the web page now:



The web page appears. Click “Places search”:



Found places appear on the map:



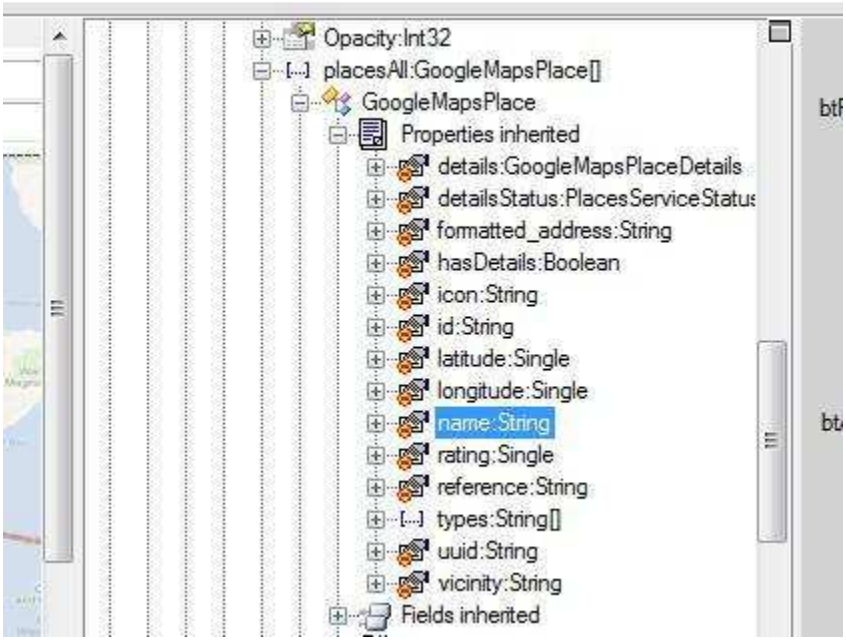
Note that these places are from the first page. We need to use a `GetNextPlacesPage` action to get the next page. We will do it later.

Handle PlaceMarkerClick Event

We have seen that a marker appears on the map to represent one place. Clicking a marker, event `PlaceMarkerClick` occurs.

Suppose we want to display an information window when event `PlaceMarkerClick` occurs.

We can find information about a place from Object Explorer:



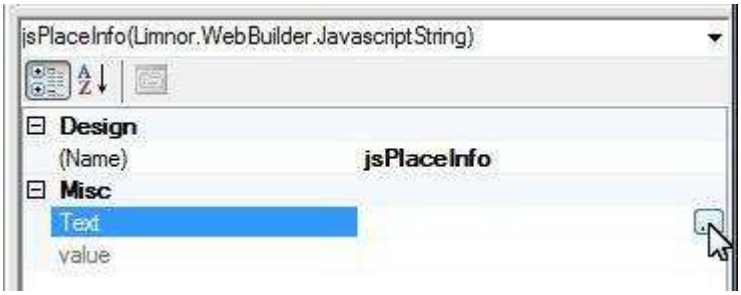
Suppose we want to display latitude, longitude, name, rating, and vicinity.

An information window may display HTML contents. In previous examples, we have used expressions to form text for information windows. Here we introduce another way to form HTML text for information windows.

We may use a JavascriptString component to visually form HTML contents which contains fields for latitude, longitude, name, rating, and vicinity. When a place marker is clicked, values for latitude, longitude, name, rating, and vicinity are substituted by the place.

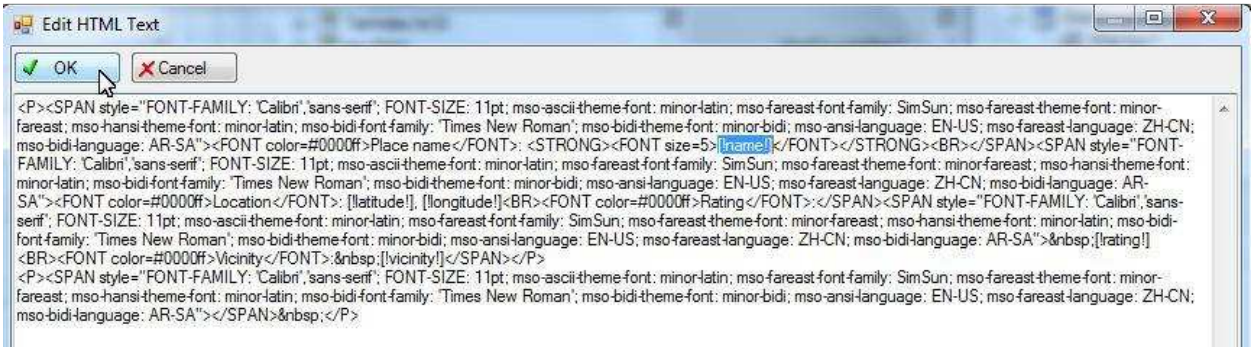
Drop a JavascriptString to the web page:

Set its Text property to create HTML contents:



Each field is marked by [! and !]. A field name cannot contain spaces. The first character of a field name must be a letter or an underscore. The other characters can be letters, number, or underscores.

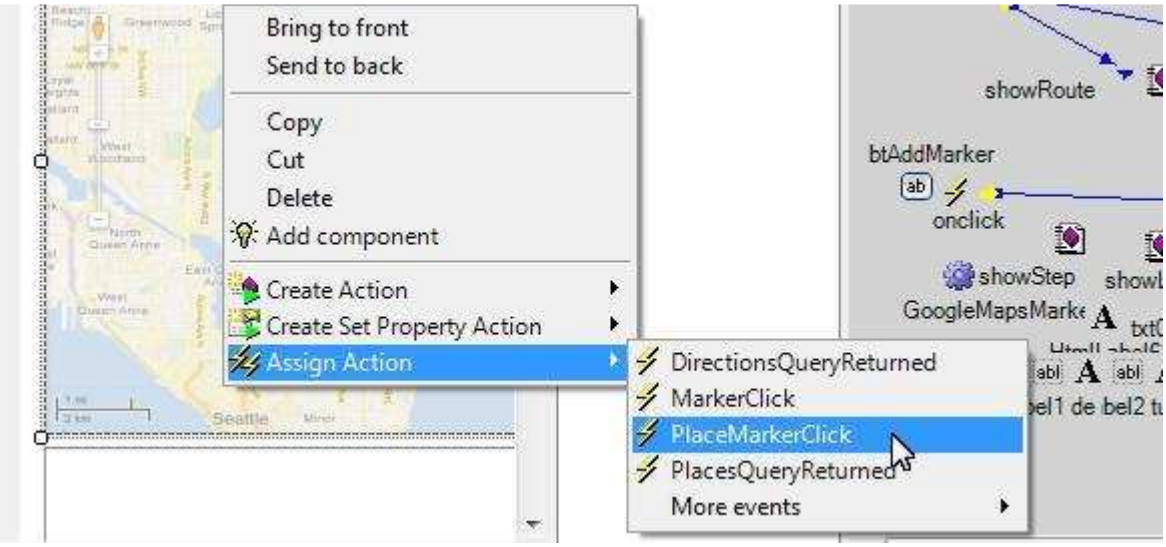
Note that HTML tags might get into symbols, ” [“,”]” and “!”, without your notice because HTML tags are invisible. For example, HTML code [!name!] looks like [!name!], because and are invisible, but they damage field marking. You may click “Edit HTML” to see HTML contents, and fix field markings if needed:



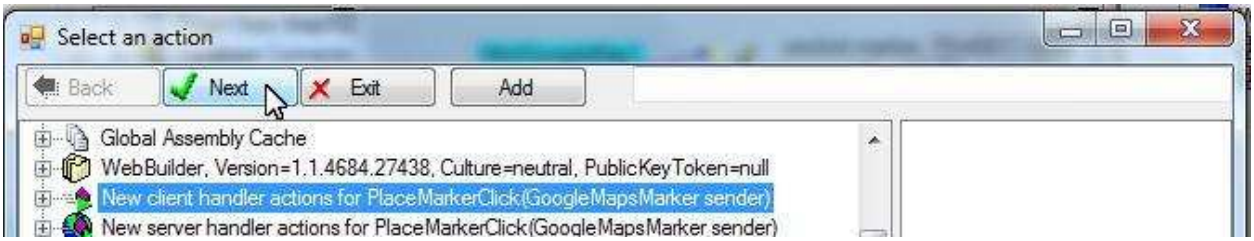
Each field will become a property:

If a field you want to create does not appear as a property then there are HTML tags getting into its field markings. You may use “Edit HTML” to move those HTML tags out of [! and !].

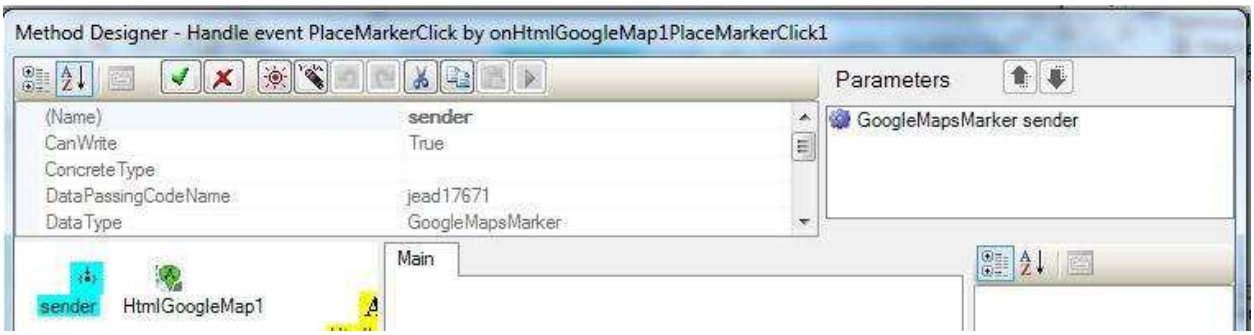
Now let’s handle event `PlaceMarkerClick` to show an information window:



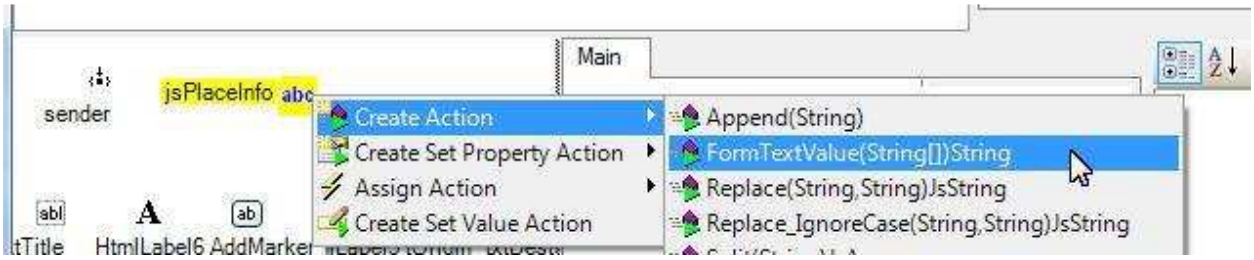
Select “New client handler actions for PlaceMarkerClick”:



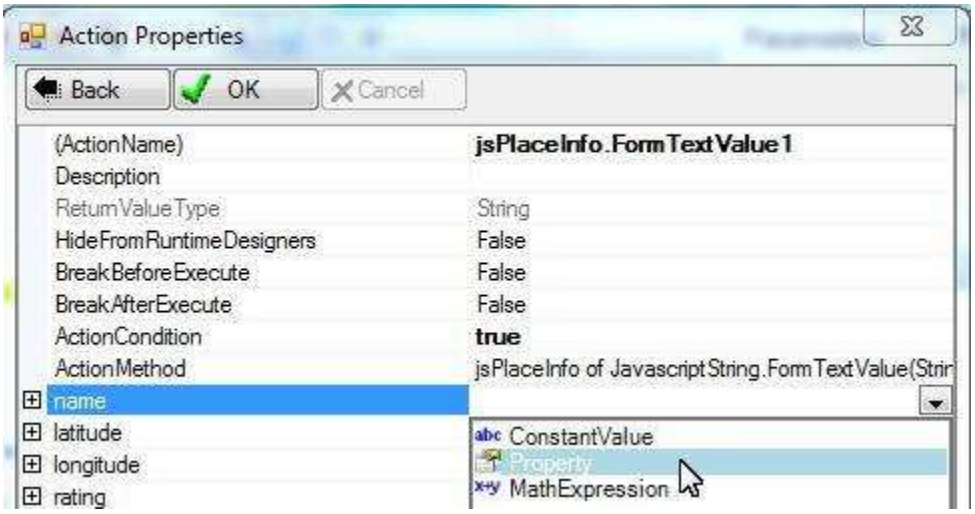
A method editor appears. Note that “sender” is the marker being clicked.

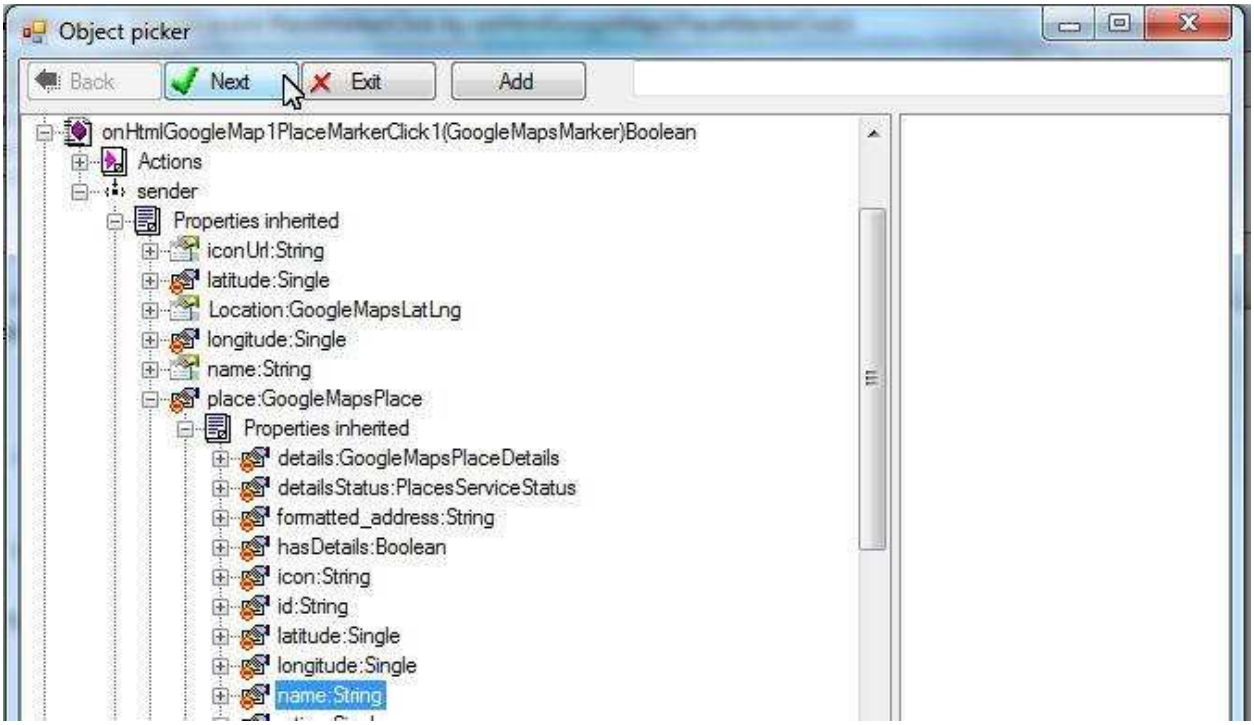


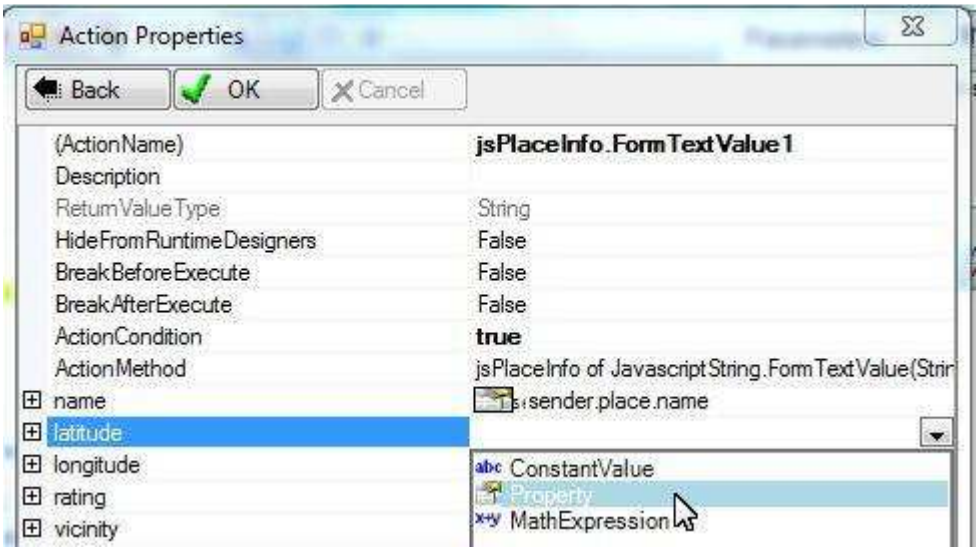
Let’s use JavascriptString to form display information:



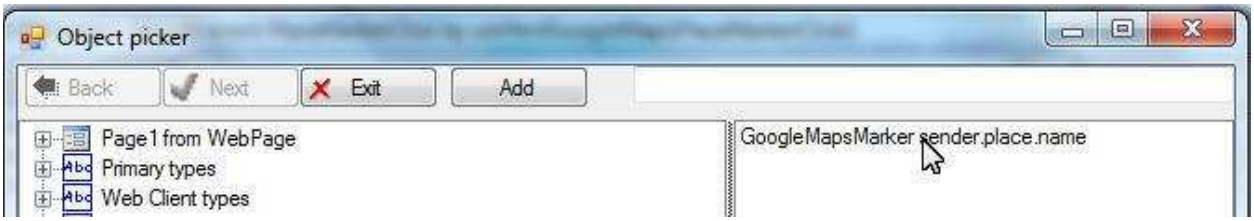
Pass properties of the place to the action:

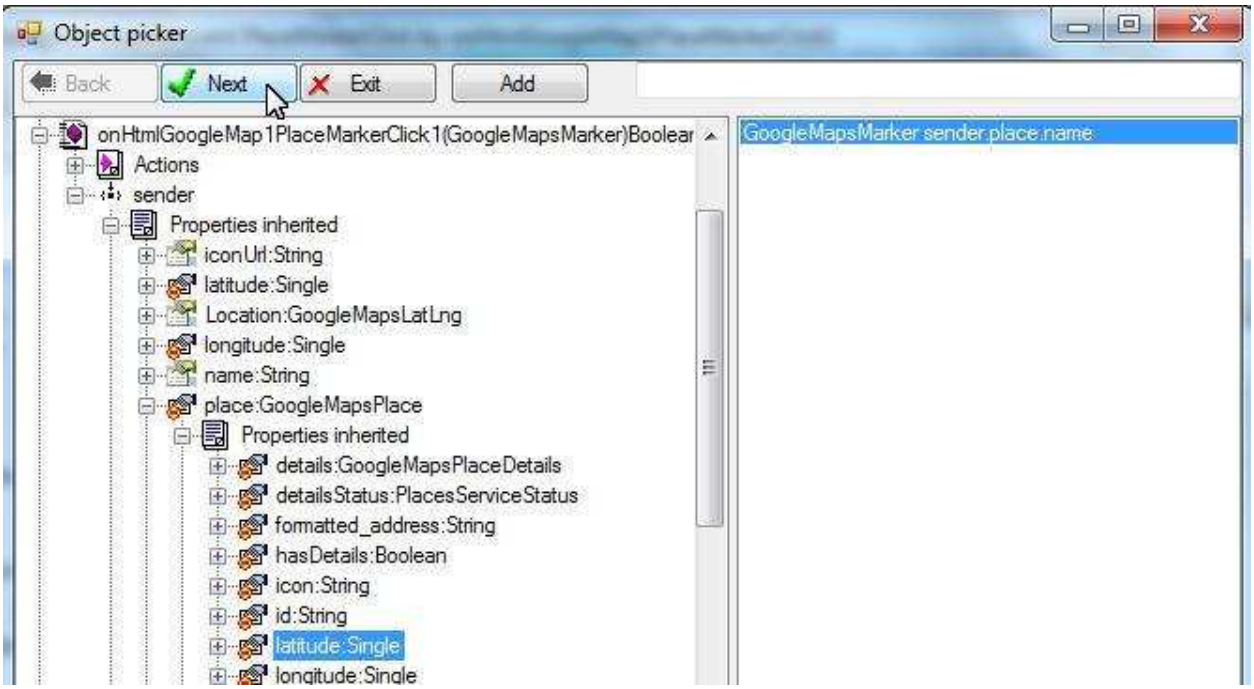




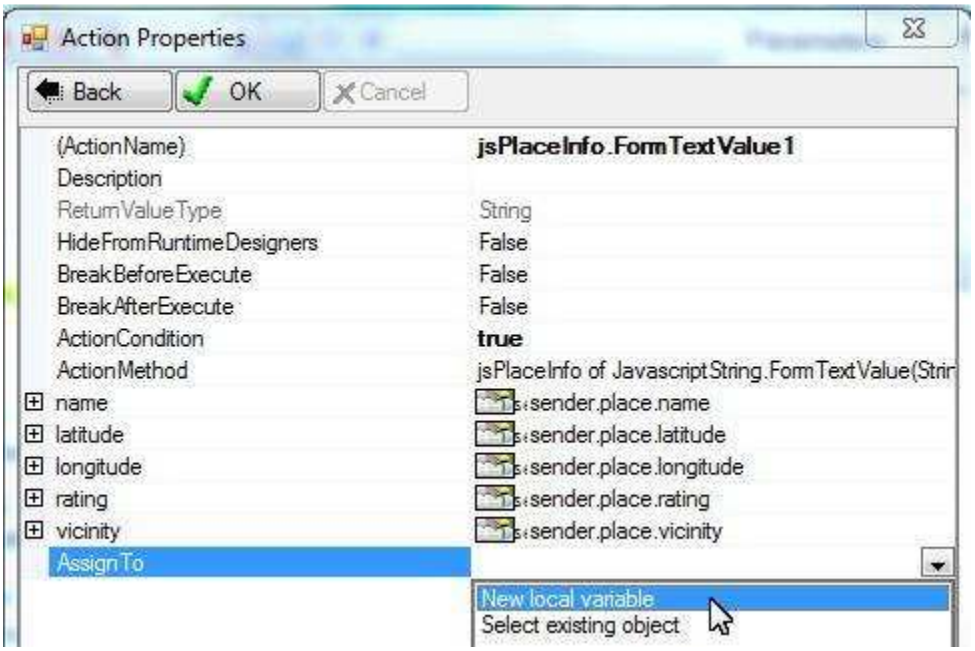


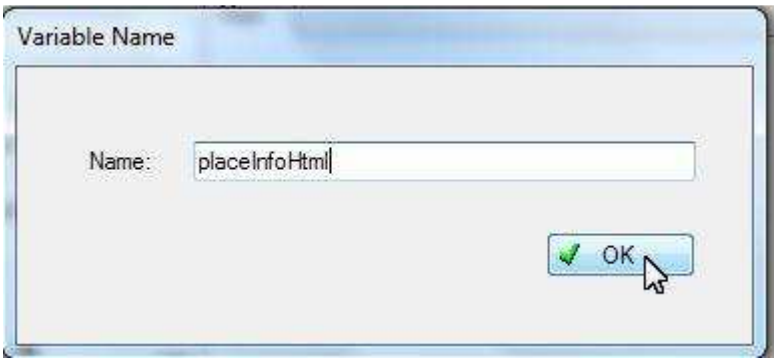
Click a recent selection to help quickly locate a property we want to select:

