

Use Google Maps Control

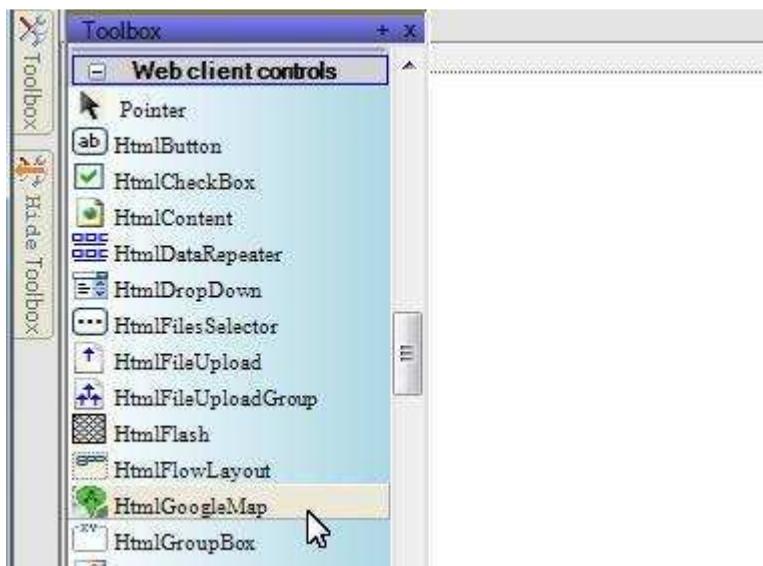
Contents

Introduction	2
Setup Google Maps.....	3
Use Markers.....	4
Add markers at design time	4
Marker Properties.....	4
Marker Methods.....	6
Marker Events.....	6
Handle marker click event at design time	6
Add markers at runtime.....	10
Handle event for all markers	20
Marker Related Methods.....	28
Get Directions	29
Create GetDirections action.....	29
Display Directions	34
Show Step	36
Add Parameters	37
Display step	39
Display sub-steps	45
Show Leg	49
Add parameters	49
Display leg	50
Display steps	54
Show route.....	58
Add route parameter	58
Create a root node	58
Display legs.....	61
Handle Directions Query Event.....	65

Show status.....	65
Clear tree view	67
Show route in tree view.....	68
Test.....	72
Places Search.....	73
Action to Search Places.....	74
Handle PlaceMarkerClick Event	79
List places.....	88
Get next page of places.....	99
Get place details.....	104
Programming sample	104
Display details	104
Handle place item selection.....	111
Hide all places	111
Find place marker	112
Handle event gotPlaceDetails	119
Test.....	122
Feedback	123

Introduction

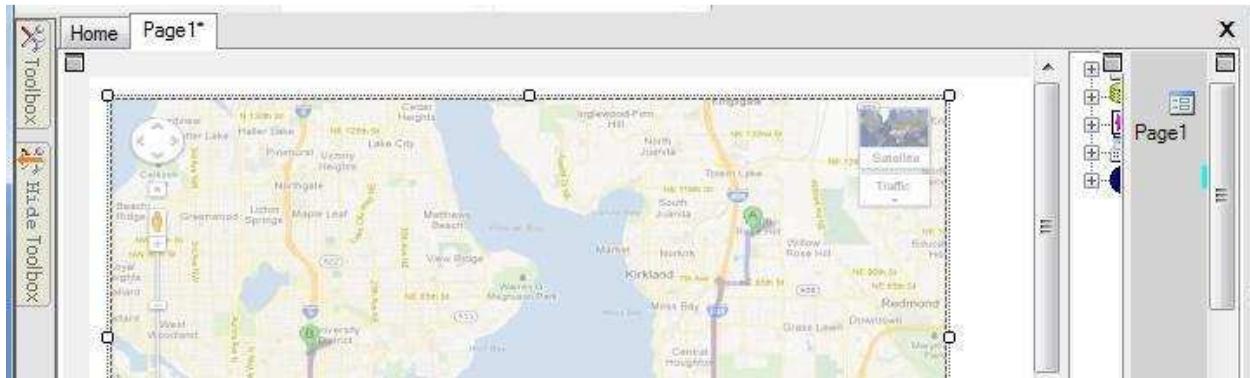
HtmlGoogleMap is a web control for using Google Maps services on web pages.