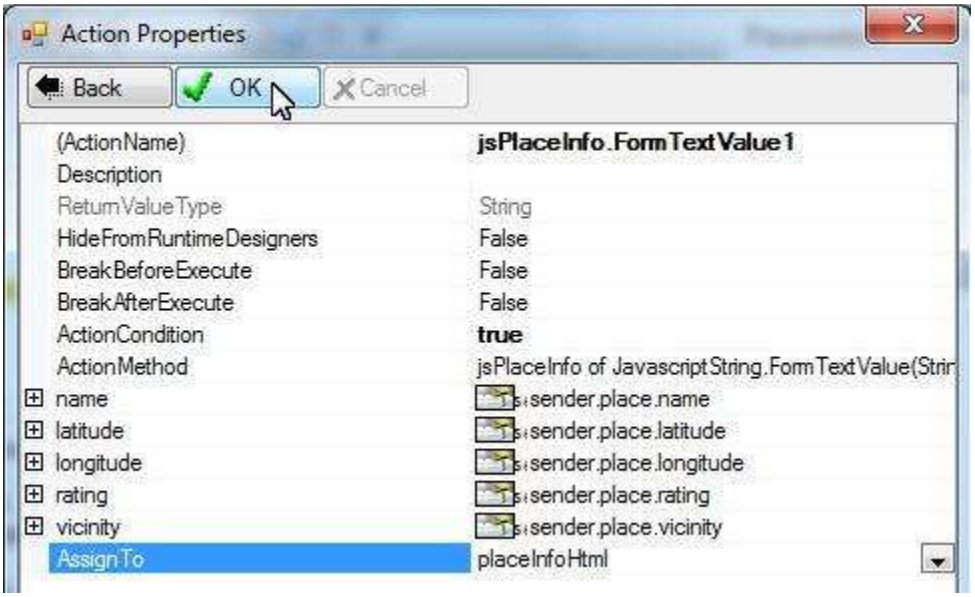




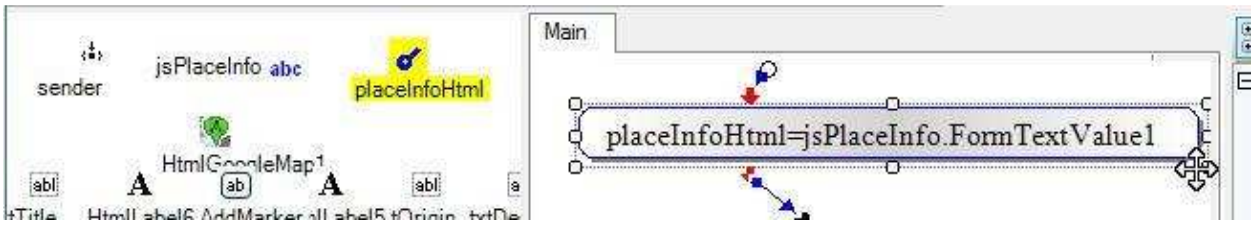
Use a new variable for the information it forms:


Action Properties
Back
OK
Cancel
(ActionName)
jsPlaceInfo.FormTextValue1
Description
ReturnValueType
String
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
jsPlaceInfo of JavascriptString.FormTextValue(Strir
name
sender.place.name
latitude
sender.place.latitude
longitude
sender.place.longitude
rating
sender.place.rating
vicinity
sender.place.vicinity
AssignTo
New local variable
Select existing object

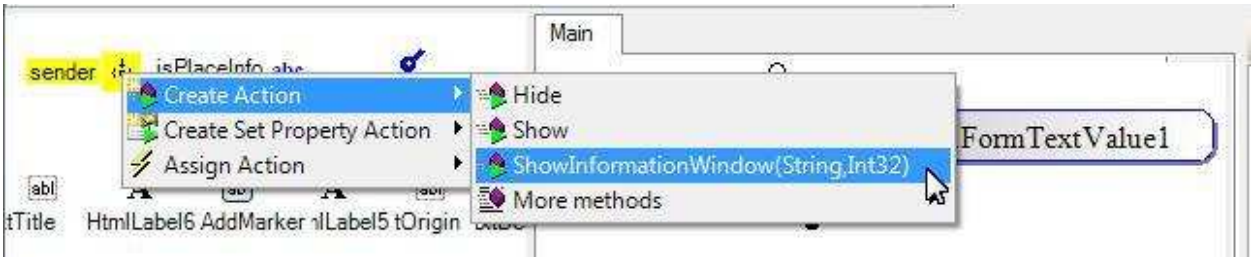

Variable Name
Name:
placeInfoHtml
OK


Action Properties
Back
OK
Cancel
(ActionName)
jsPlaceInfo.FormTextValue1
Description
ReturnValueType
String
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
jsPlaceInfo of JavascriptString.FormTextValue(Strir
name
sender.place.name
latitude
sender.place.latitude
longitude
sender.place.longitude
rating
sender.place.rating
vicinity
sender.place.vicinity
AssignTo
placeInfoHtml

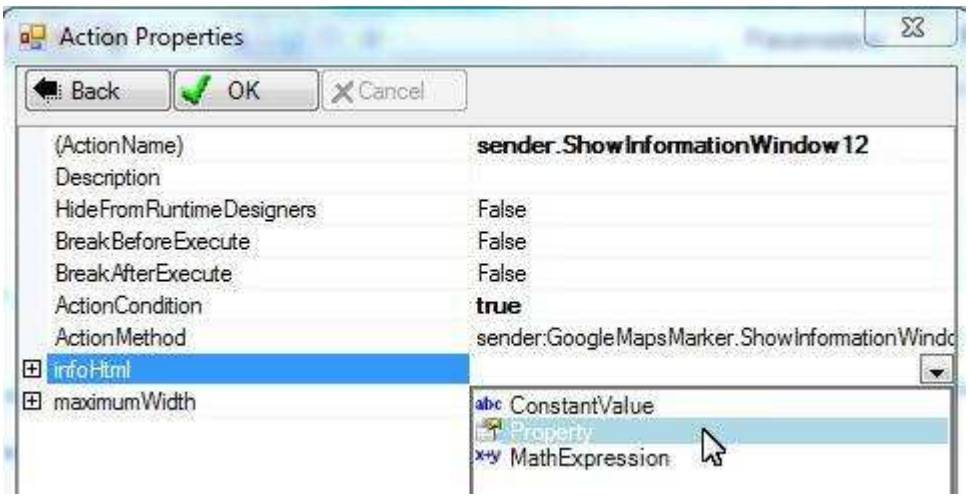
The action and variable appear:

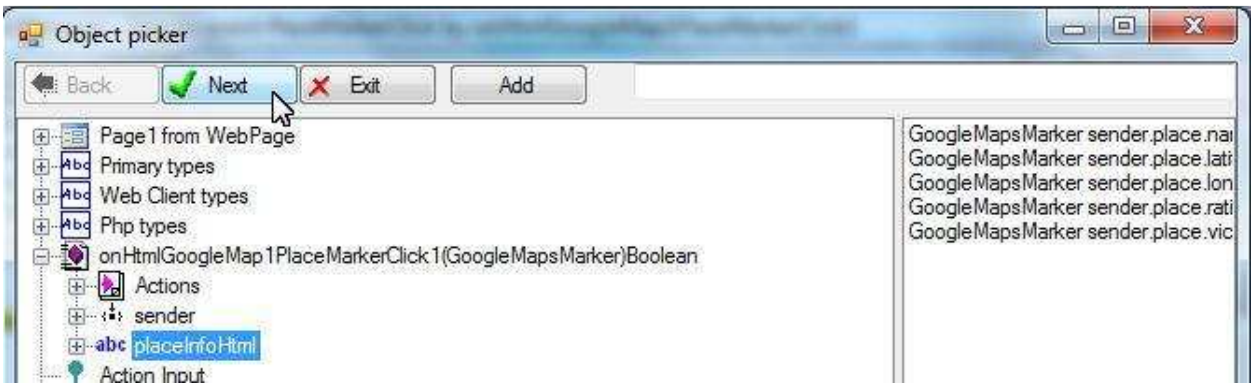


Create an action to show information window:

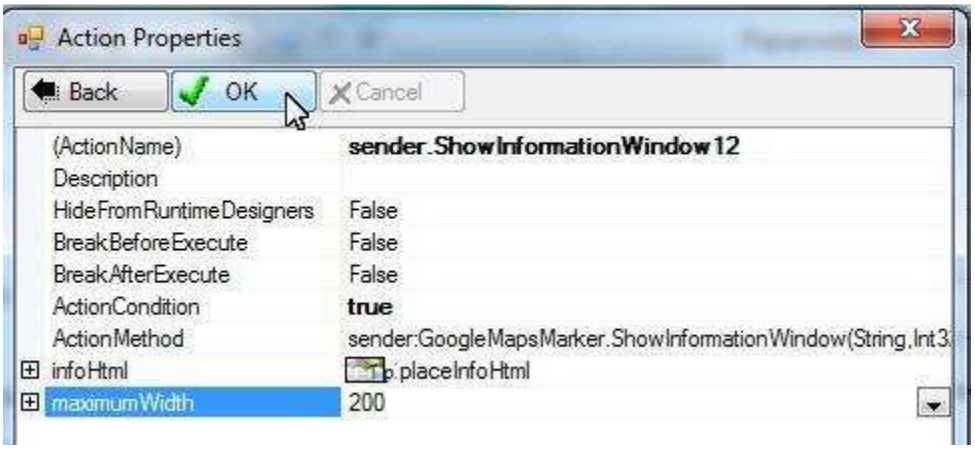


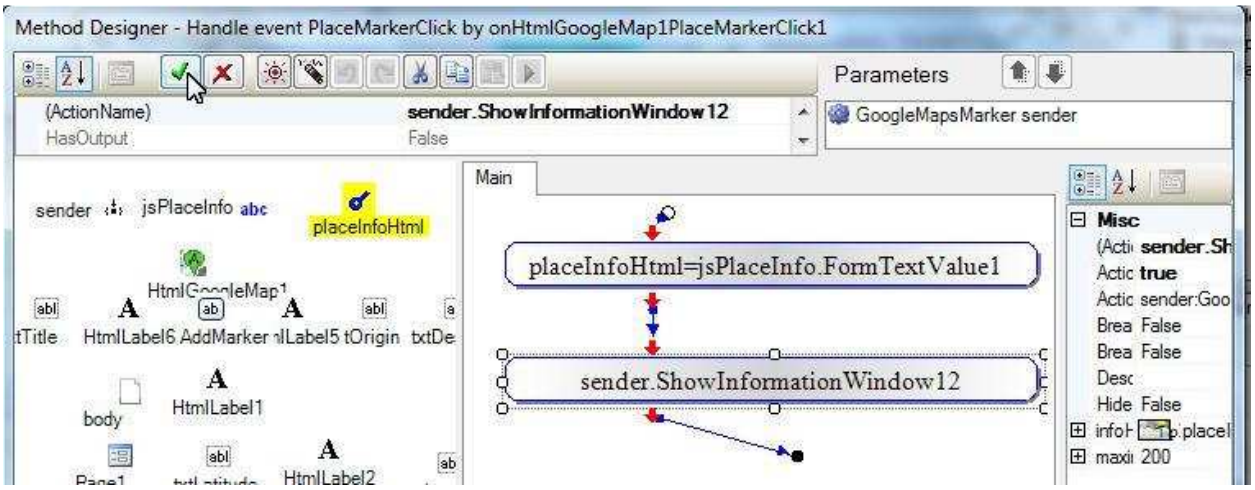
Use the information we just formed:





Set maximum width and click OK:





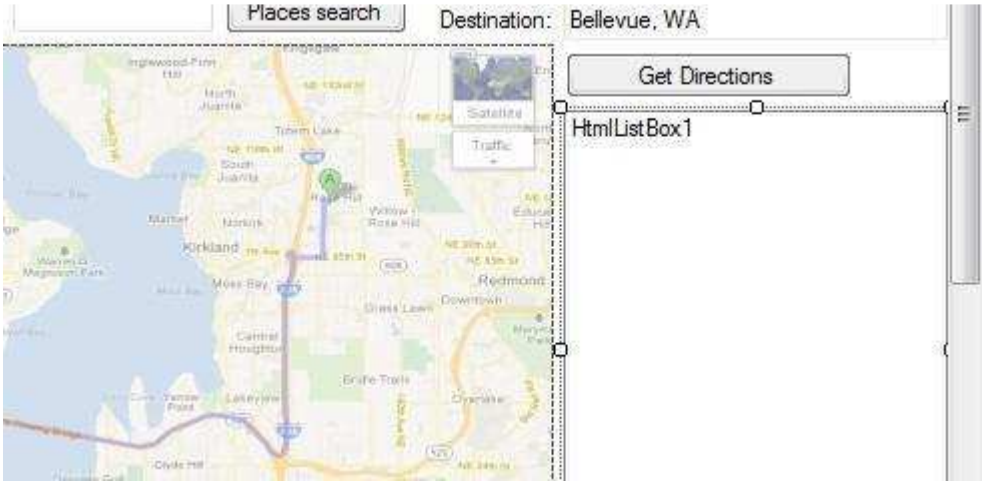
We may test it now. Launch the web page and click “Places search”. Places appear on the map. Click on a place:

Information window appears:

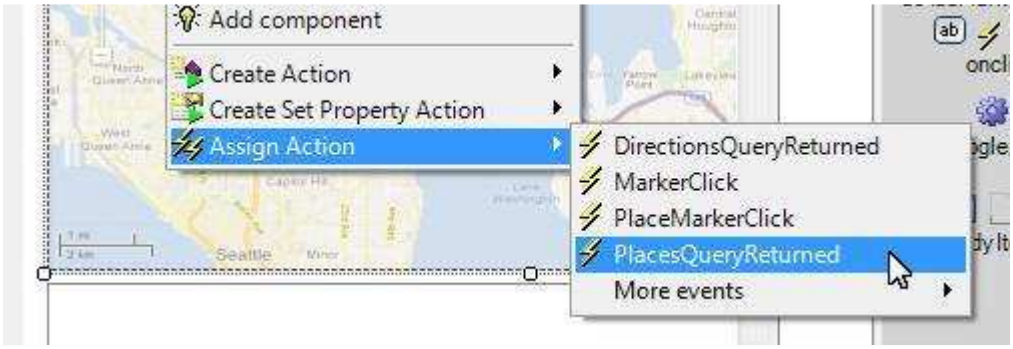
List places

Place objects are stored in array `placesOnLastPage` and `placesAll`. You may use them if needed. For example, save them to a database. Here we list them in a list box.

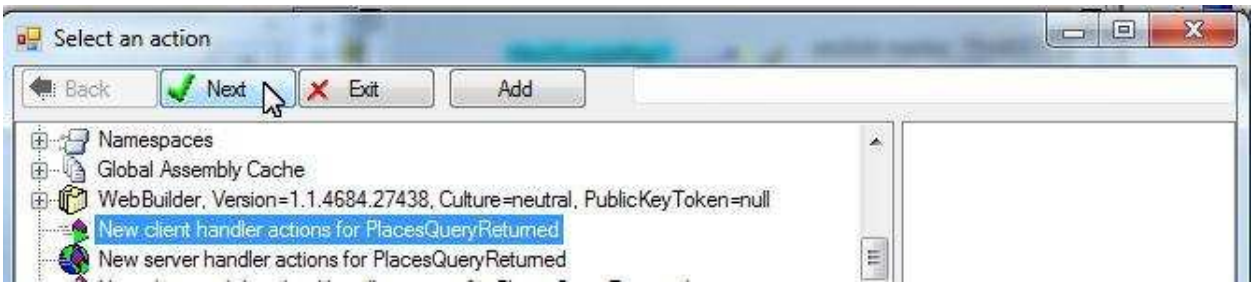
Add a list box to the web page:



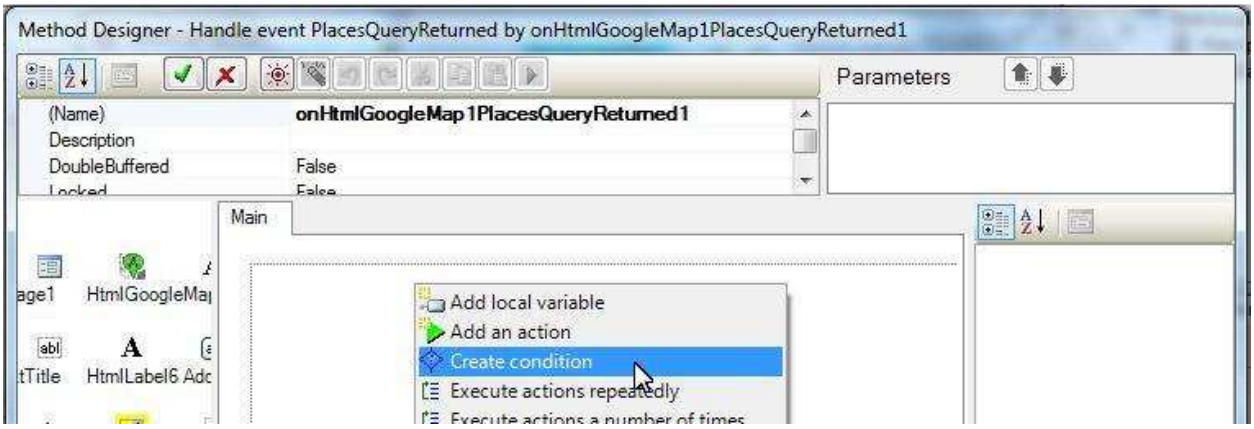
We'll handle event `PlacesQueryReturned` to add places to the list box:

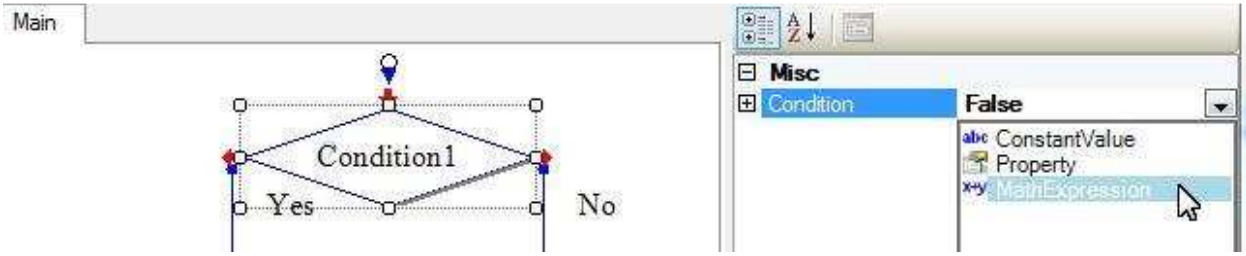


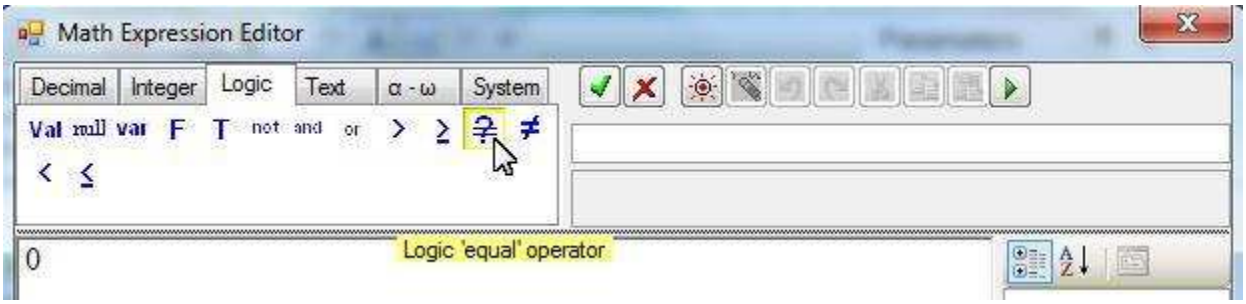
Select "New client handler actions for PlacesQueryReturned":



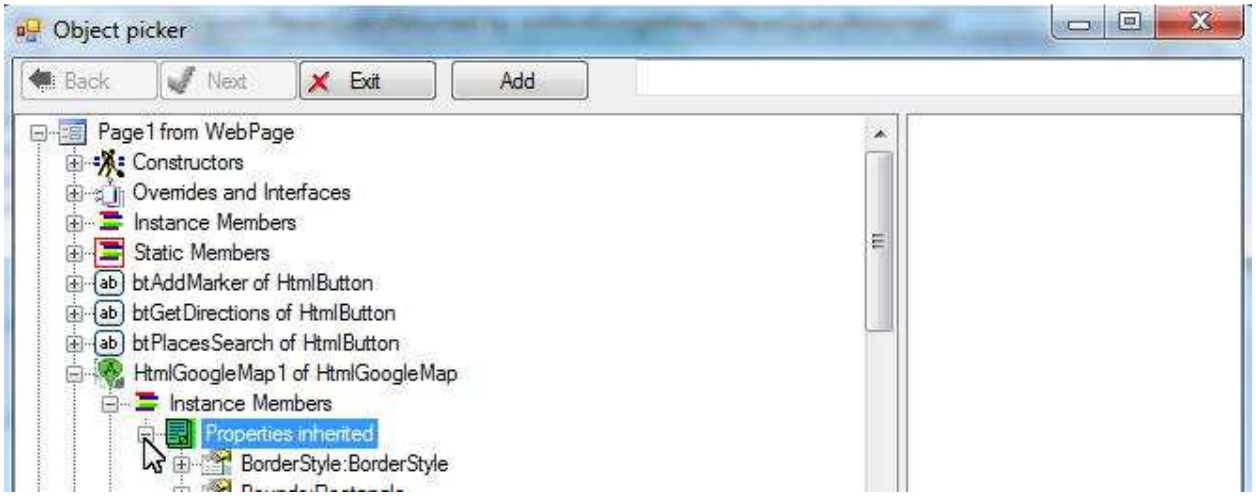
Add a condition action to check places search status:

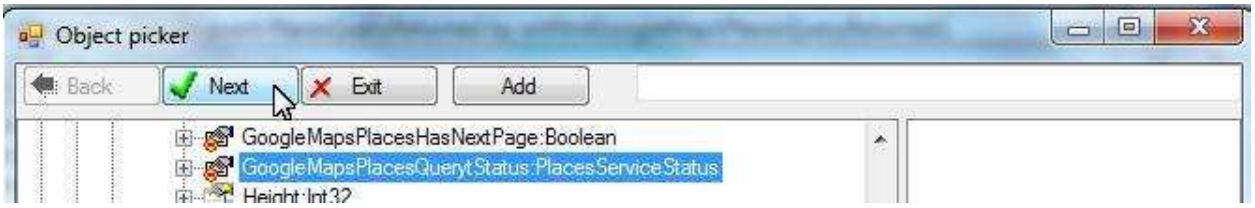

Method Designer - Handle event PlacesQueryReturned by onHtmlGoogleMap1PlacesQueryReturned1
Parameters
(Name)
onHtmlGoogleMap1PlacesQueryReturned1
Description
DoubleBuffered
False
Main
Add local variable
Add an action
Create condition
Execute actions repeatedly


Main
Condition1
Yes
No
Misc
Condition
False
ConstantValue
Property
MathExpression