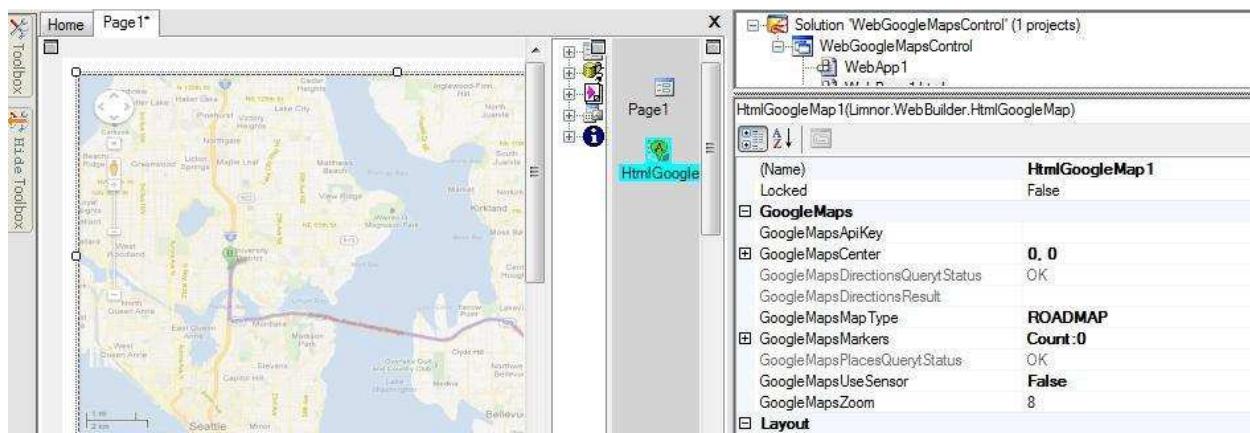
This document shows examples of using HtmlGoogleMap.

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:

The screenshot shows the Google Maps Control interface with a single marker configuration. The marker is of type ROADMAP, with a count of 1. The marker's name is "markere10bb1a4". The location is set to 0.0. The icon URL is listed as "Create a new map marker".

Property	Value
GoogleMapsMapType	ROADMAP
Count	1
marker_55a45817	markere10bb1a4
iconUrl	Create a new map marker
latitude	0
Location	0.0
longitude	0
name	markere10bb1a4
place	
title	
New Marker	

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

The screenshot shows the Google Maps Control interface with the iconUrl property set to a local file path: C:\Samples\WebGoogleMapsControl\icon1.png. The marker is of type ROADMAP, with a count of 1. The location is set to 0.0.

Property	Value
GoogleMapsMapType	ROADMAP
Count	1
marker_55a45817	markere10bb1a4
iconUrl	C:\Samples\WebGoogleMapsControl\icon1.png
latitude	0
Location	0.0

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

The screenshot shows the Google Maps Control interface with the location set to 47.61658, -122.1992. The marker is of type ROADMAP, with a count of 1. The location is set to 47.61658, -122.1992. The latitude is 47.61658 and the longitude is -122.1992.

Property	Value
GoogleMapsMarkers	Count:1
marker_55a45817	m1
iconUrl	C:\Samples\WebGoogleMap:
latitude	47.61658
Location	47.61658, -122.1992
Latitude	47.61658
Longitude	-122.1992

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

The screenshot shows the Google Maps Control interface with the title set to "This is a test marker set at design time.". The marker is of type ROADMAP, with a count of 1. The location is set to 47.61658, -122.1992. The latitude is 47.61658 and the longitude is -122.1992. The name is m1.

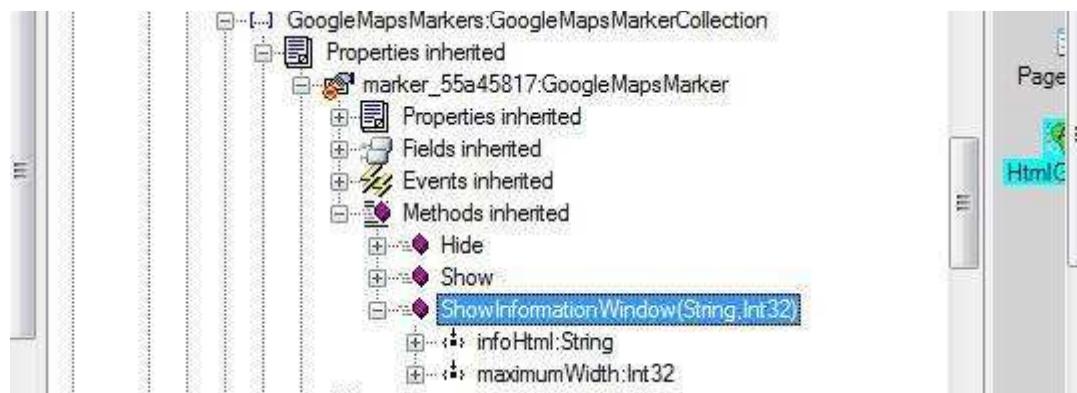
Property	Value
GoogleMapsMarkers	Count:1
marker_55a45817	m1
iconUrl	C:\Samples\WebGoogleMapsControl\icon1.png
latitude	47.61658
Location	47.61658, -122.1992
Latitude	47.61658
Longitude	-122.1992
longitude	-122.1992
name	m1
place	
title	This is a test marker set at design time.