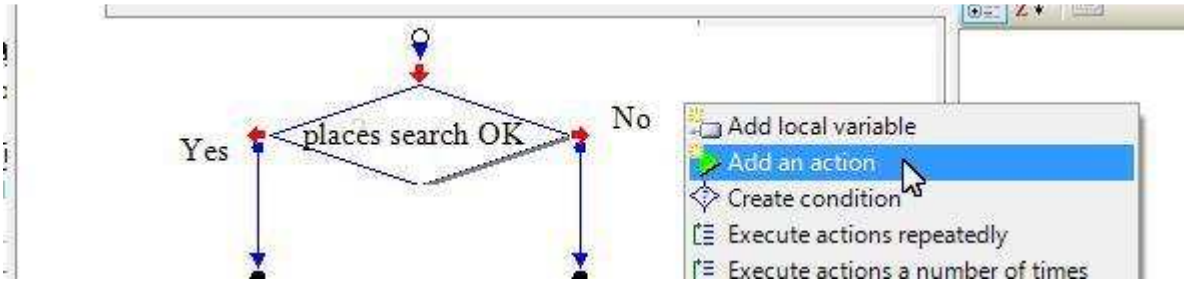

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
var
F
T
not
and
or
Logic 'equal' operator
0

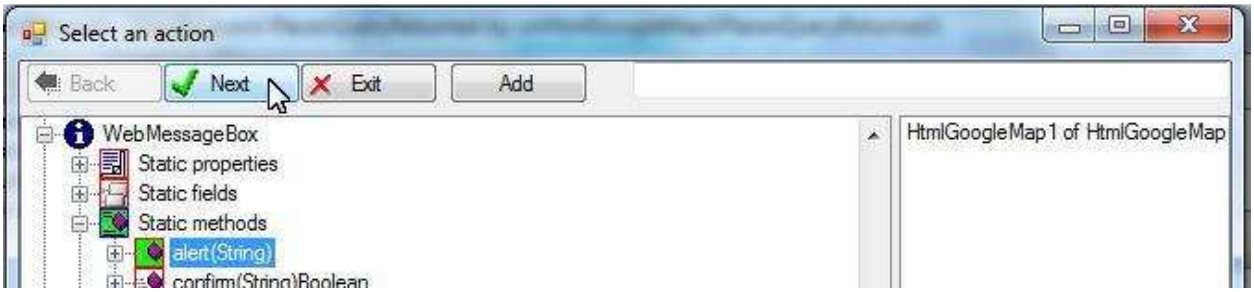

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(Boolean)
DataType
TRUE
TRUE
A property

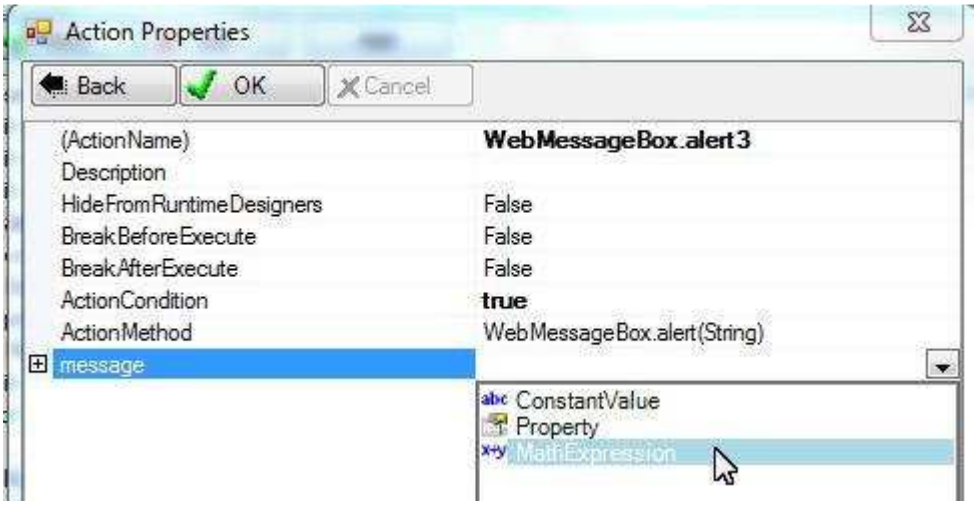


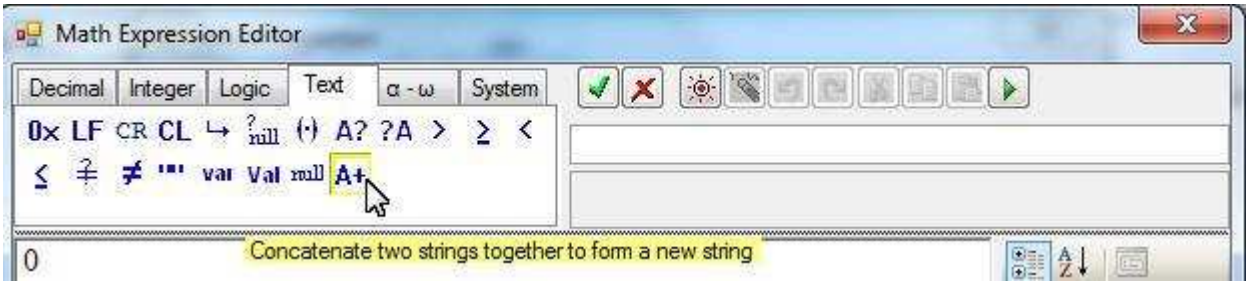


Add a message box action if the status is not OK:

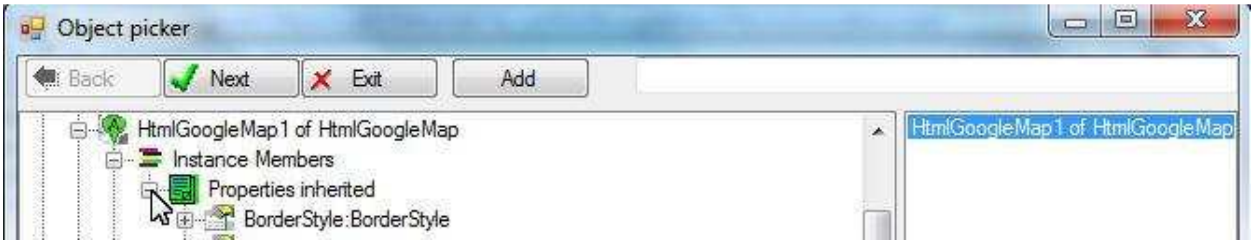

Yes
places search OK
No
Add local variable
Add an action
Create condition
Execute actions repeatedly
Execute actions a number of times

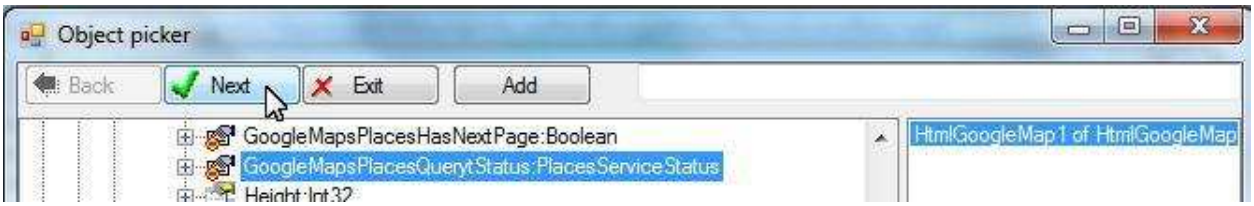

Select an action
Back
Next
Exit
Add
WebMessageBox
Static properties
Static fields
Static methods
alert(String)
confirm(String)Boolean
HtmlGoogleMap1 of HtmlGoogleMap


Action Properties
Back
OK
Cancel
(ActionName)
WebMessageBox.alert3
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
WebMessageBox.alert(String)
message
ConstantValue
Property
MathExpression

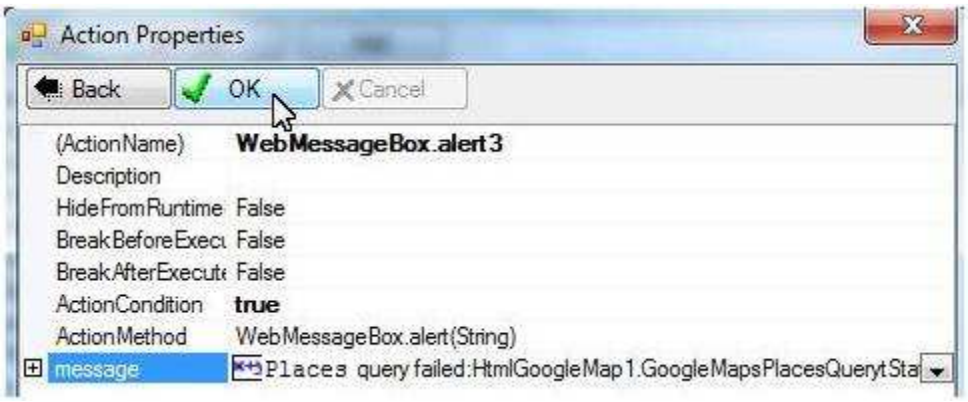

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Concatenate two strings together to form a new string

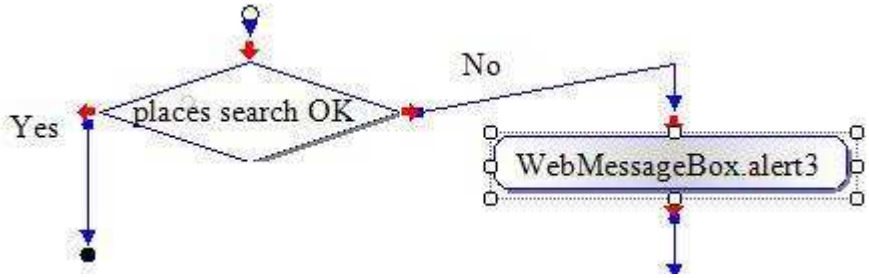

Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
null
DataType
(String)
DataType
"Places query failed:"
A property


Object picker
Back
Next
Exit
Add
HtmlGoogleMap1 of HtmlGoogleMap
Instance Members
Properties inherited
BorderStyle:BorderStyle
HtmlGoogleMap1 of HtmlGoogleMap

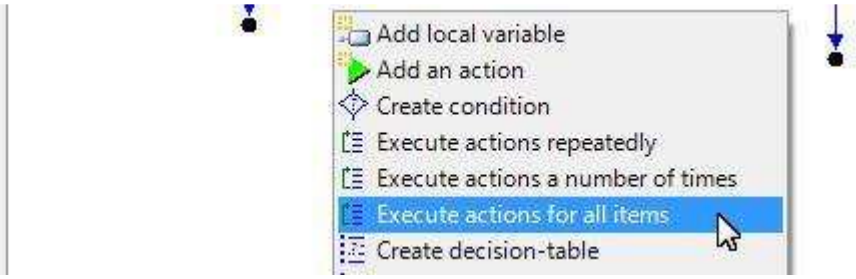

Object picker
Back
Next
Exit
Add
GoogleMapsPlacesHasNextPage:Boolean
GoogleMapsPlacesQueryStatus:PlacesServiceStatus
HtmlGoogleMap1 of HtmlGoogleMap

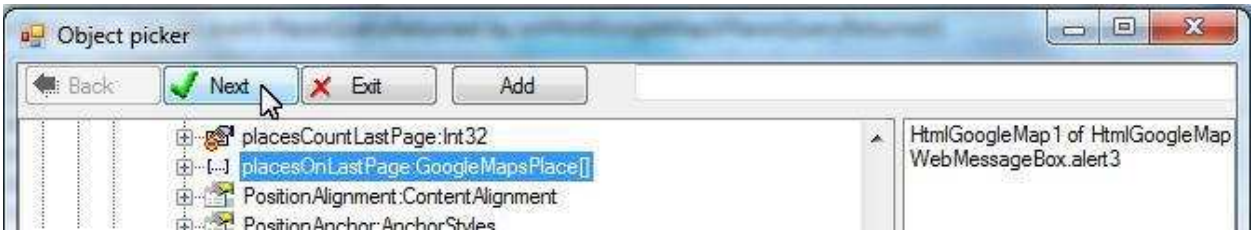

Math Expression Editor
Finish the editing.
Decimal
Integer
Logic
Text
α - ω
System
Val
null
RunAt
Client
TaskId
0
"Places query failed:"
HtmlGoogleMap1.GoogleMapsPlacesQuerytStatus


Action Properties
Back
OK
Cancel
(ActionName)
WebMessageBox.alert3
Description
HideFromRuntime
False
BreakBeforeExecu
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
WebMessageBox.alert(String)
message
Places query failed:HtmlGoogleMap1.GoogleMapsPlacesQuerytSta

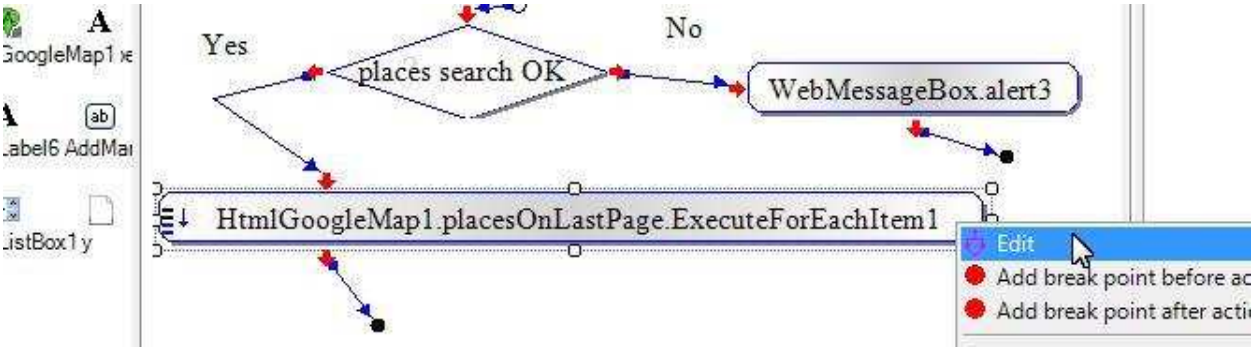


Create an "Execute actions for all items" action for `placesOnLastPage`:

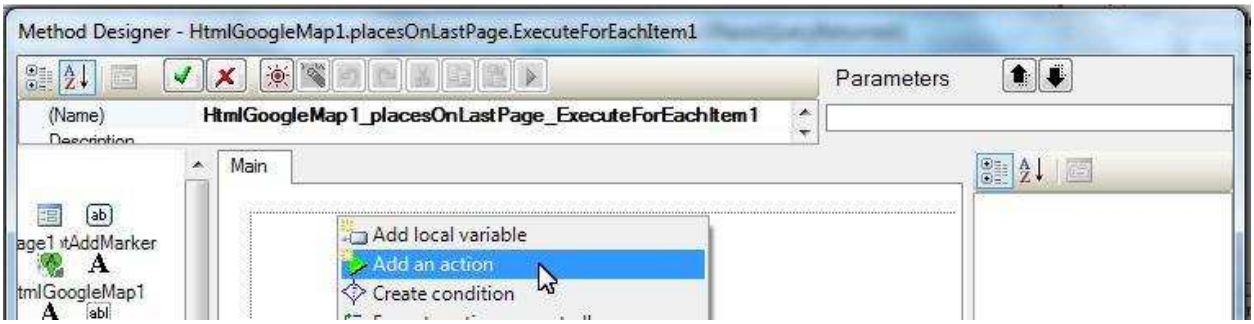


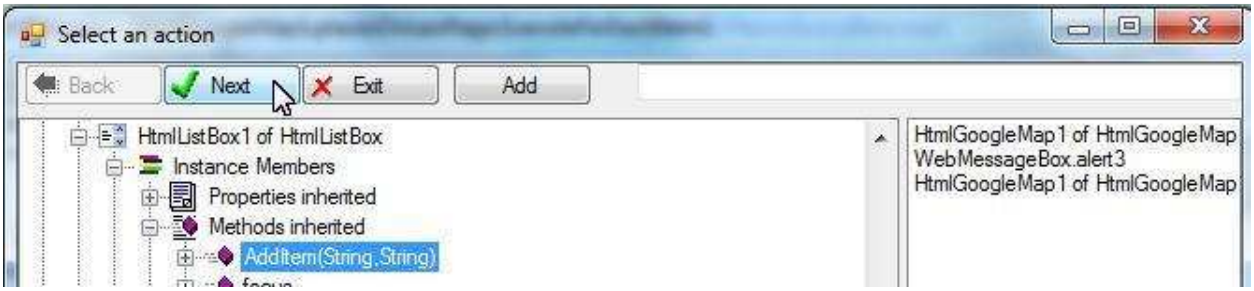


Edit the action to process each place:

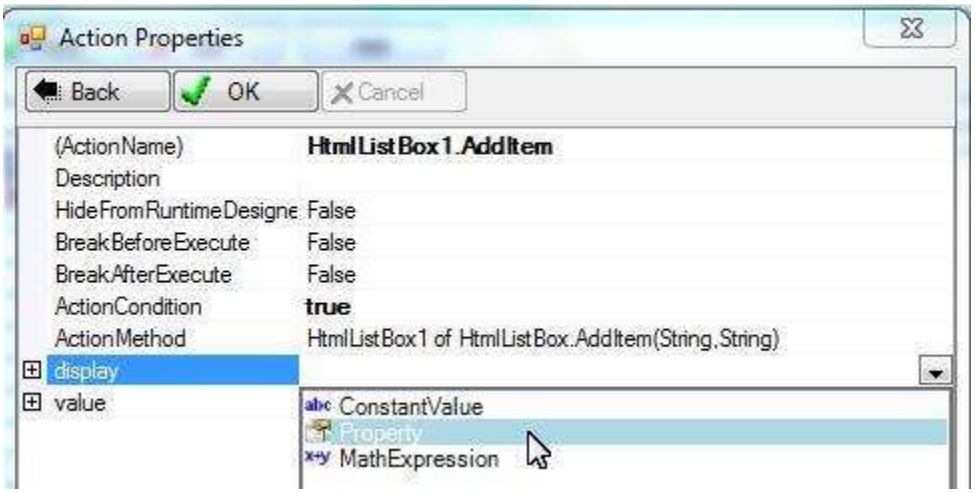


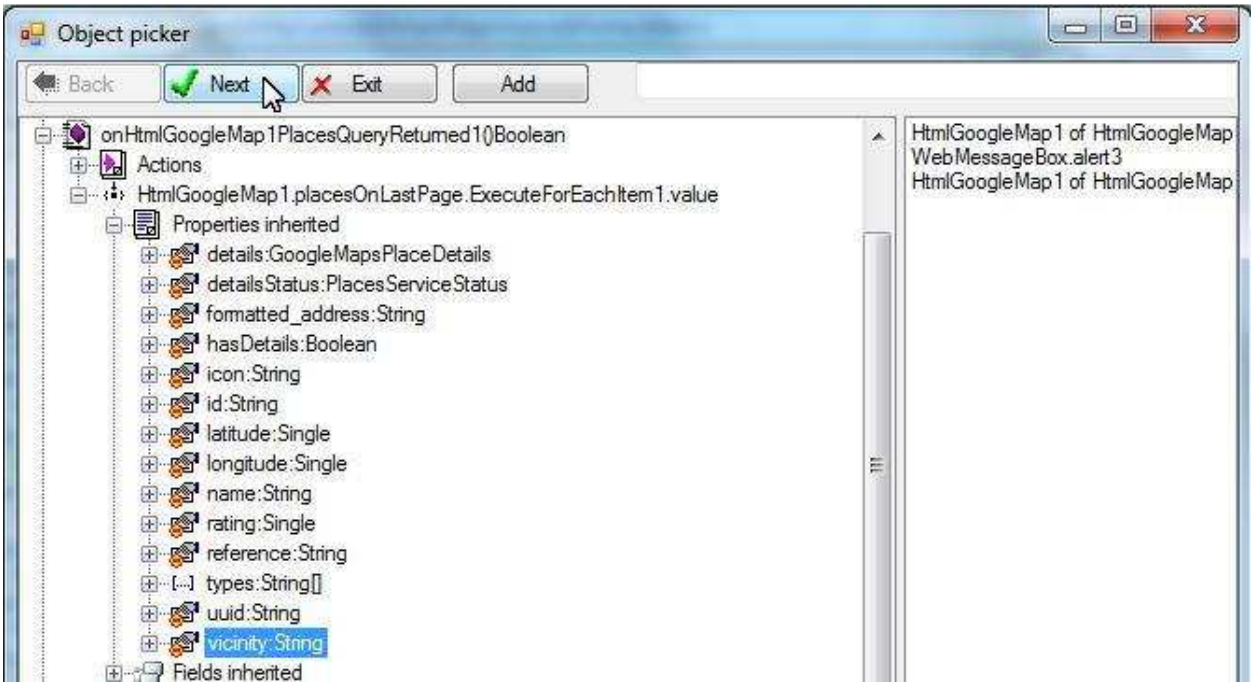
A method editor appears. Create an action to add a list item:



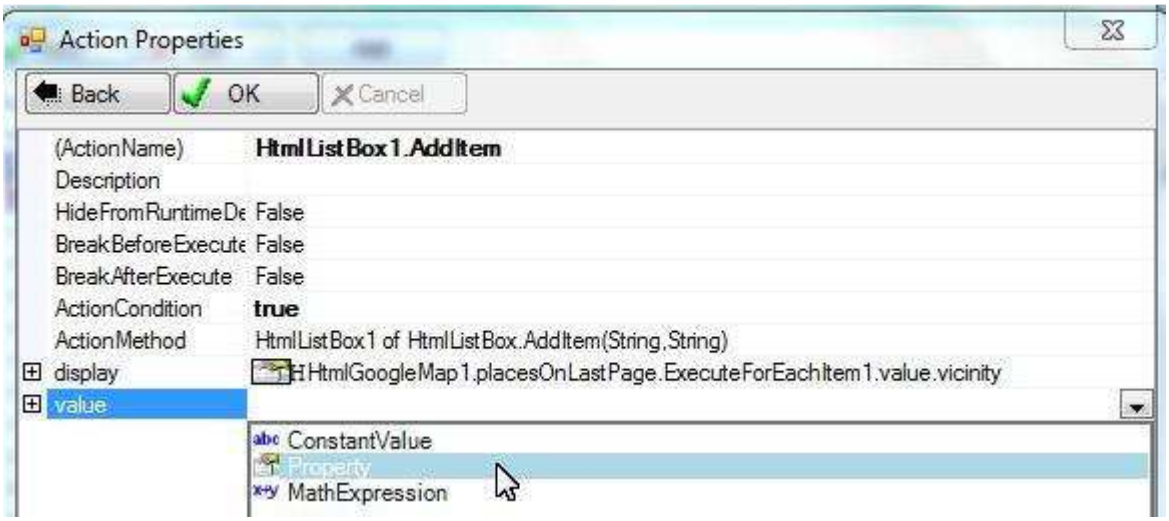


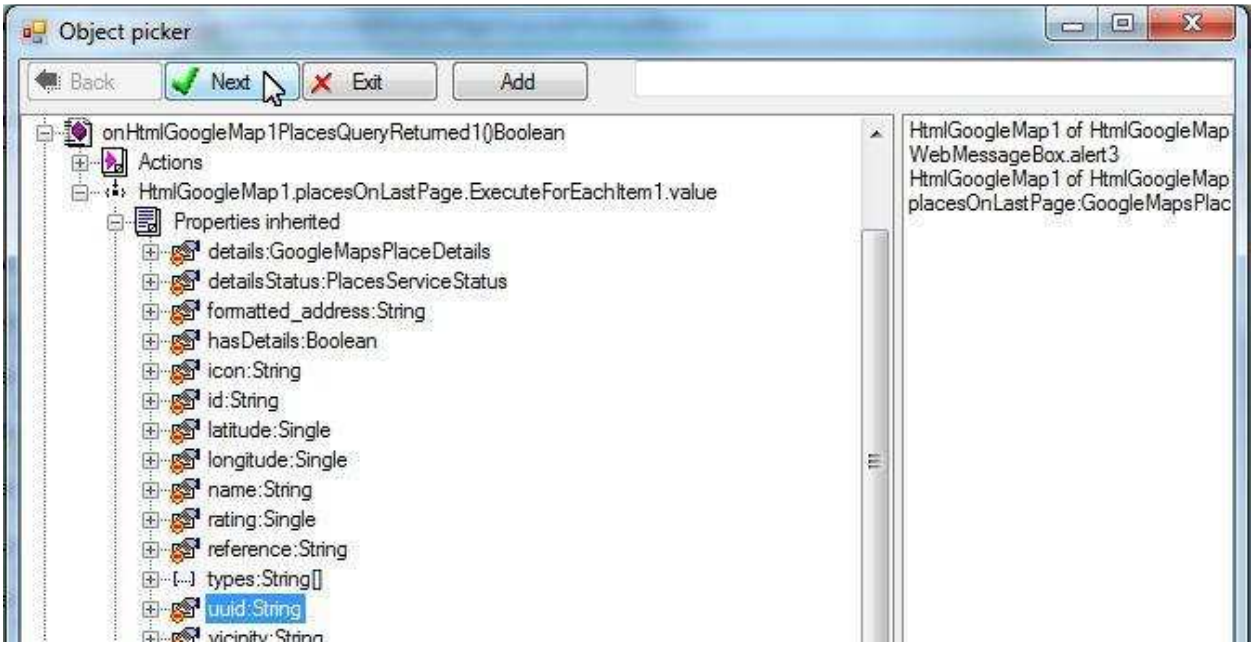
Display vicinity on the list box:

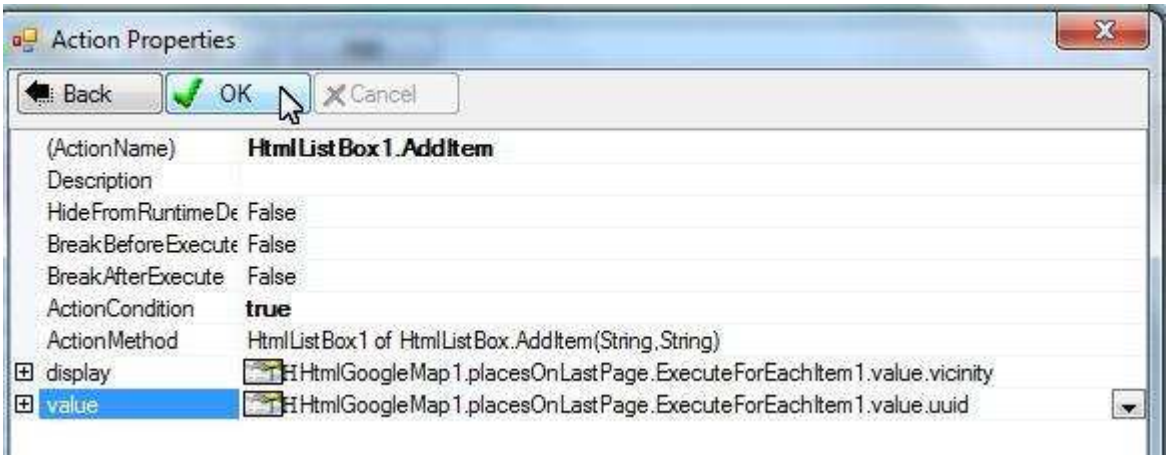


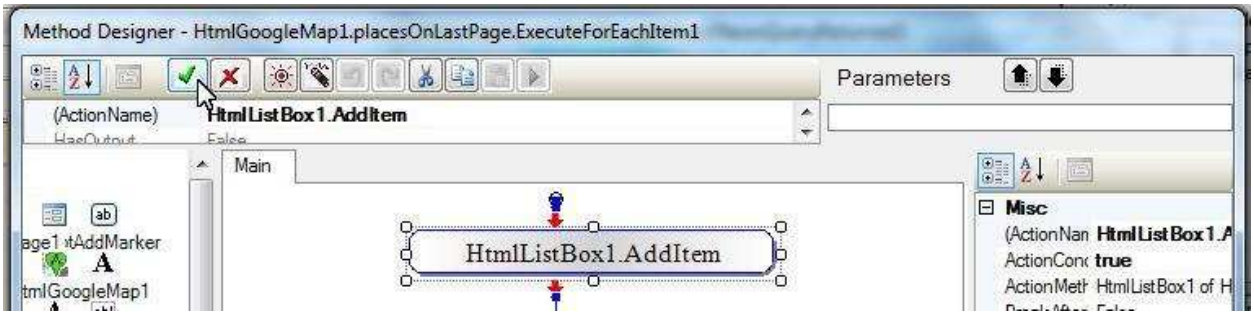


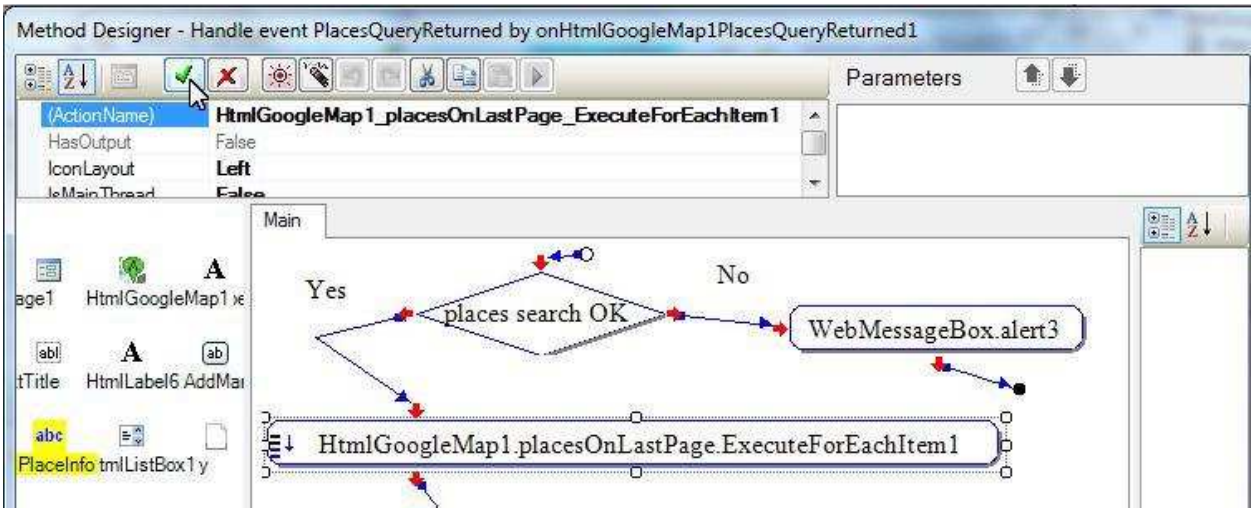
Each place has a uuid property to uniquely identifying the place among a web page. This uuid has the same value as the uuid for the marker of the place. We may use it to locate the place or marker later. So, let's set it to the value of the list box item:




Object picker
Back
Next
Exit
Add
onHtmlGoogleMap1PlacesQueryReturned1()Boolean
Actions
HtmlGoogleMap1.placesOnLastPage.ExecuteForEachItem1.value
Properties inherited
details:GoogleMapsPlaceDetails
detailsStatus:PlacesServiceStatus
formatted_address:String
hasDetails:Boolean
icon:String
id:String
latitude:Single
longitude:Single
name:String
rating:Single
reference:String
types:String[]
uuid:String
HtmlGoogleMap1 of HtmlGoogleMap
WebMessageBox.alert3
HtmlGoogleMap1 of HtmlGoogleMap
placesOnLastPage:GoogleMapsPlac

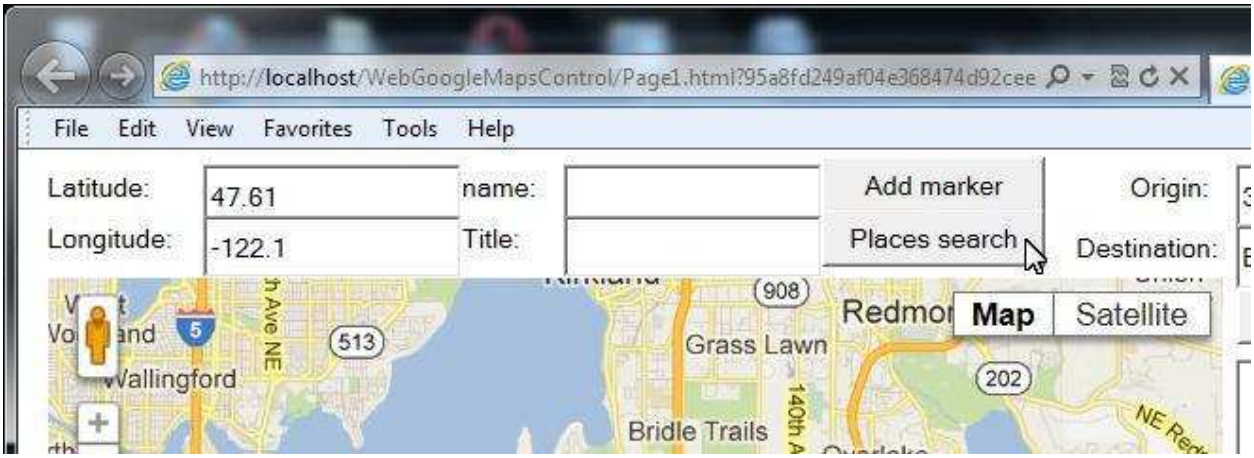

Action Properties
Back
OK
Cancel
(ActionName) HtmlListBox1.AddItem
Description
HideFromRuntimeDe False
BreakBeforeExecute False
BreakAfterExecute False
ActionCondition true
ActionMethod HtmlListBox1 of HtmlListBox.AddItem(String,String)
display HtmlGoogleMap1.placesOnLastPage.ExecuteForEachItem1.value.vicinity
value HtmlGoogleMap1.placesOnLastPage.ExecuteForEachItem1.value.uuid


Method Designer - HtmlGoogleMap1.placesOnLastPage.ExecuteForEachItem1
Parameters
(ActionName) HtmlListBox1.AddItem
Main
HtmlListBox1.AddItem
Misc
ActionCon true
ActionMeth HtmlListBox1 of H



The event handler method is created:

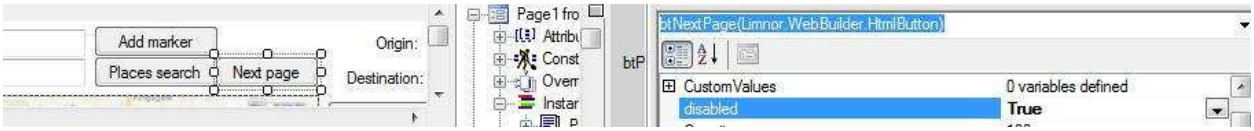
We may test it now. Launch the web page. Click “Places search”:

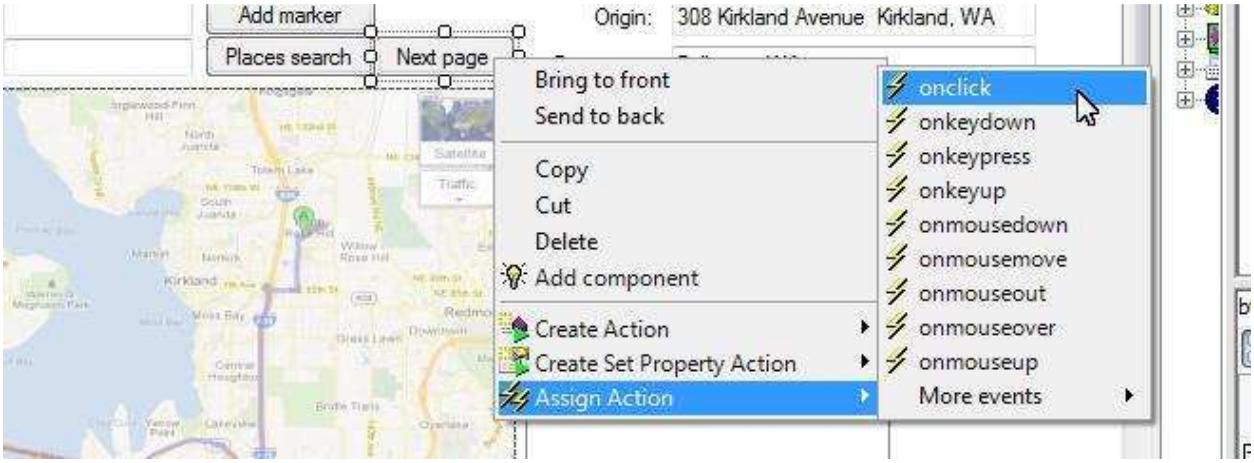


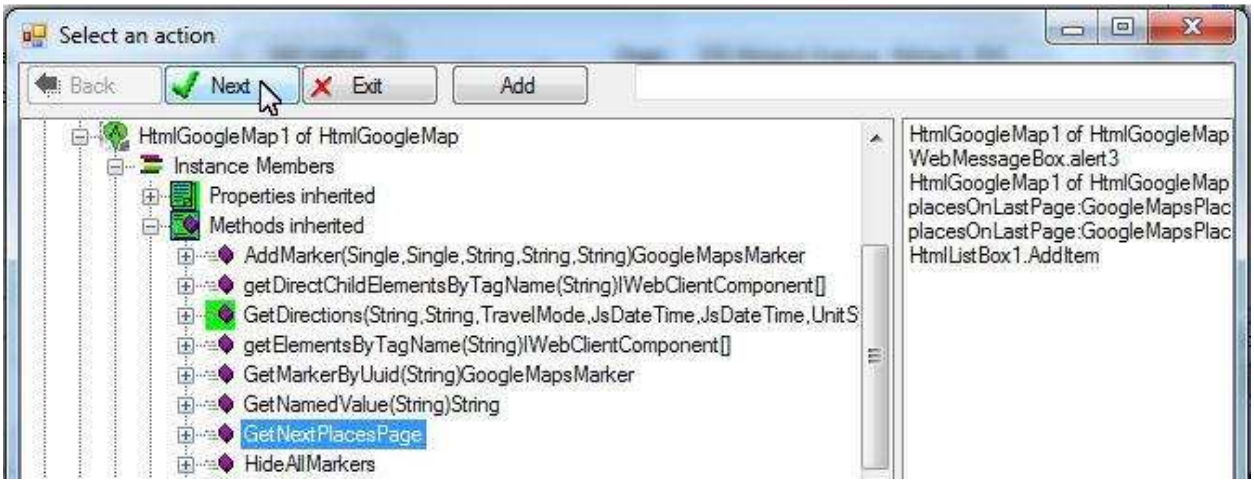
Places appear in the list box:

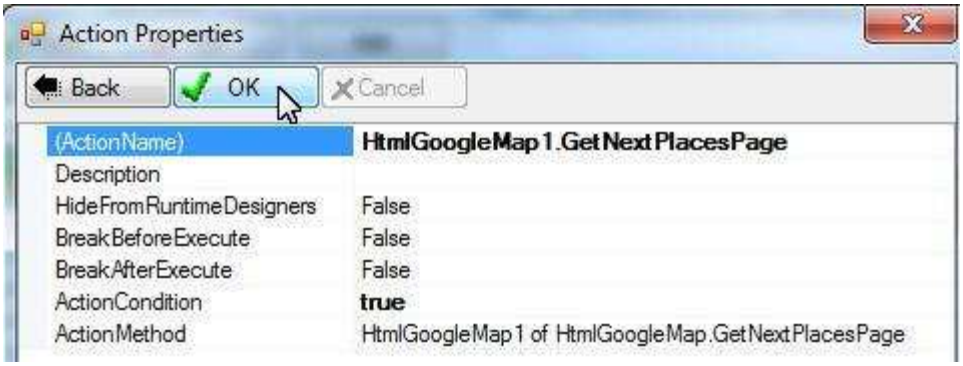
Get next page of places

If property `GoogleMapsPlacesHasNextPage` is True then we may execute `GetNextPlacesPage` to get places of the next page. We use a button to execute `GetNextPlacesPage`. Initially the button is disabled. At event `PlacesQueryReturned` we enable/disable the button according to property `GoogleMapsPlacesHasNextPage`.



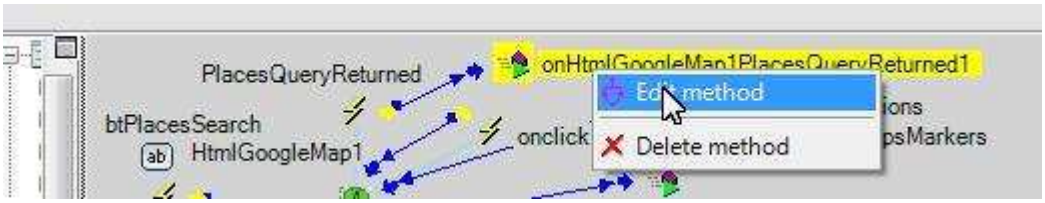


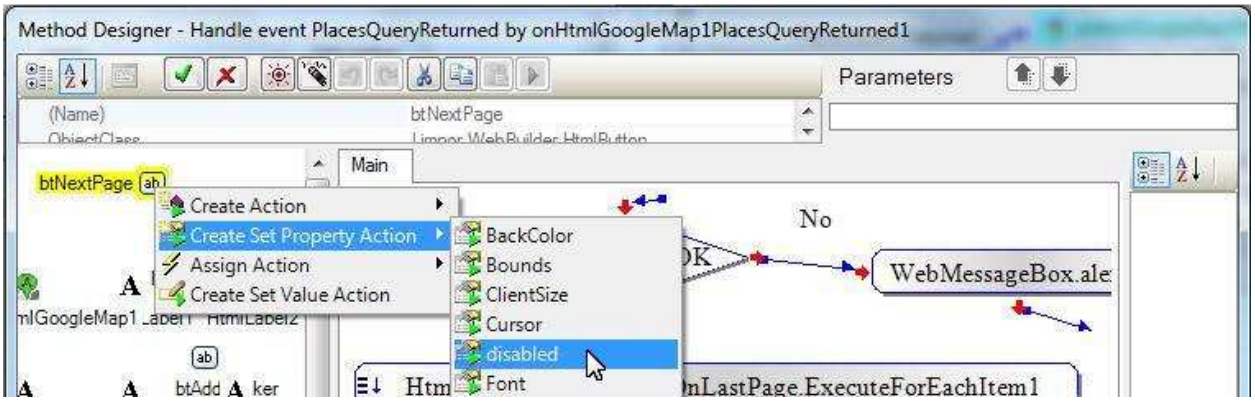


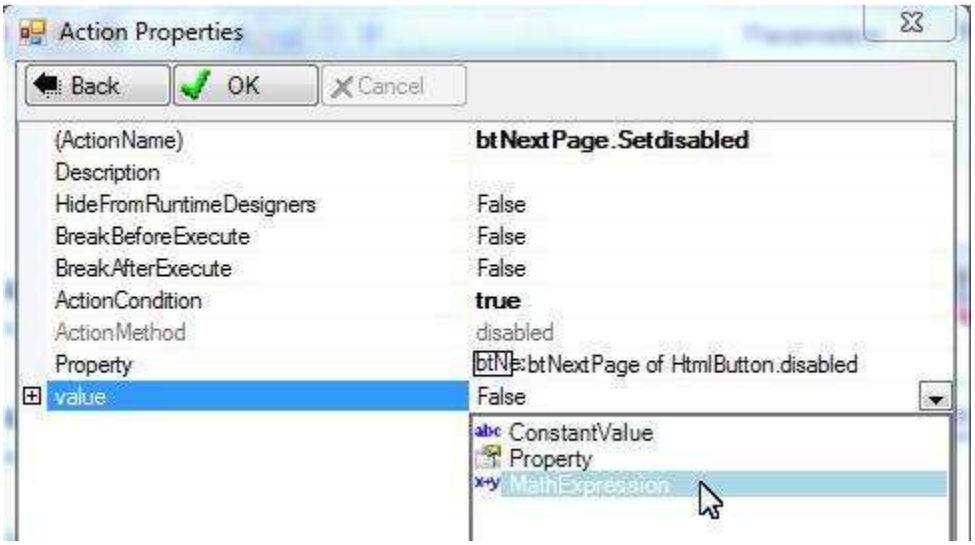


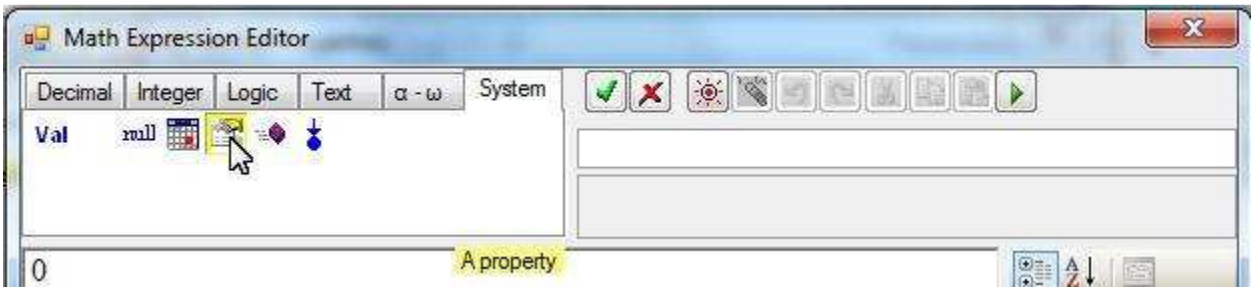
The action is created and assigned to the button:

Let's enable or disable the button at event `PlacesQueryReturned`. We may do it on the existing event handler:

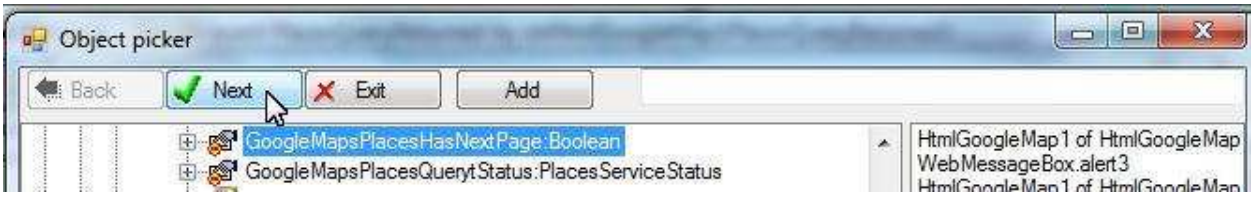


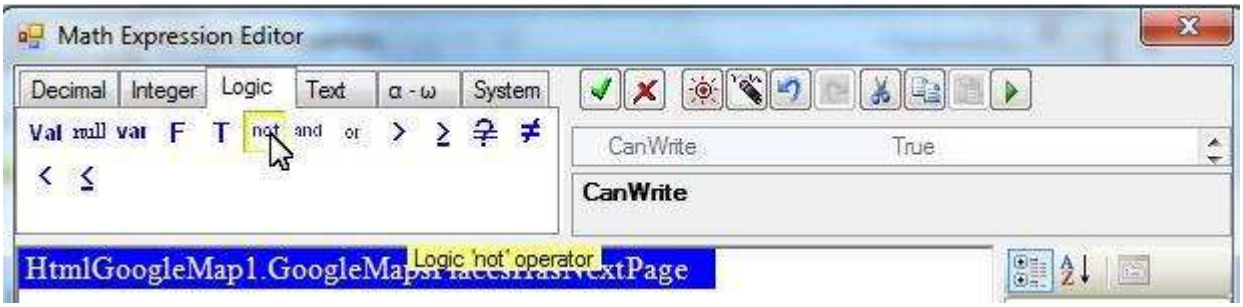

Method Designer - Handle event PlacesQueryReturned by onHtmlGoogleMap1PlacesQueryReturned1
Parameters
(Name)
btNextPage
Main
btNextPage
Create Action
Create Set Property Action
Assign Action
Create Set Value Action
BackColor
Bounds
ClientSize
Cursor
disabled
Font
No
WebMessageBox.ale