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



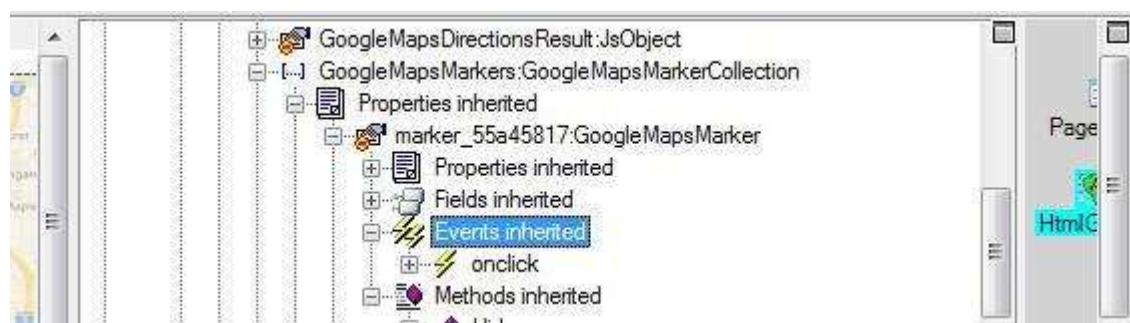
- markerColor** – You may give a color for the marker if iconUrl is empty

Marker Methods



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

Marker Events

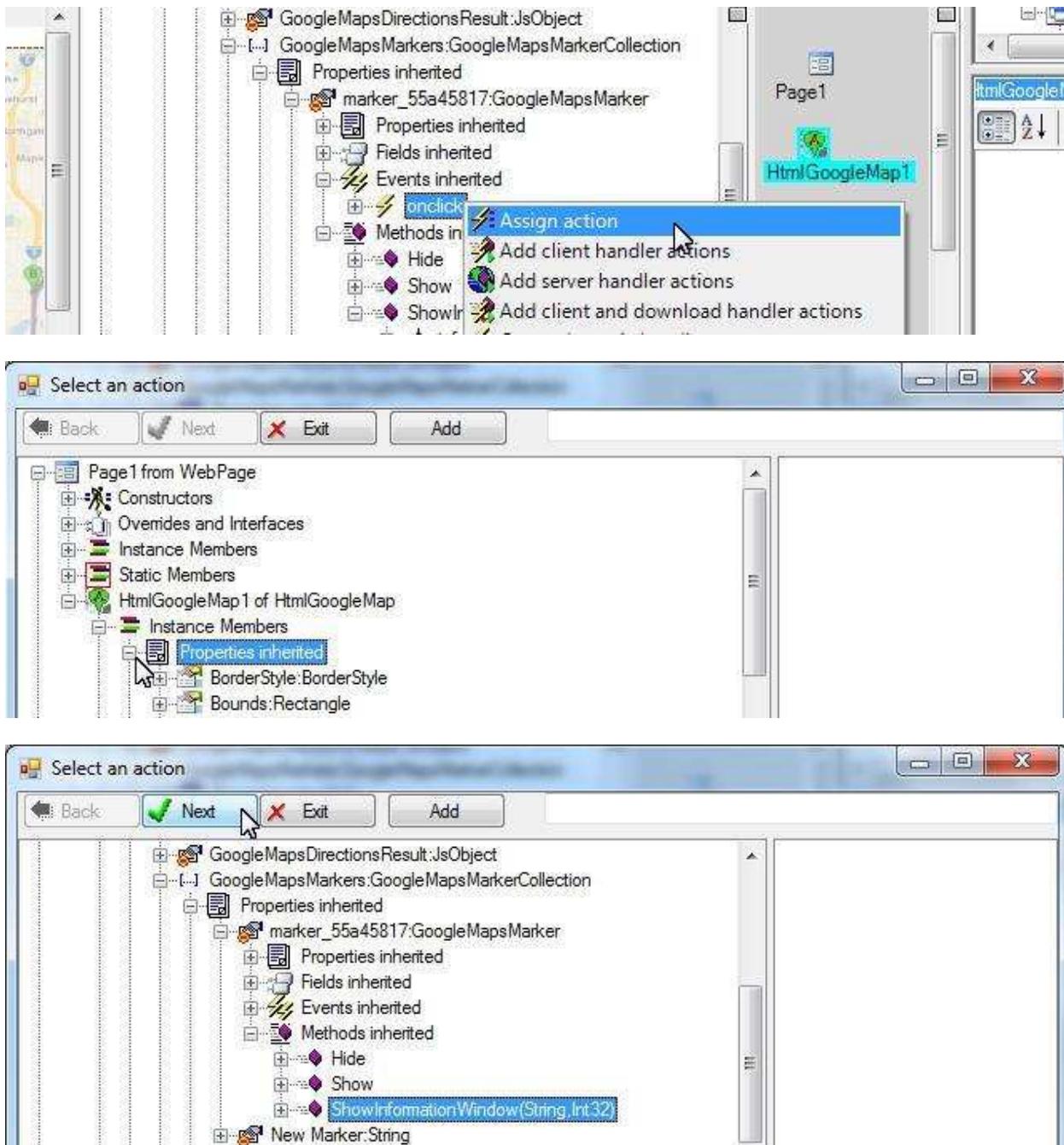


- onclick** – occurs when the marker is clicked.

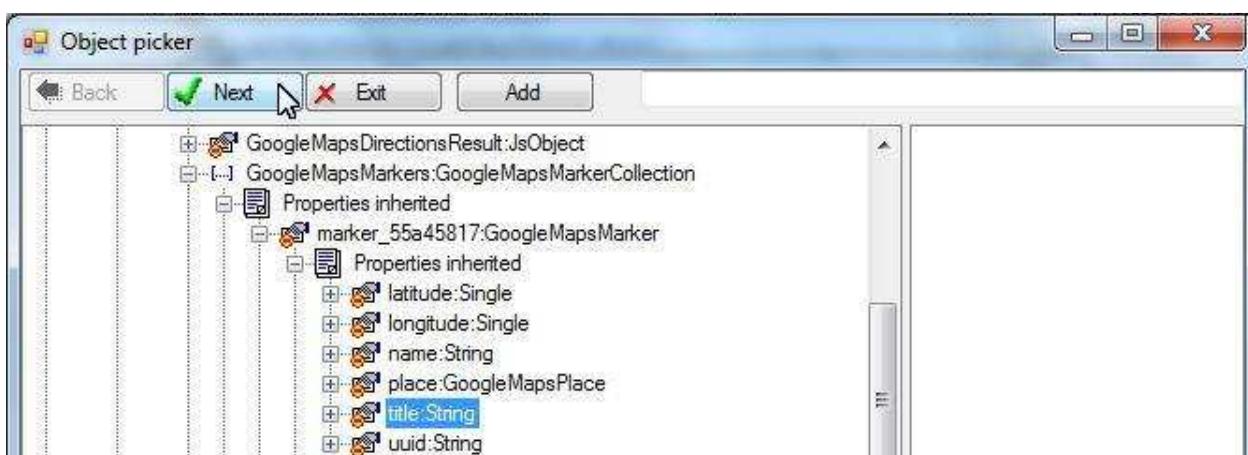
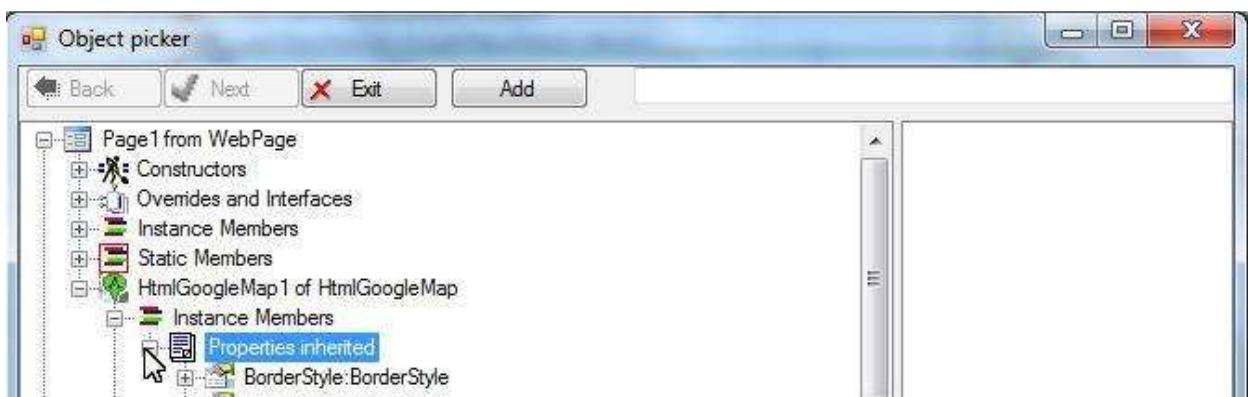
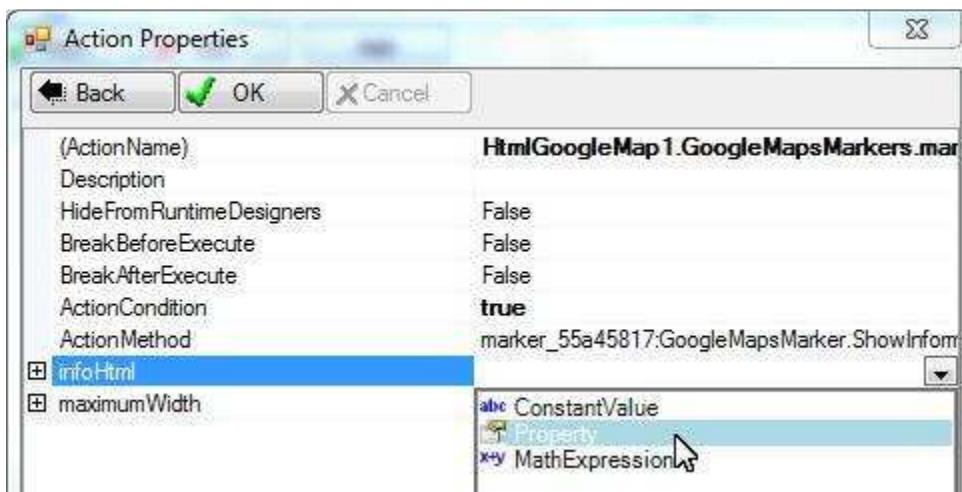
Handle marker click event at design time