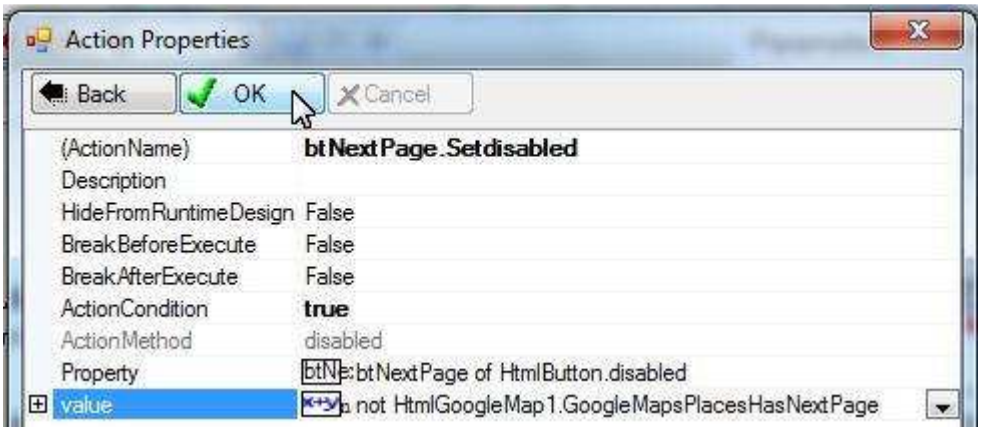

Action Properties
Back
OK
Cancel
(ActionName)
btNextPage.Setdisabled
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
disabled
Property
btNe:btNextPage of HtmlButton.disabled
value
False
ConstantValue
Property
MathExpression


Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
0
A property

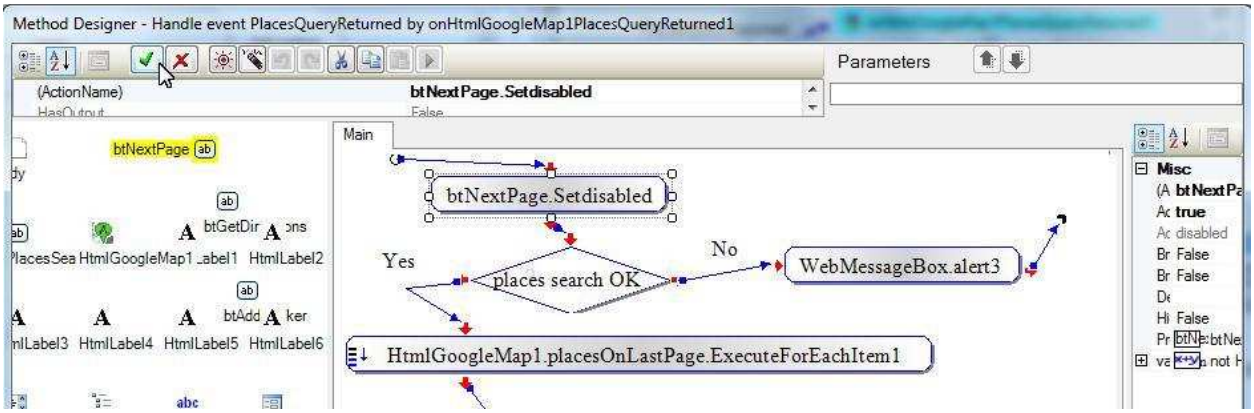

HtmlGoogleMap1 of HtmlGoogleMap
Instance Members
Properties inherited
BorderStyle:BorderStyle
HtmlGoogleMap1 of HtmlGoogleMap
WebMessageBox.alert3
HtmlGoogleMap1 of HtmlGoogleMap
placesOnLastPage:GoogleMapsPlac
placesOnLastPage:GoogleMapsPlac


Object picker
Back
Next
Exit
Add
GoogleMapsPlacesHasNextPage:Boolean
GoogleMapsPlacesQueryStatus:PlacesServiceStatus
HtmlGoogleMap1 of HtmlGoogleMap
WebMessageBox.alert3



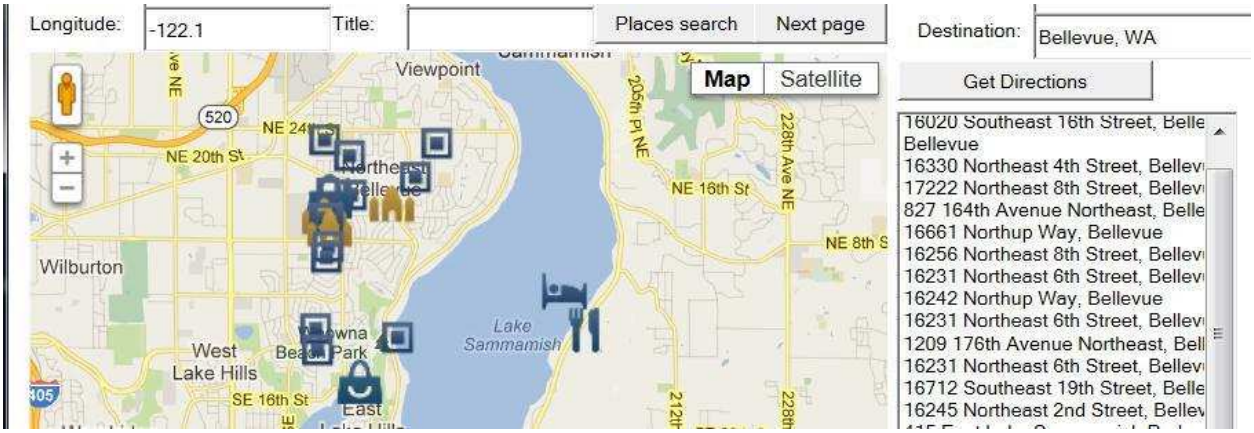


Link the new action as the first action:

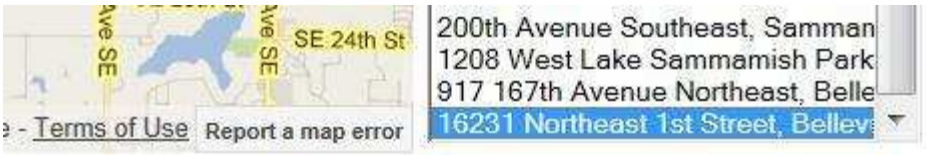


Let's launch the web page and try it. Initially button "Next page" is disabled:

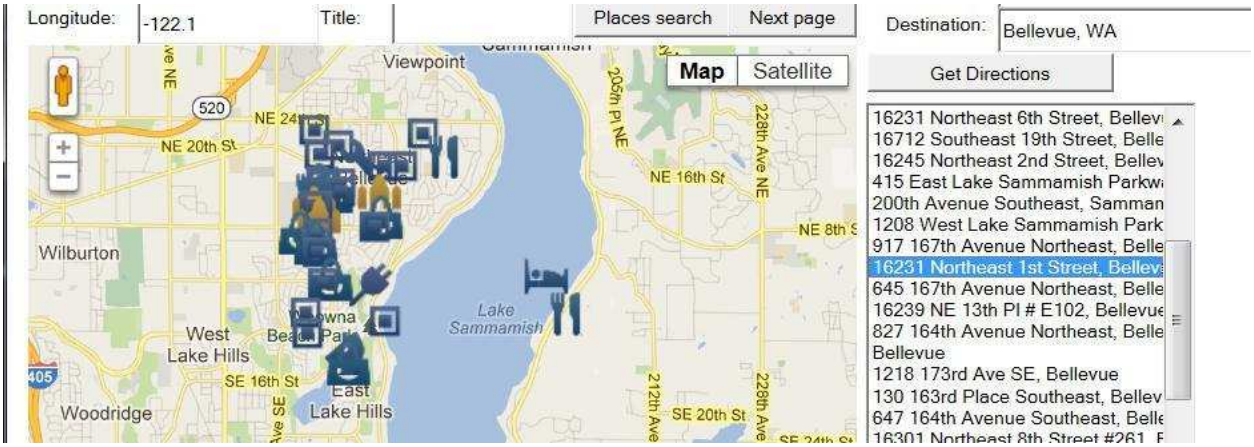
Click “Places search”. We can see that “Next page” is enabled:



Also note how many items in the list box:



Now click “Next page”. We can see more places appear on the map and appended to the list box:



We may keep clicking “Next page” until button “Next page” is disabled, indicating no more places:

Get place details

Following properties, methods and events of a place-object are related to getting details for a place.

- A `fetchDetails` action is used to request place details.
- An event, `gotDetails`, occurs when place details are available after executing `fetchDetails`.
- Property `detailsStatus` – it indicates the result of a `fetchDetails` action at event gotDetails.
- Property `details` – it contains place details information.
- Property `hasDetails` – it indicates validity of property `details`.
- HtmlGoogleMap also has a gotPlaceDetails event. It occurs after gotDetails event of a place.

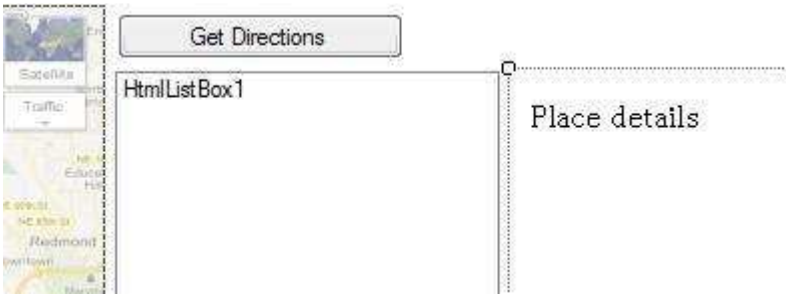
Programming sample

Suppose we want to do such programming:

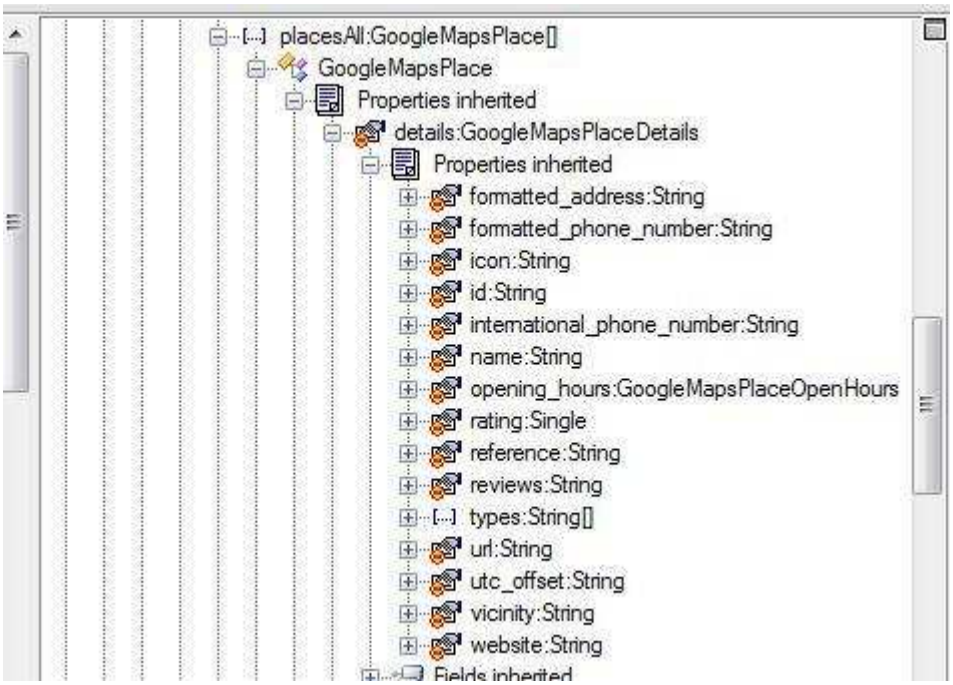
- An HTML box is used to display place details.
- The user clicks a list item to select a place.
- The marker of the selected place is made visible on the map and markers for all other places are made invisible.
- If the details of the place are valid then display the details in the HTML box.
- If the details of the place are invalid then execute a fetchDetails action.
- When gotPlaceDetails event of HtmlGoogleMap occurs the details are displayed in the HTML box.

Display details

Add an HtmlContent control for displaying place details:

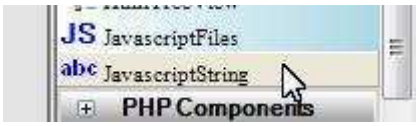


We may examine place details object from Object explorer:

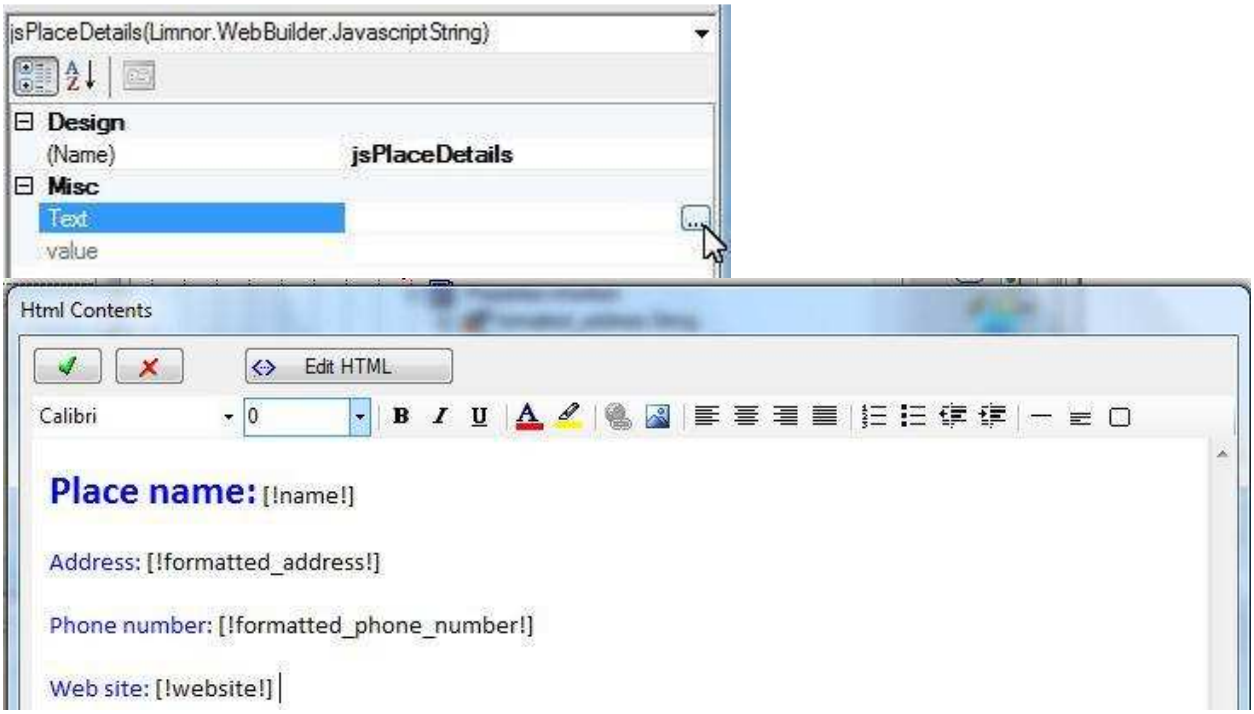


Suppose we want to display formatted_address, formatted_phone_number, name, and website.

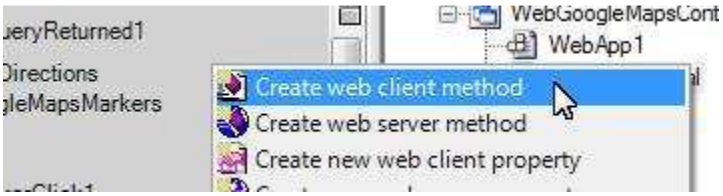
We may use a JavascriptString to form place details information display HTML:



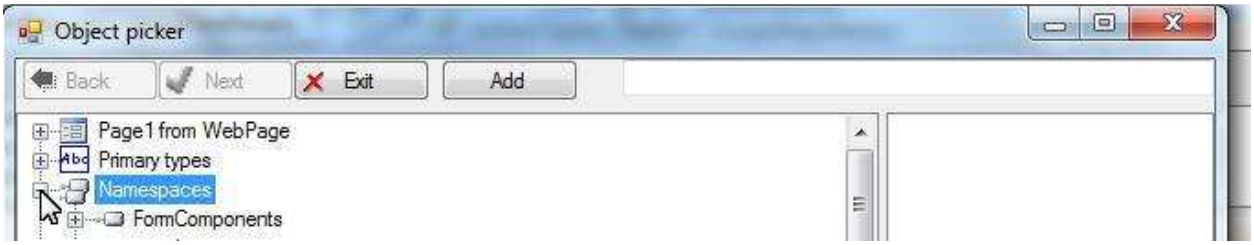
Set its Text property for creating an HTML template:

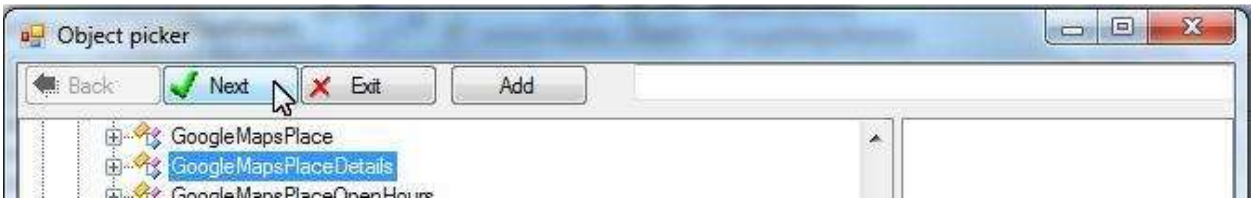


Create a method to display place details:

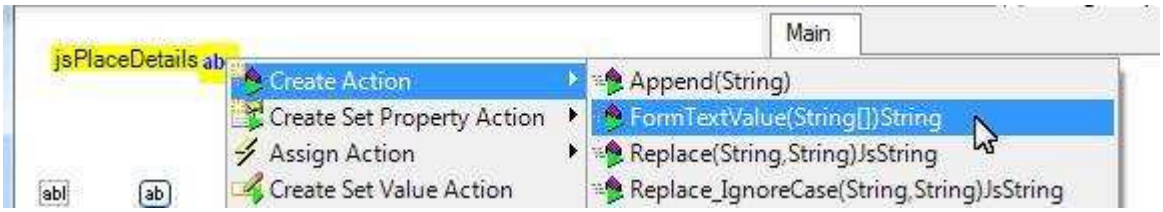


Add a parameter for place details:

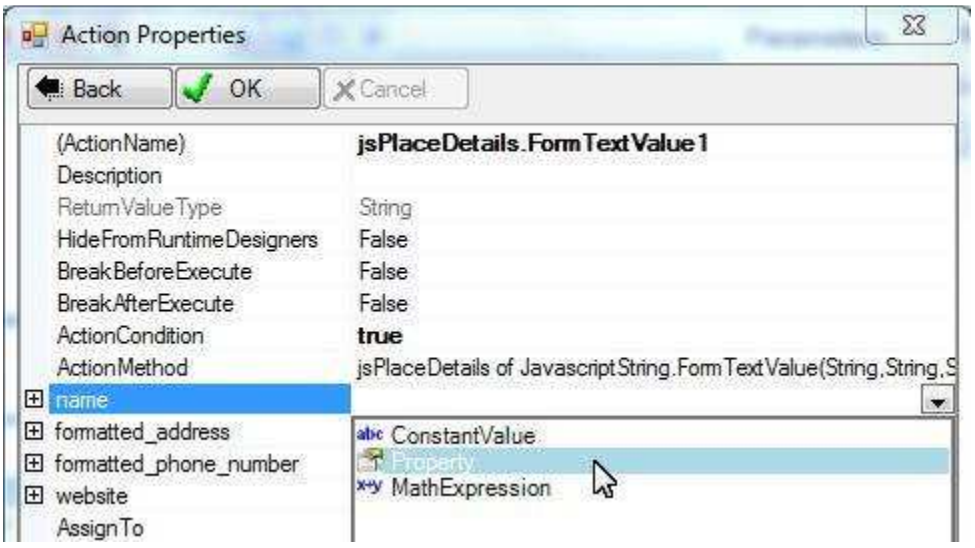


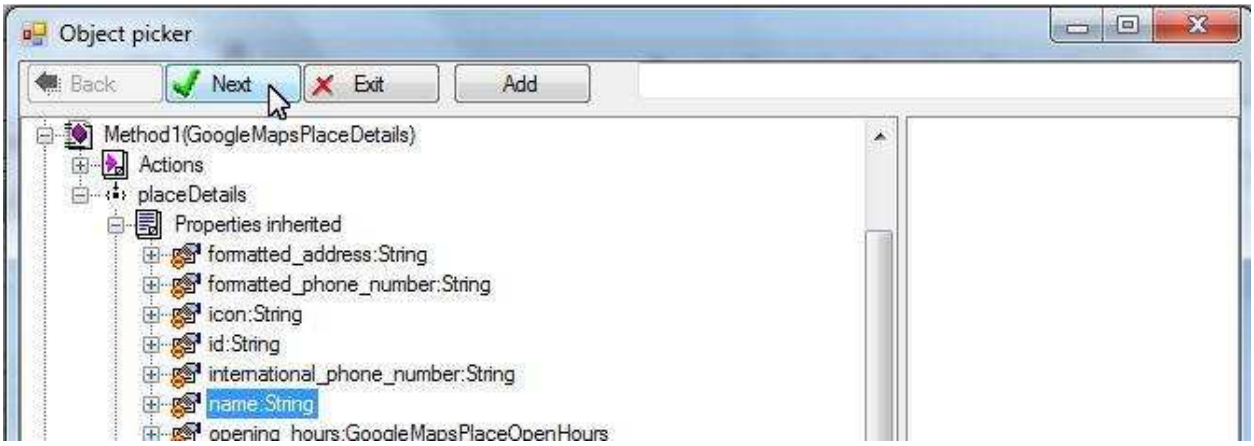


Create an action to form HTML for place details:

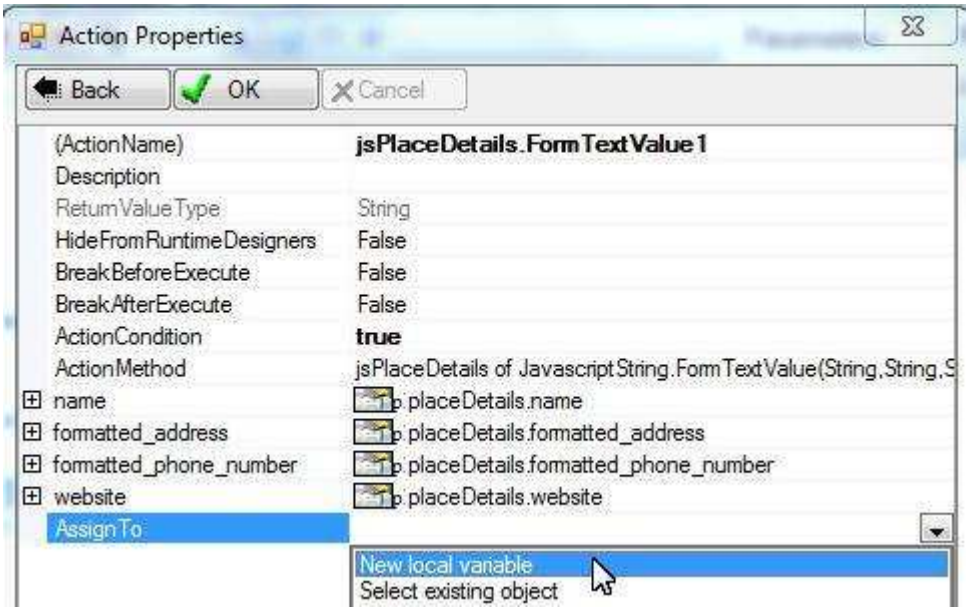


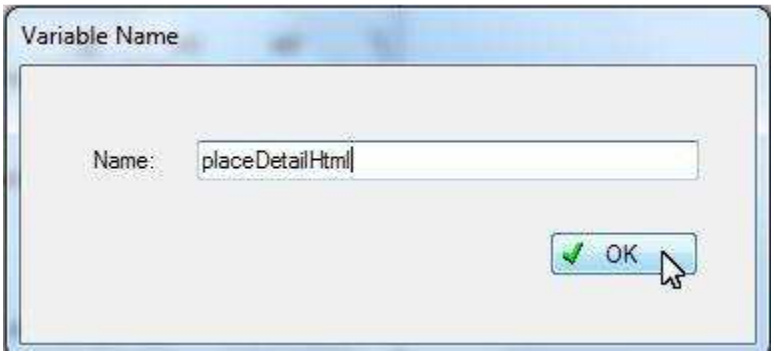
Pass place details to the action:

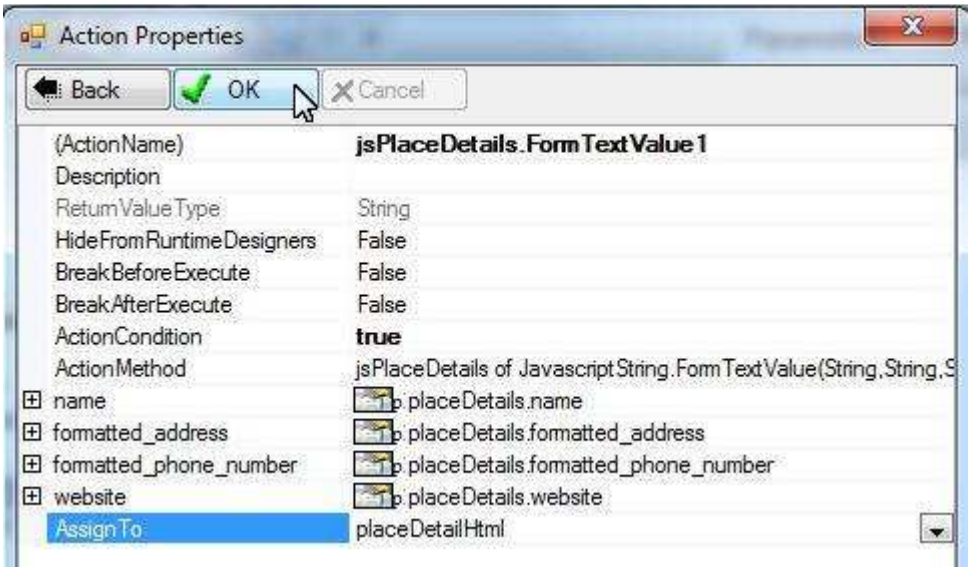




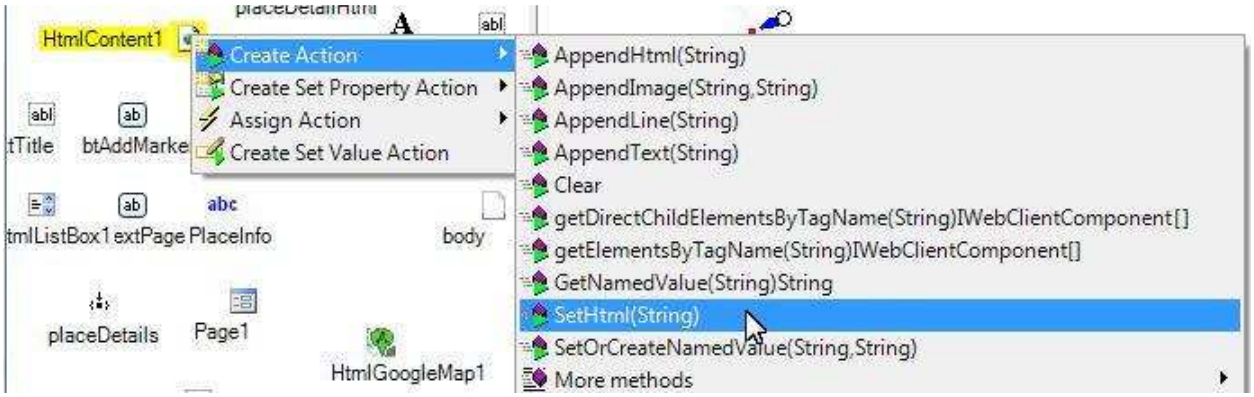
Use a variable for the result:

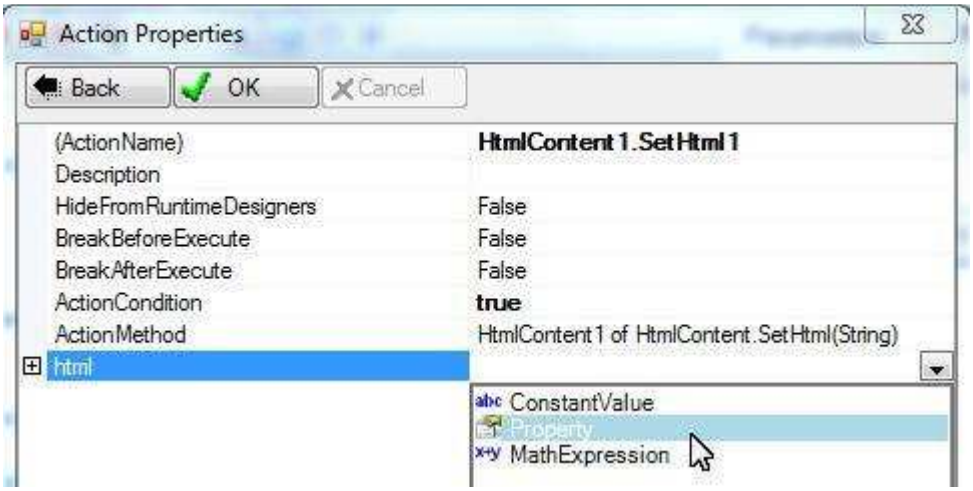


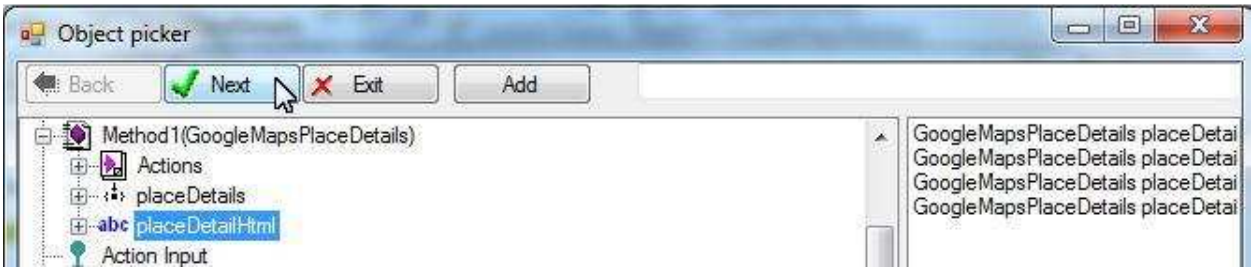


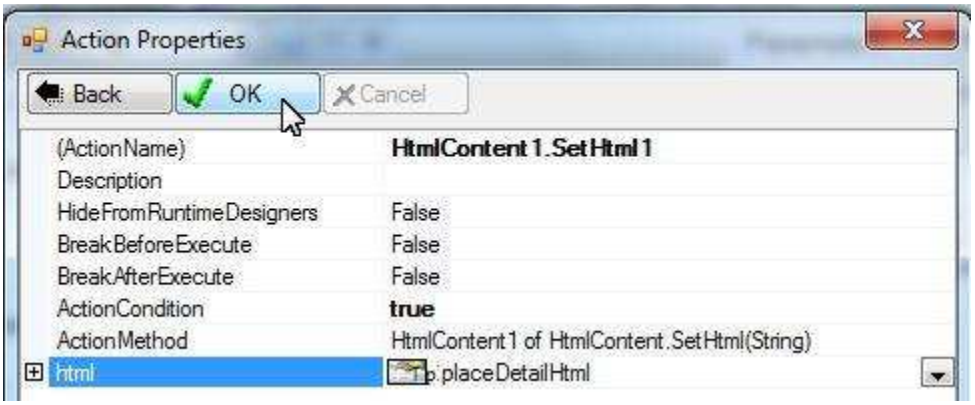


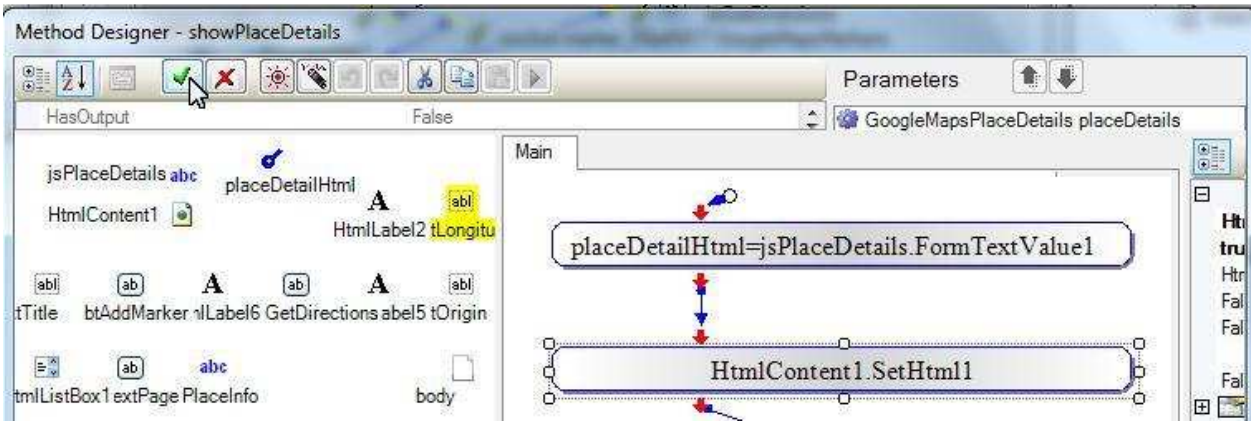
Use the html on the HTML box:



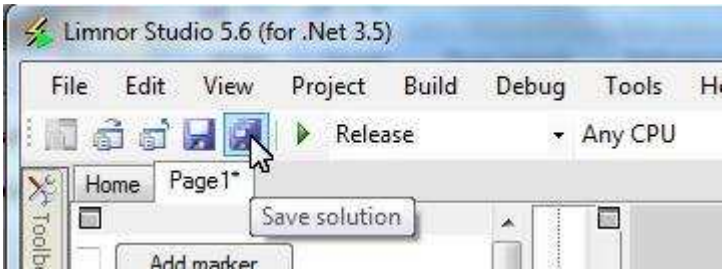




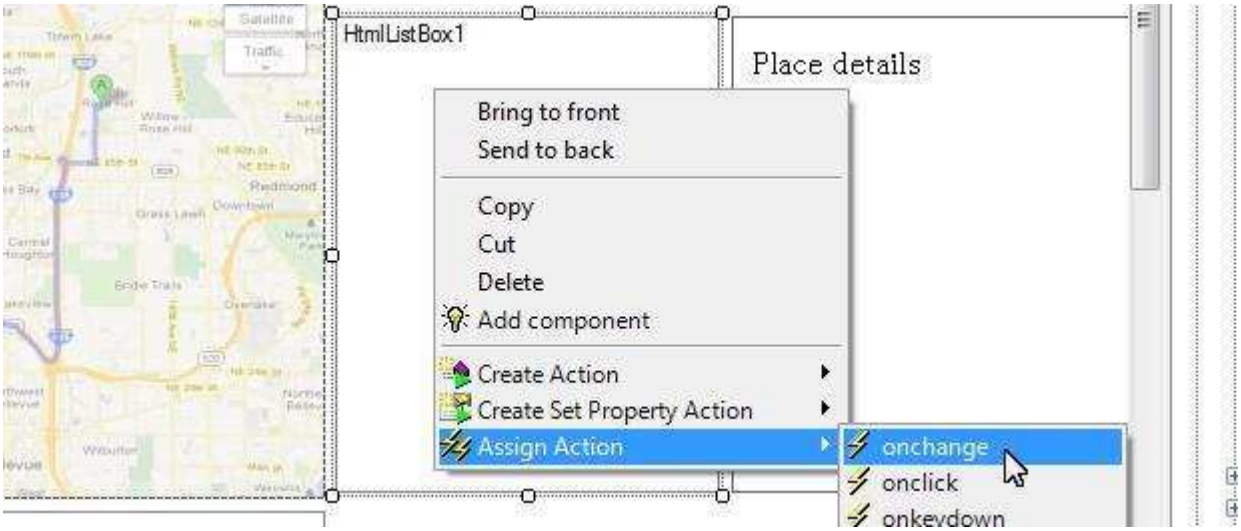


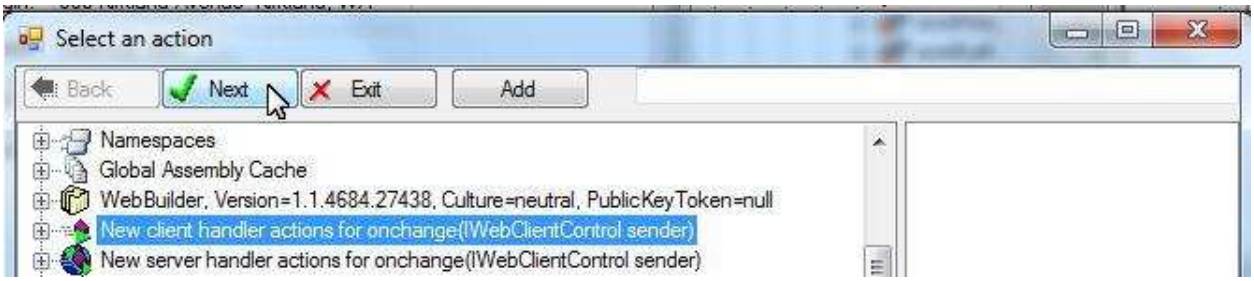


Save the project so that the method is available for programming:

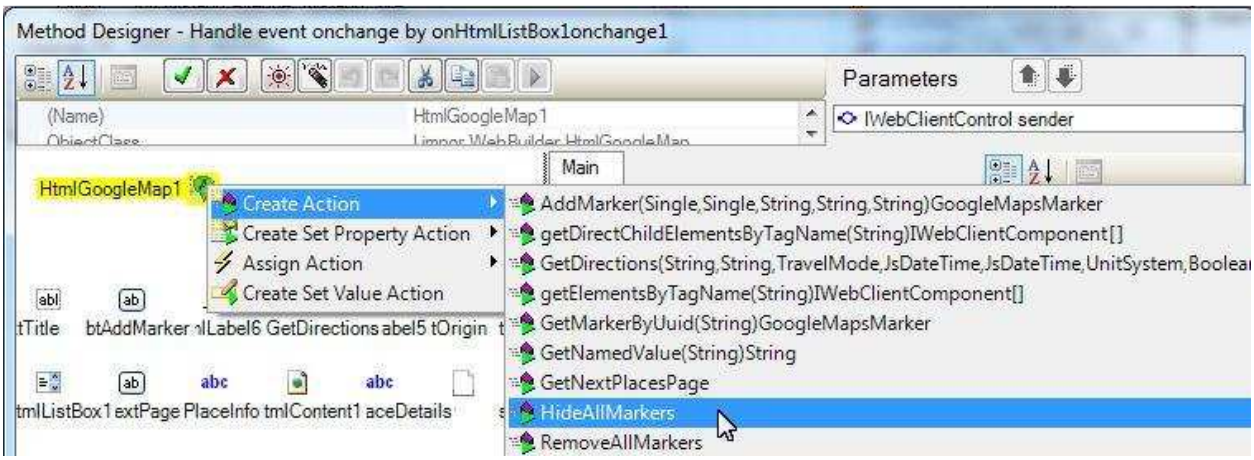


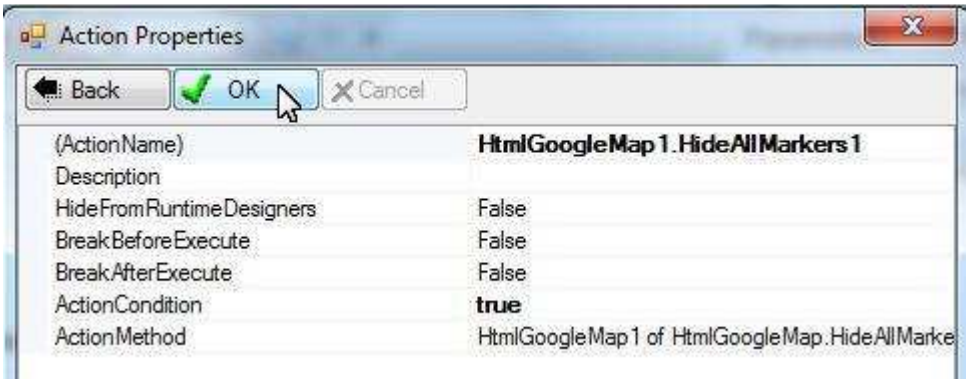
Handle place item selection



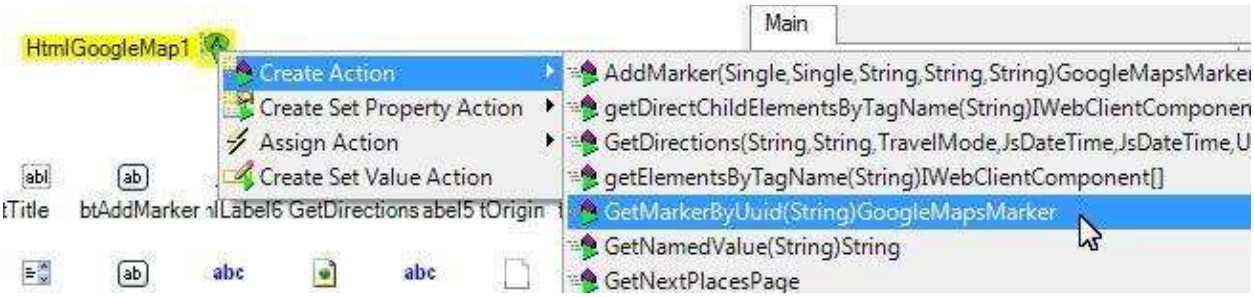


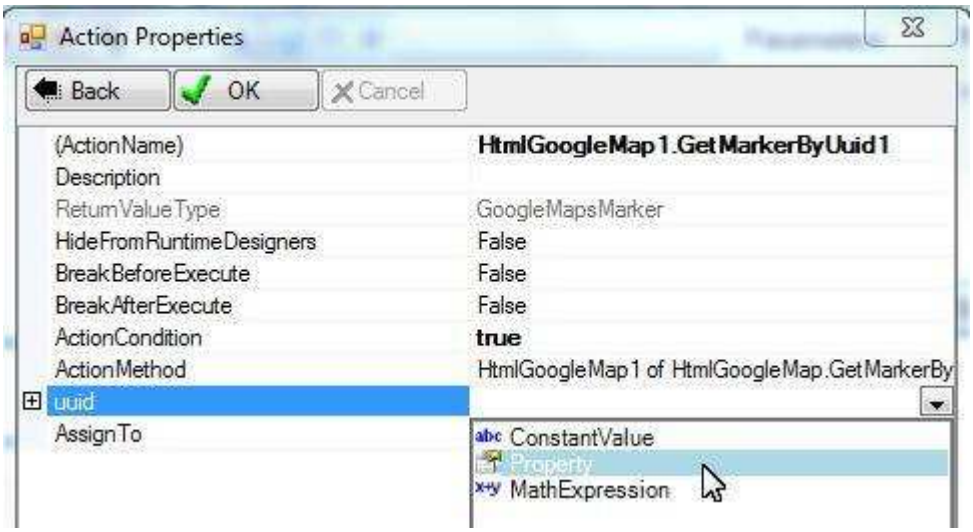
Hide all places

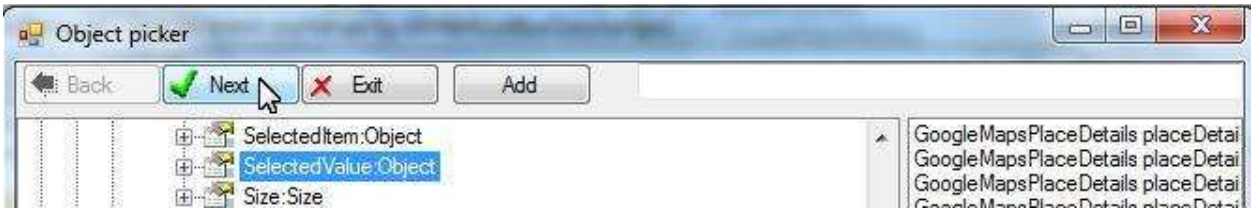




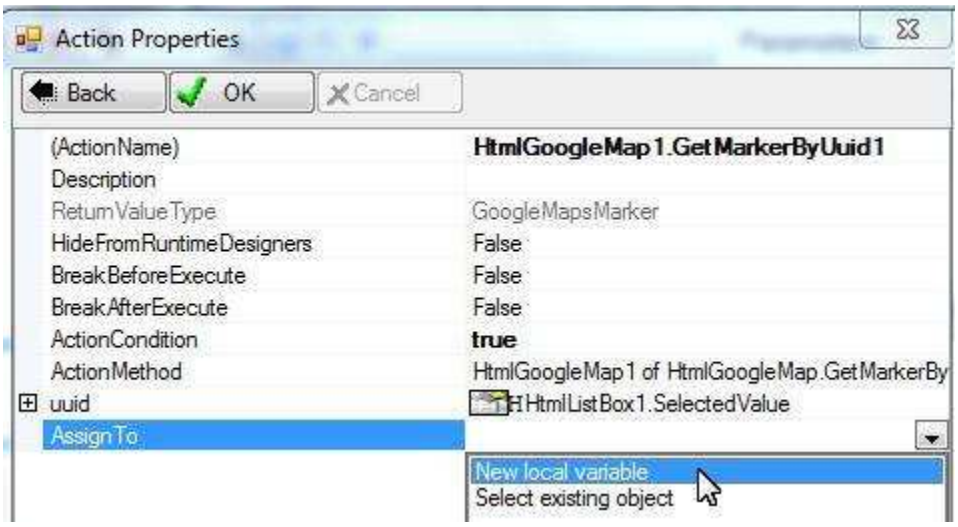
Find place marker

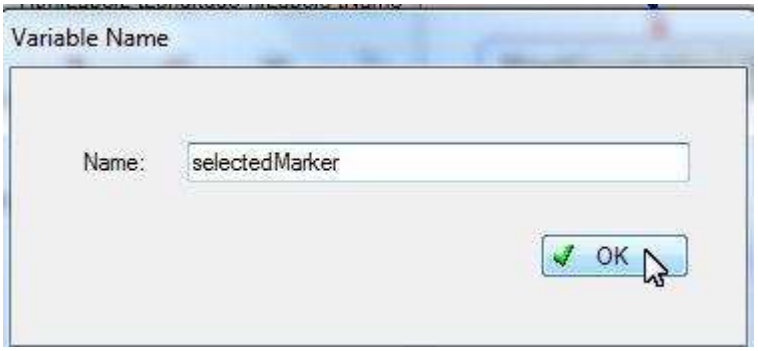


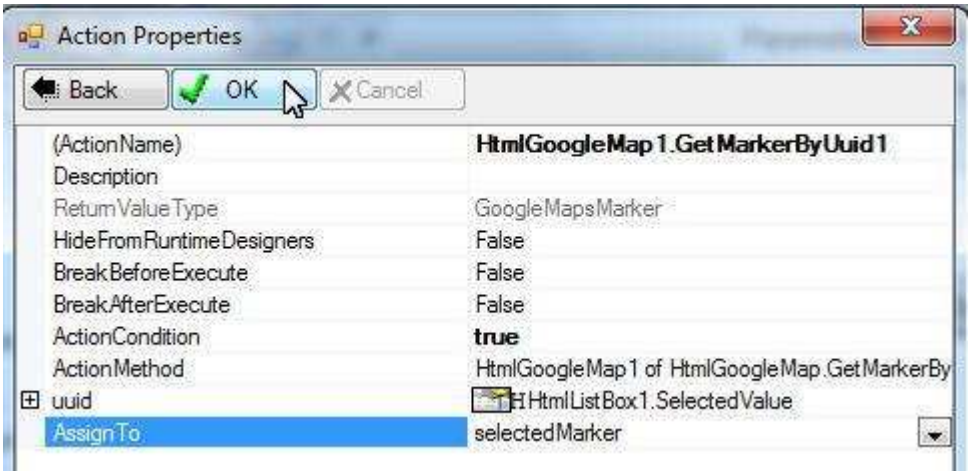




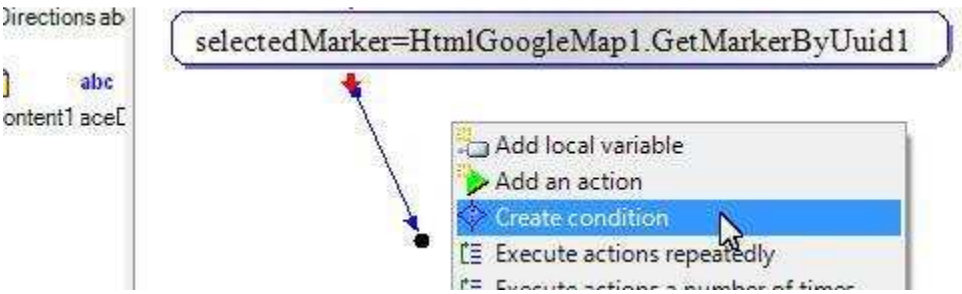
Use a variable for found marker:

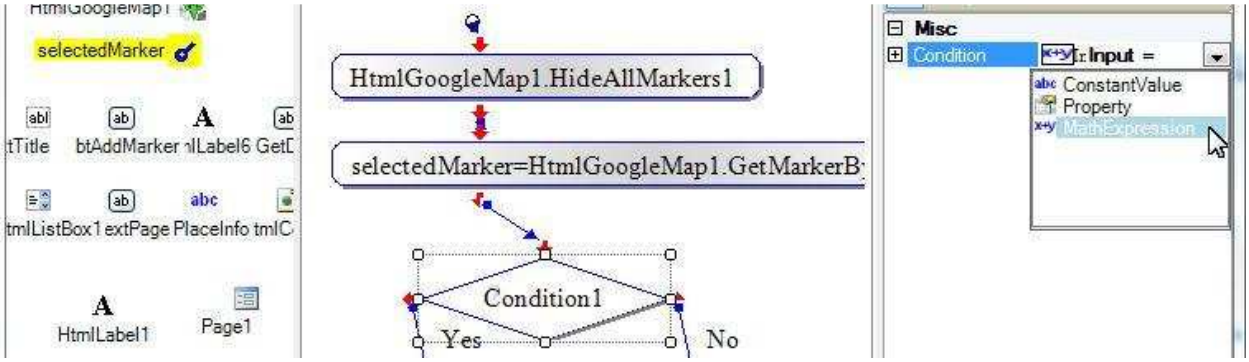




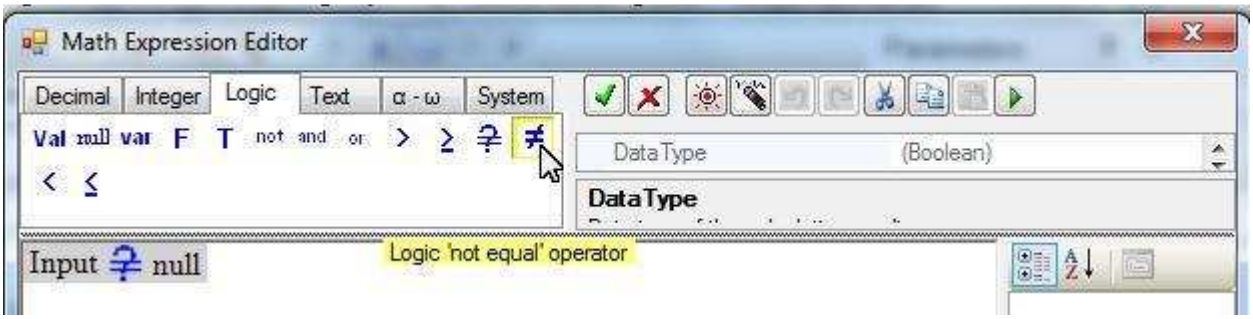


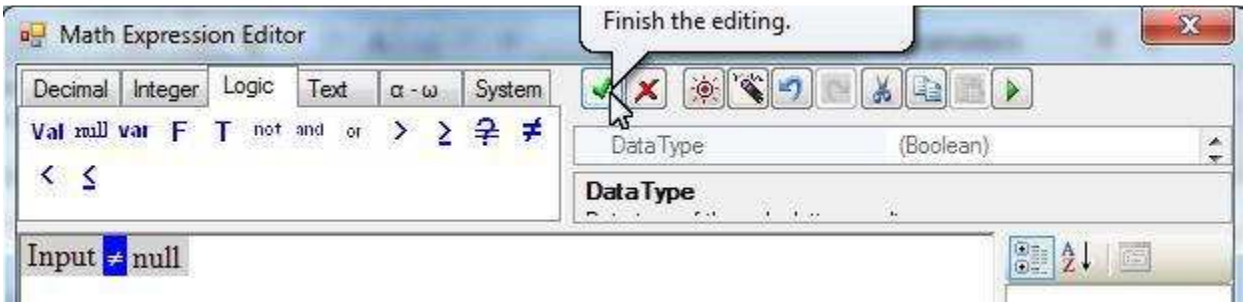
Add a Condition action to check whether a place marker is found:



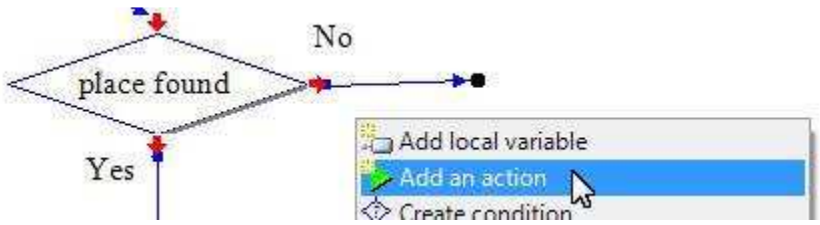


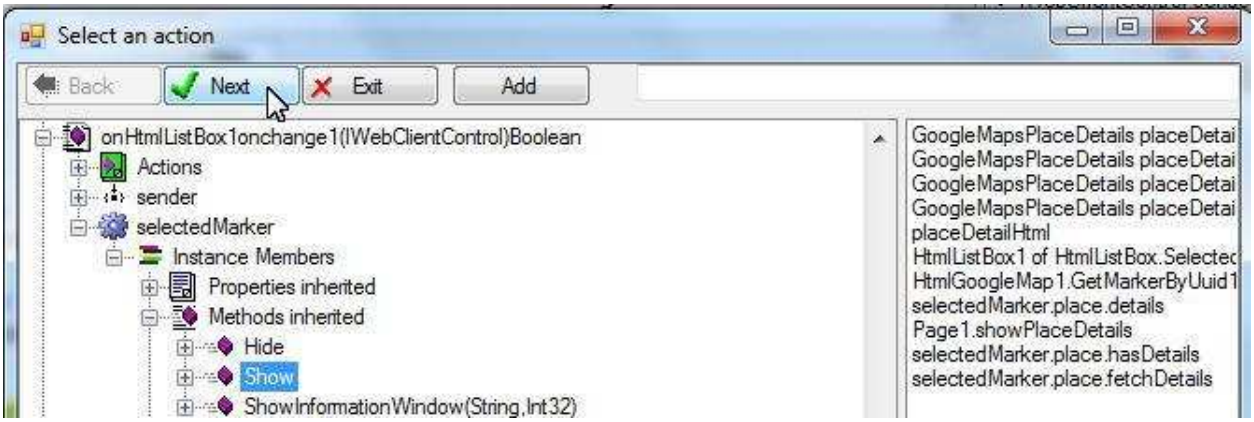
If the result is not null then we found a marker:

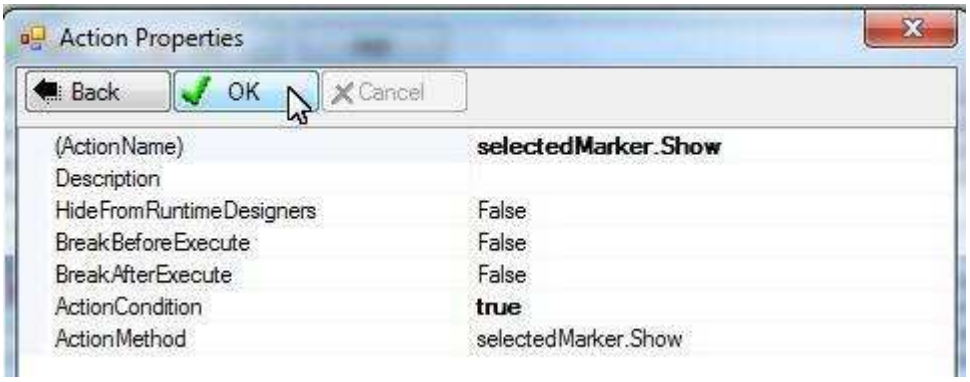




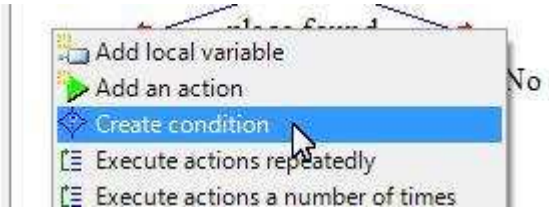
Show the marker:

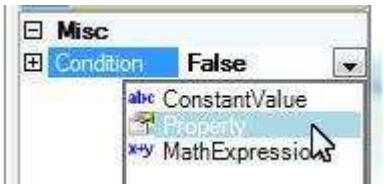


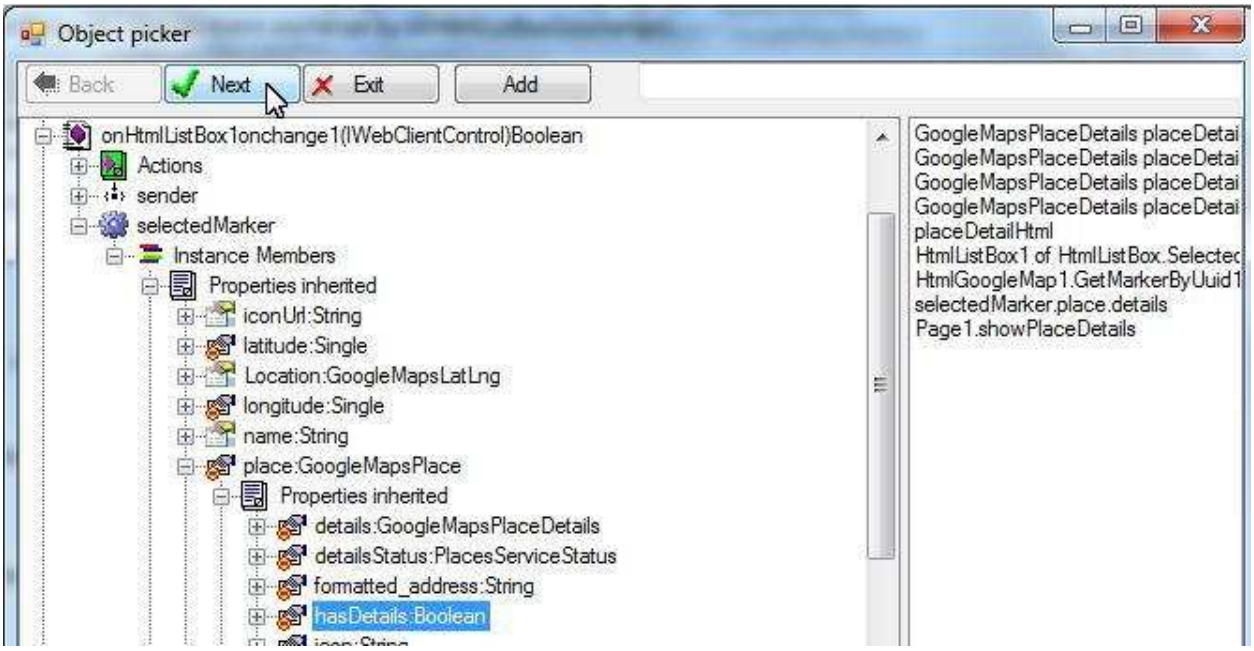




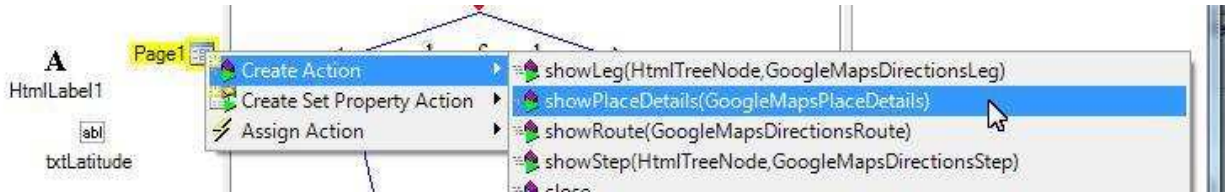
Check if place details are available:

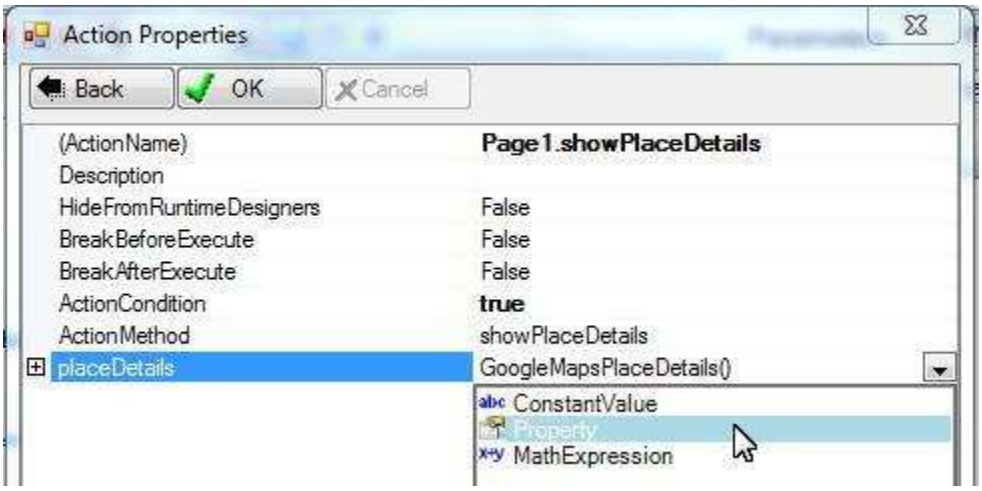


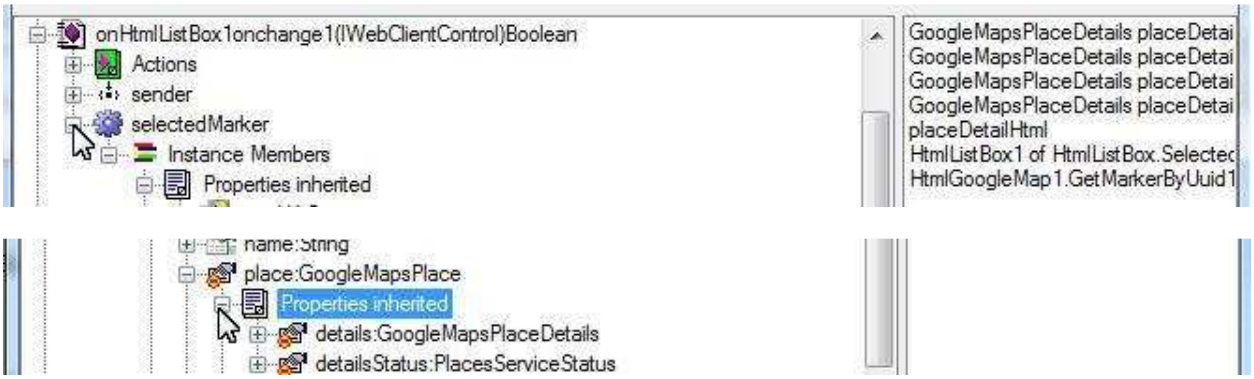


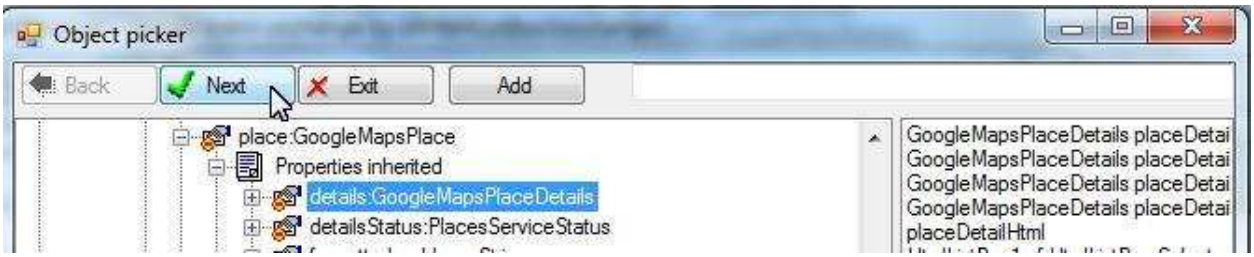


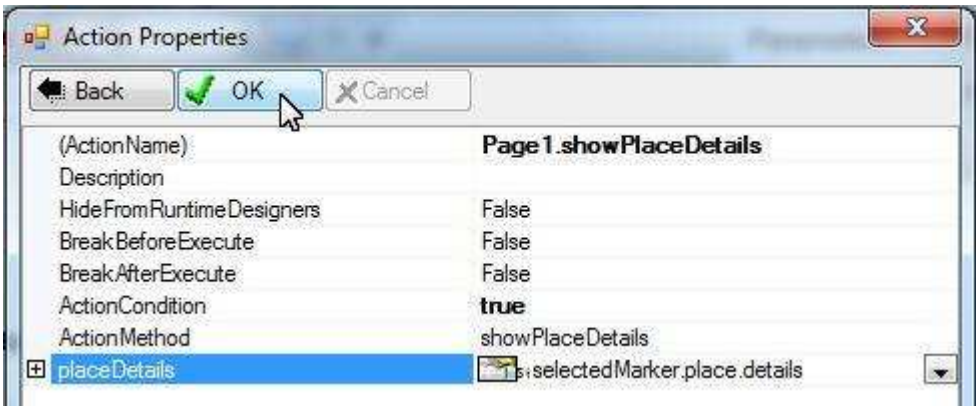
Execute showPlaceDetails if place details are available:



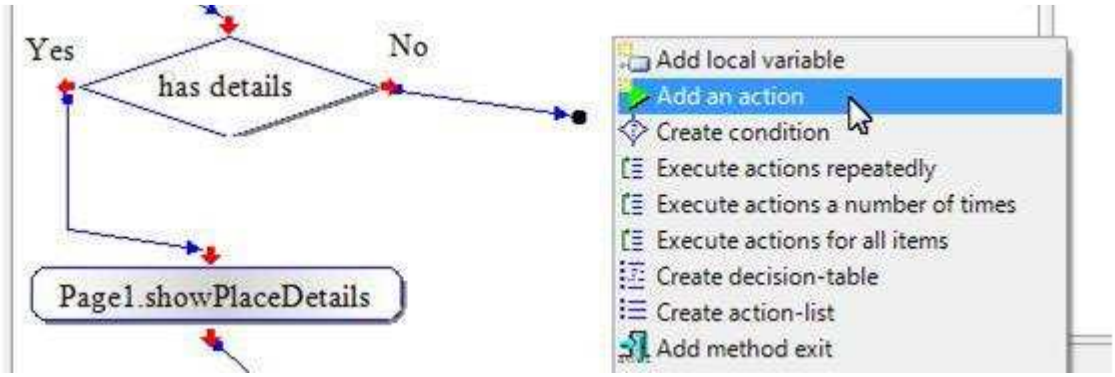


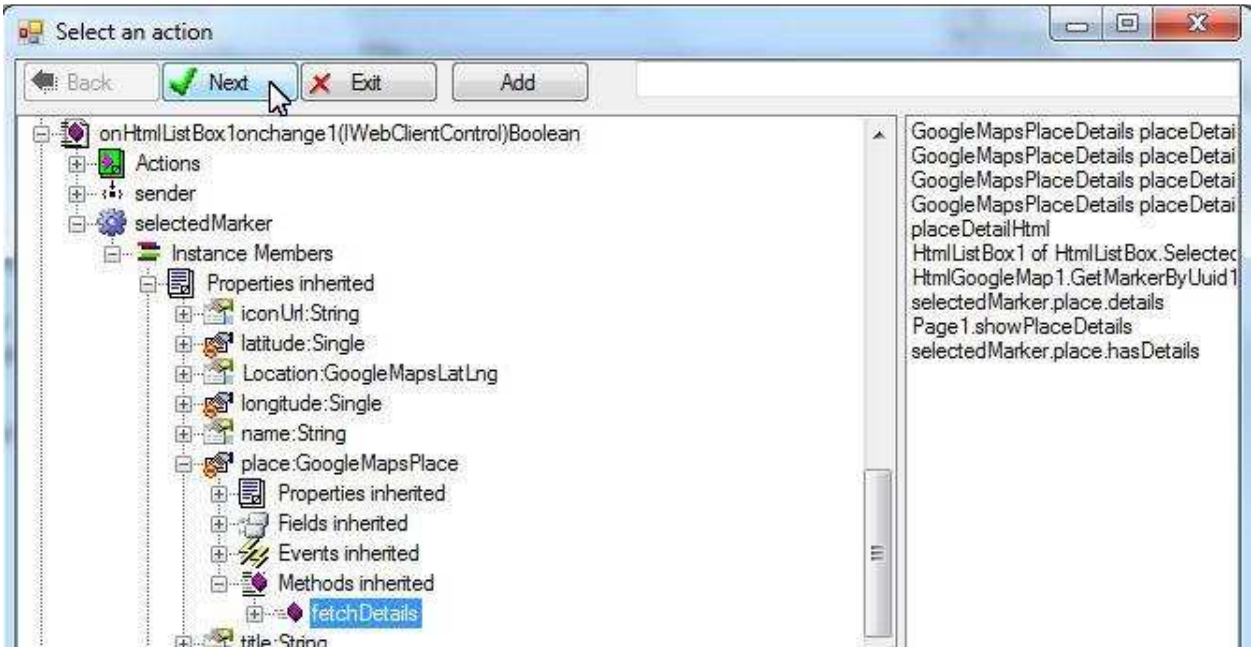


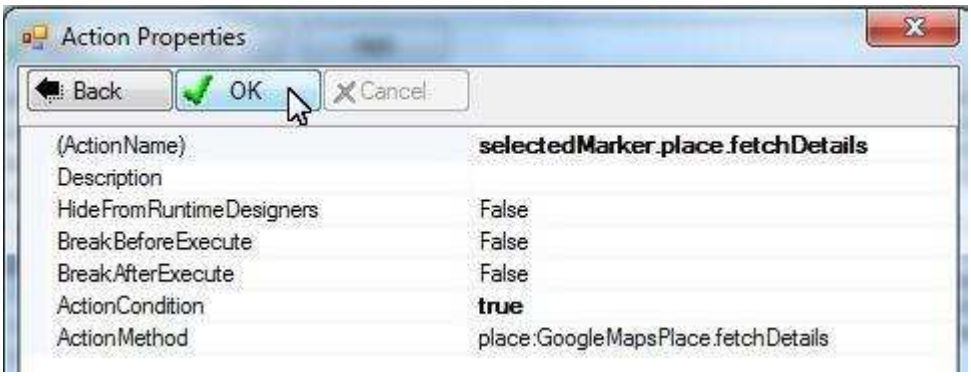




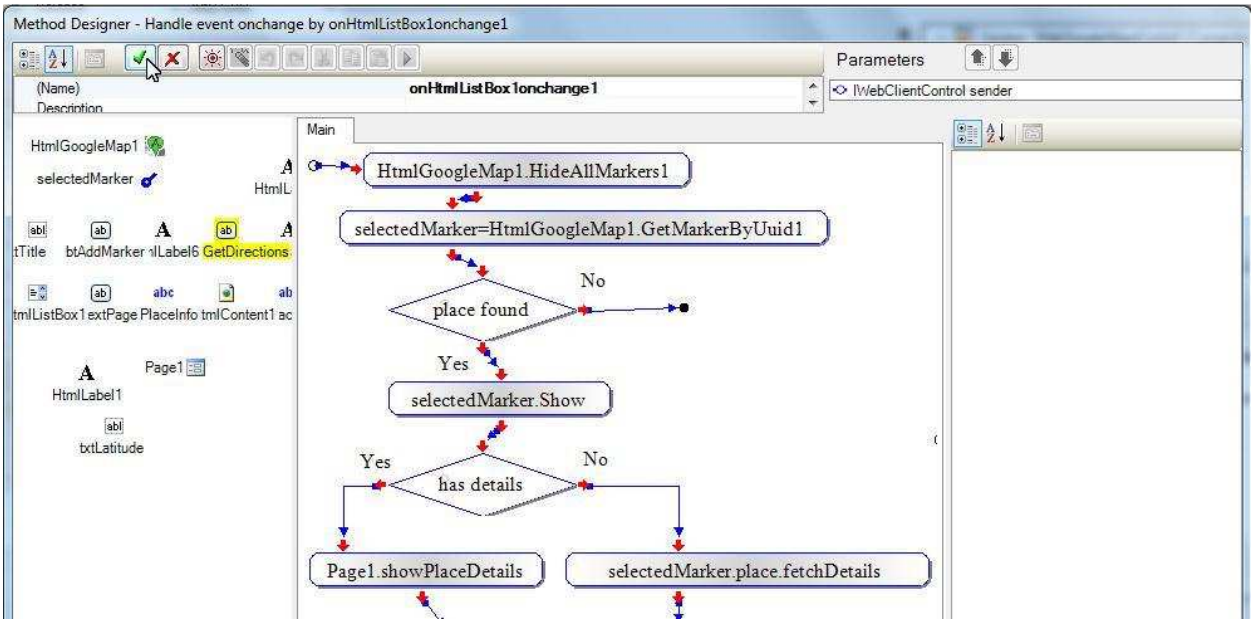
Execute fetchDetails if place details are not available:



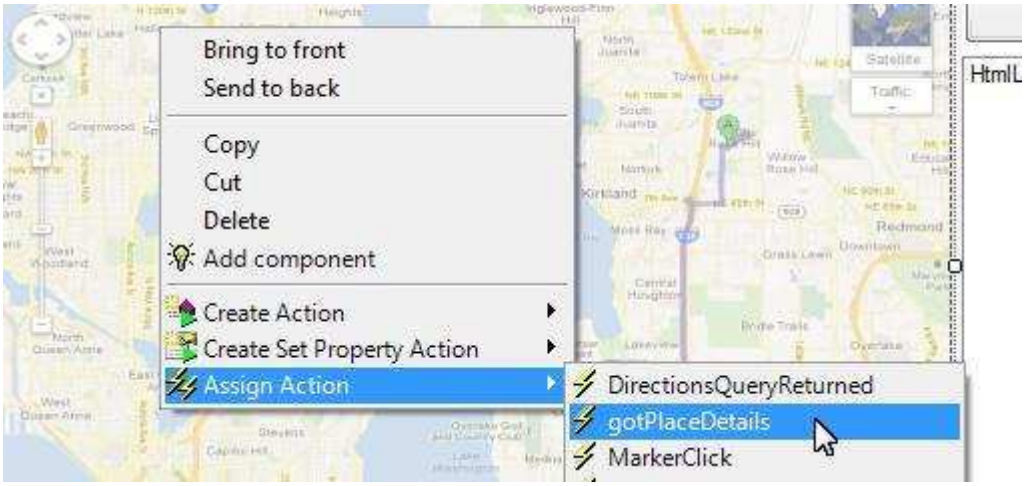


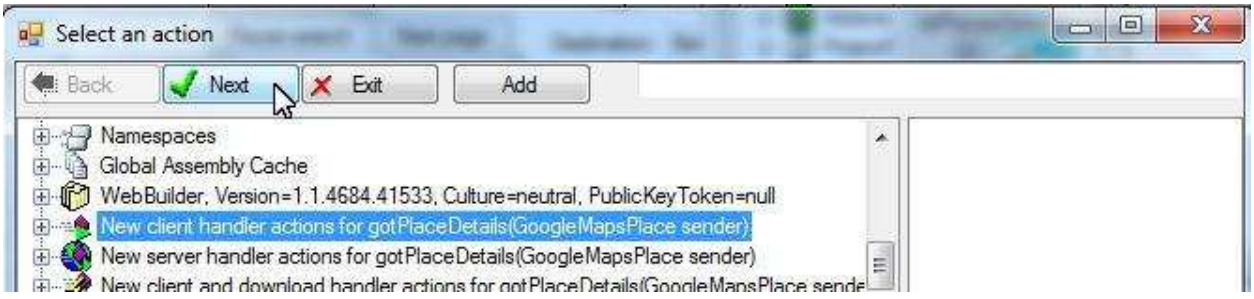


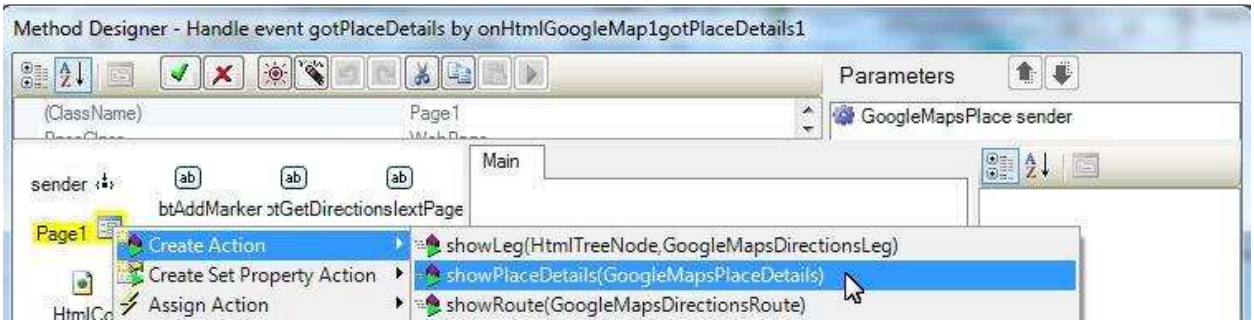
We are done handling the place selection:

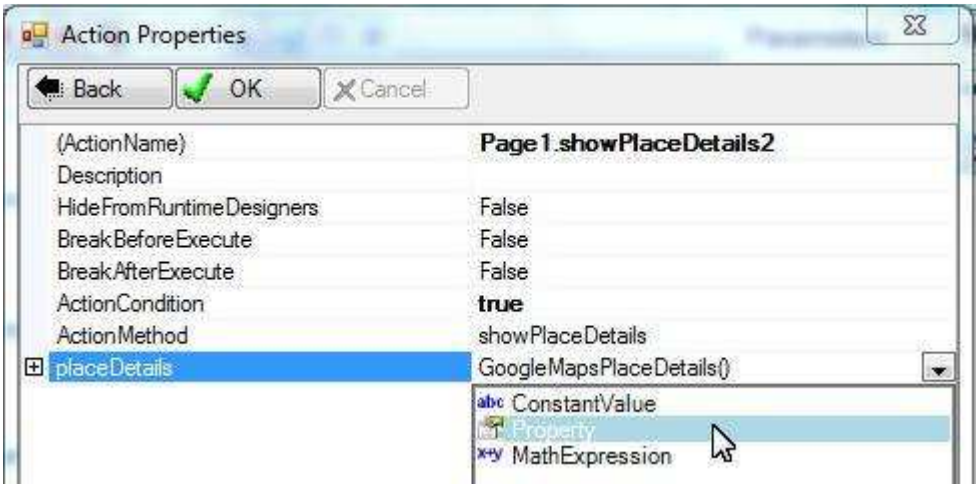


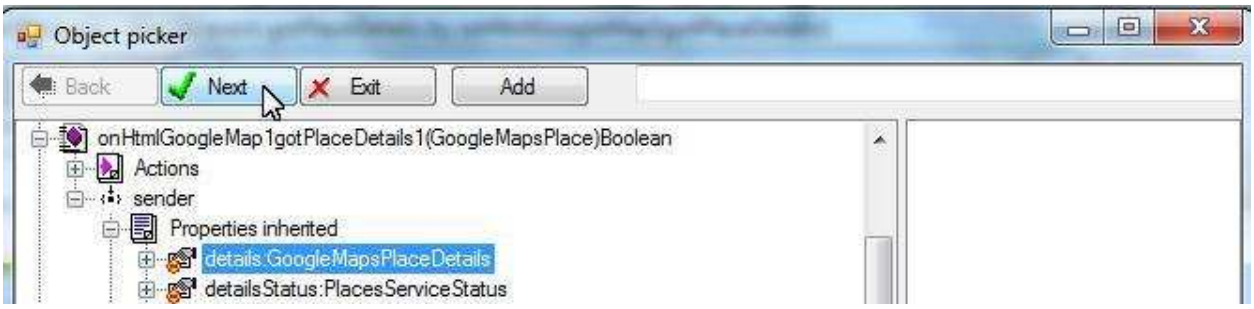
Handle event gotPlaceDetails

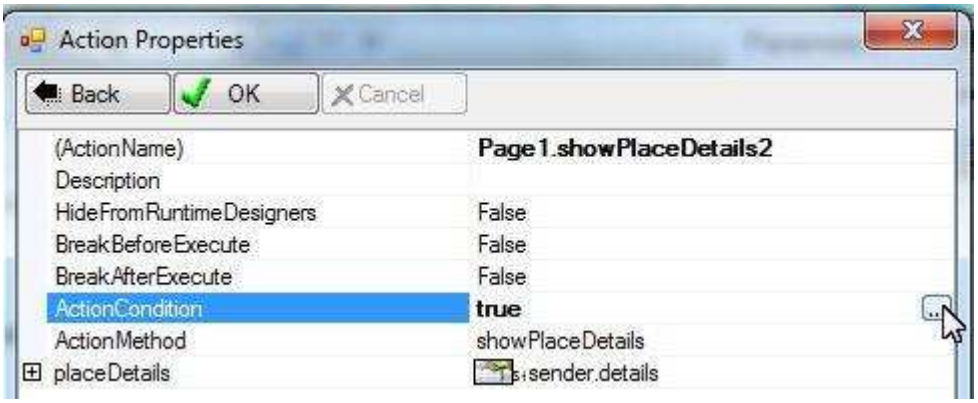




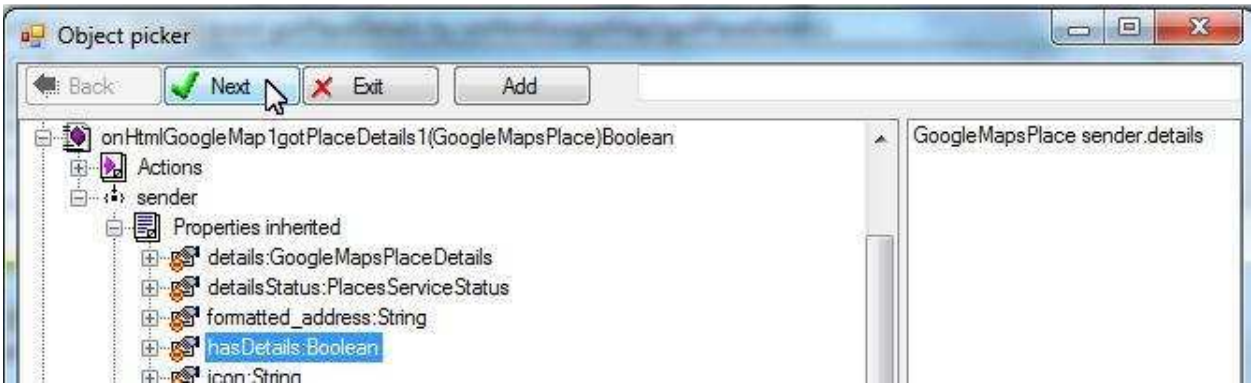

Method Designer - Handle event gotPlaceDetails by onHtmlGoogleMap1gotPlaceDetails1
Parameters
(ClassName)
Page1
GoogleMapsPlace sender
Main
sender
btAddMarker
btGetDirections
TextPage
Page1
Create Action
Create Set Property Action
Assign Action
showLeg(HtmlTreeNode,GoogleMapsDirectionsLeg)
showPlaceDetails(GoogleMapsPlaceDetails)
showRoute(GoogleMapsDirectionsRoute)


Action Properties
Back
OK
Cancel
(ActionName)
Page1.showPlaceDetails2
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
showPlaceDetails
placeDetails
GoogleMapsPlaceDetails()
ConstantValue
Property
MathExpression

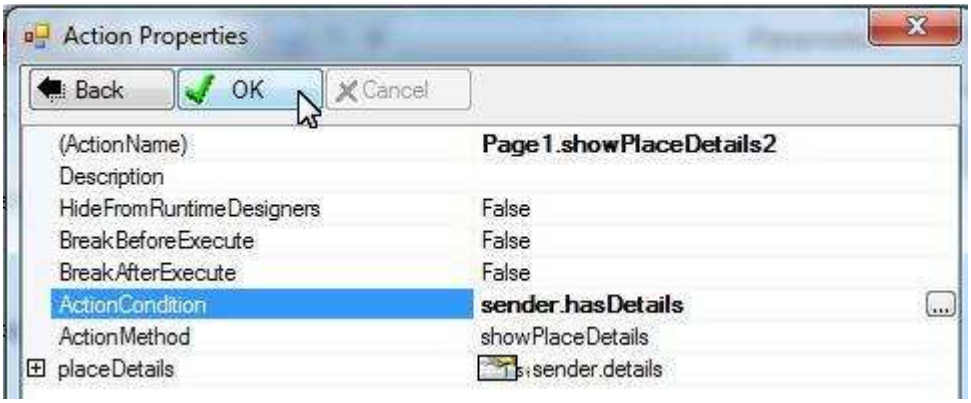

Object picker
Back
Next
Exit
Add
onHtmlGoogleMap1gotPlaceDetails1(GoogleMapsPlace)Boolean
Actions
sender
Properties inherited
details:GoogleMapsPlaceDetails
detailsStatus:PlacesServiceStatus

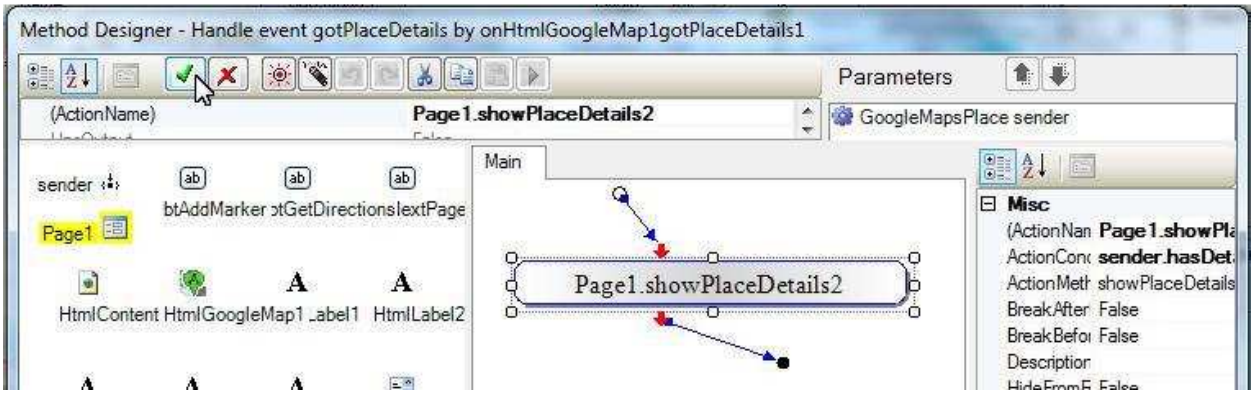

Action Properties
Back
OK
Cancel
(ActionName)
Page1.showPlaceDetails2
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
showPlaceDetails
placeDetails
sender.details


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


Object picker
Back
Next
Exit
Add
onHtmlGoogleMap1gotPlaceDetails1(GoogleMapsPlace)Boolean
Actions
sender
Properties inherited
details:GoogleMapsPlaceDetails
detailsStatus:PlacesServiceStatus
formatted_address:String
hasDetails:Boolean
GoogleMapsPlace sender.details

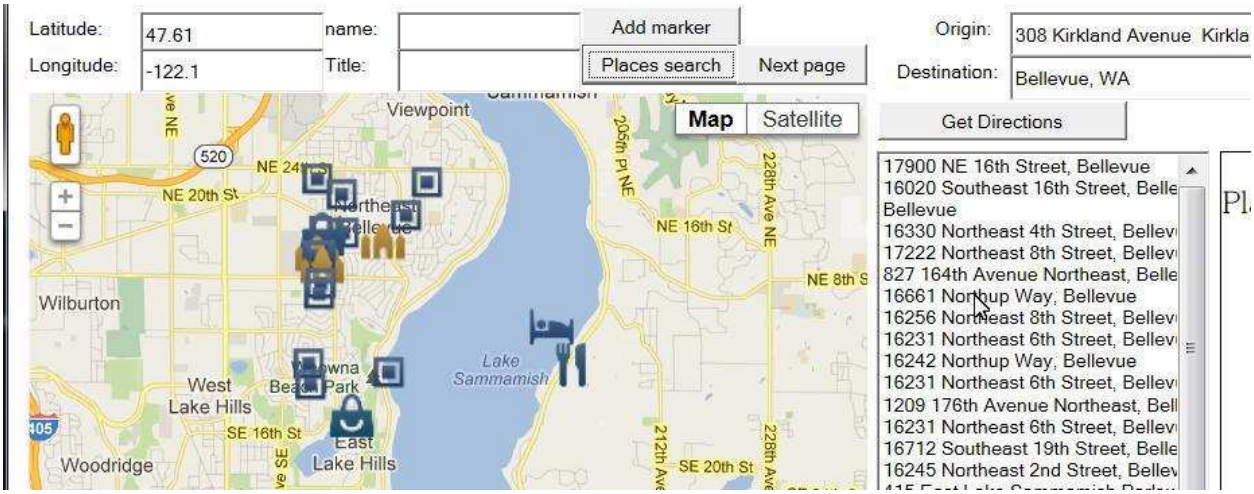

Math Expression Editor
Finish the editing.
Decimal
Integer
Logic
Text
α - ω
System
Val
null
CanWrite
True
sender.hasDetails


Action Properties
Back
OK
Cancel
(ActionName)
Page1.showPlaceDetails2
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
sender.hasDetails
ActionMethod
showPlaceDetails
placeDetails
sender.details

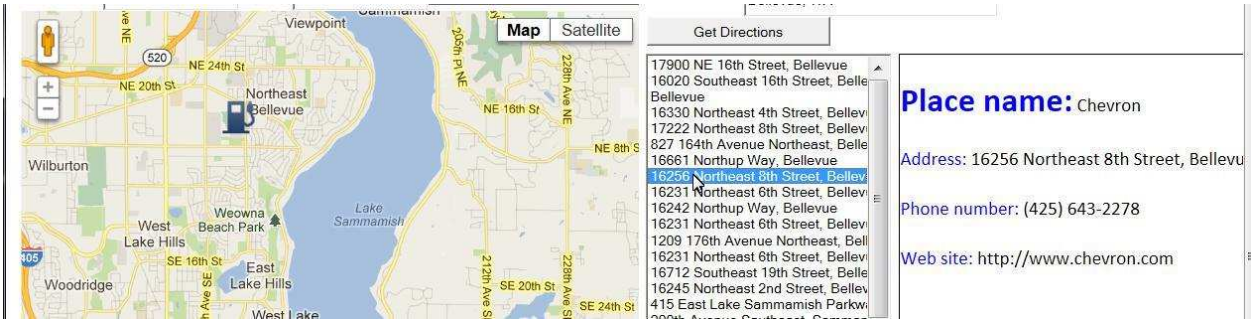


Test

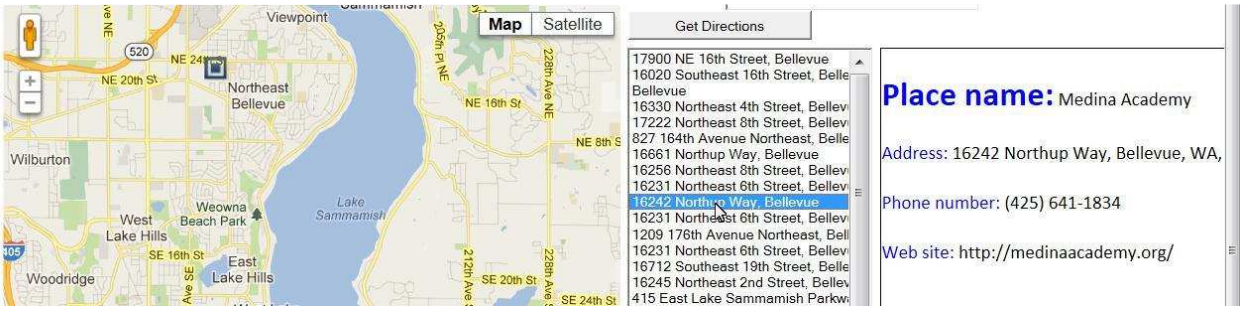
Launch the web page. Click “Places search”:



Select a list item. All place markers are hidden except the selected place. The details of selected place are displayed in the HTML box.



Select another item:



Feedback

Please send your questions, comment and suggestions to support@limnor.com. Thanks!