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

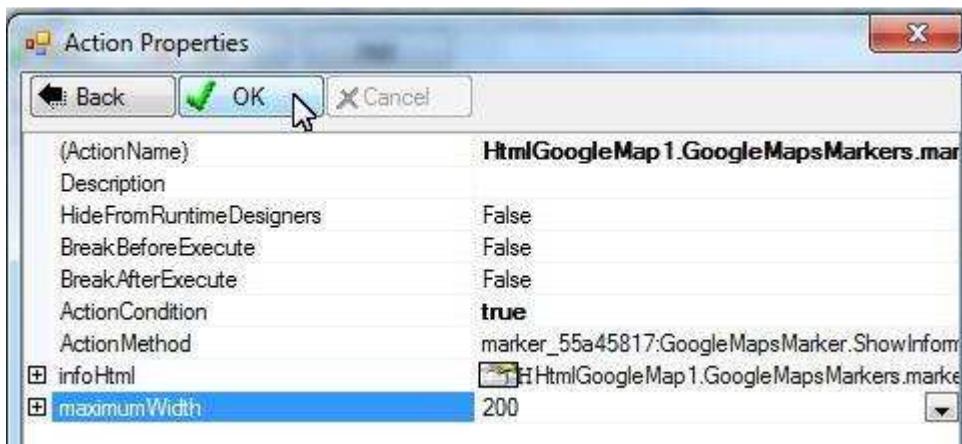
Suppose we want to display an information window when a marker is clicked. We may assign a ShowInformationWindow action to the onclick event.



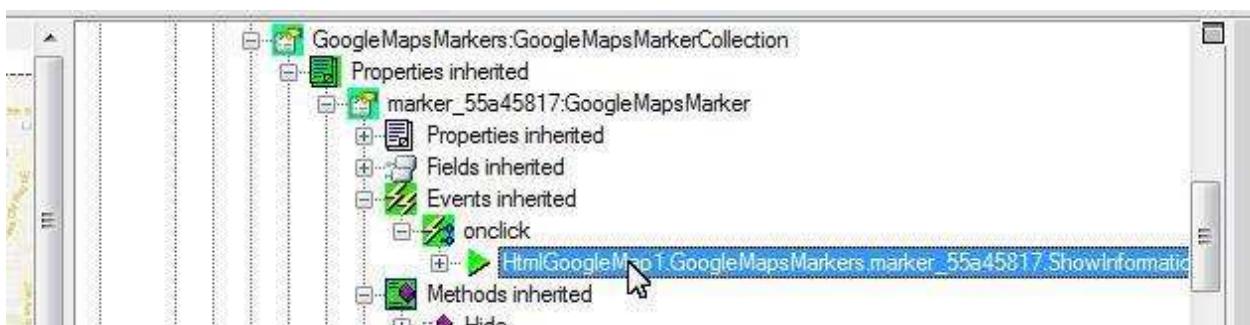
Let's show title of the marker:



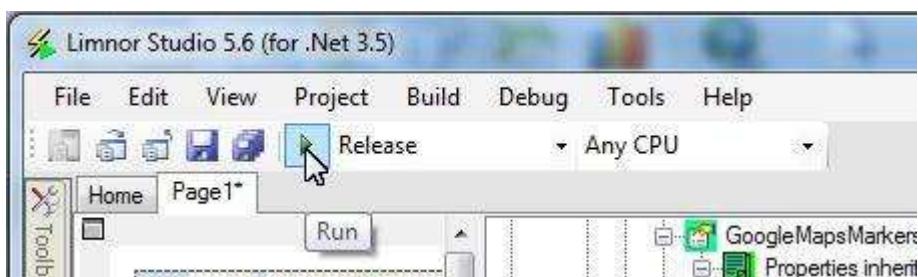
Set maximumWidth to 200. Click OK:



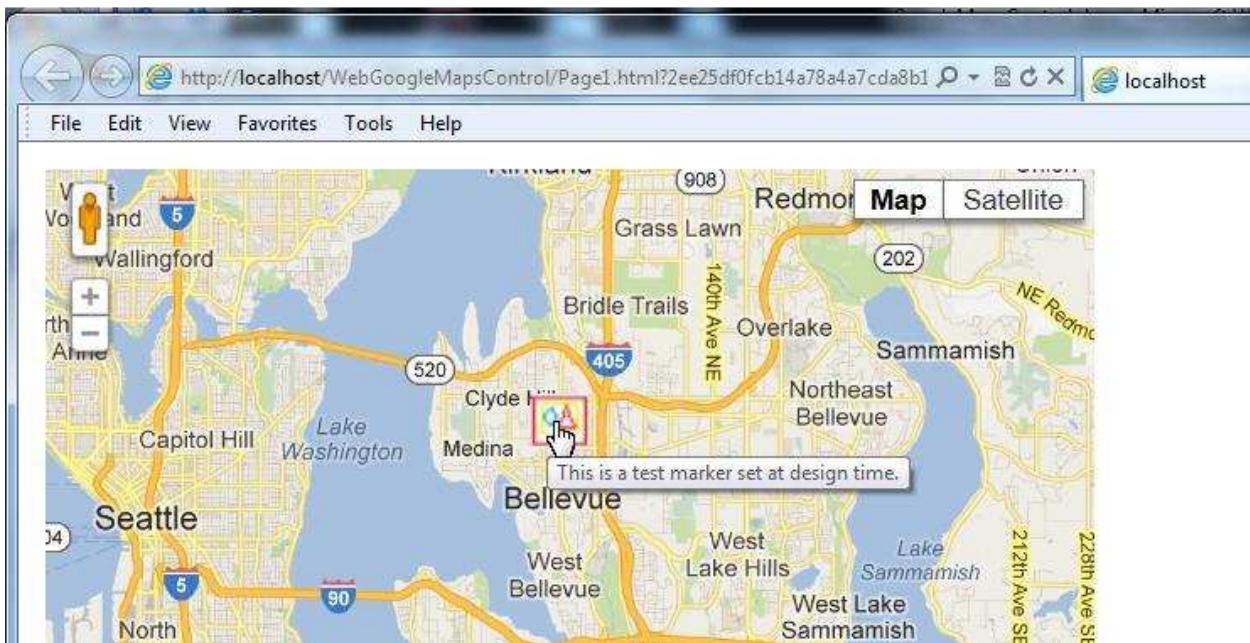
The action is created and assigned to the event:



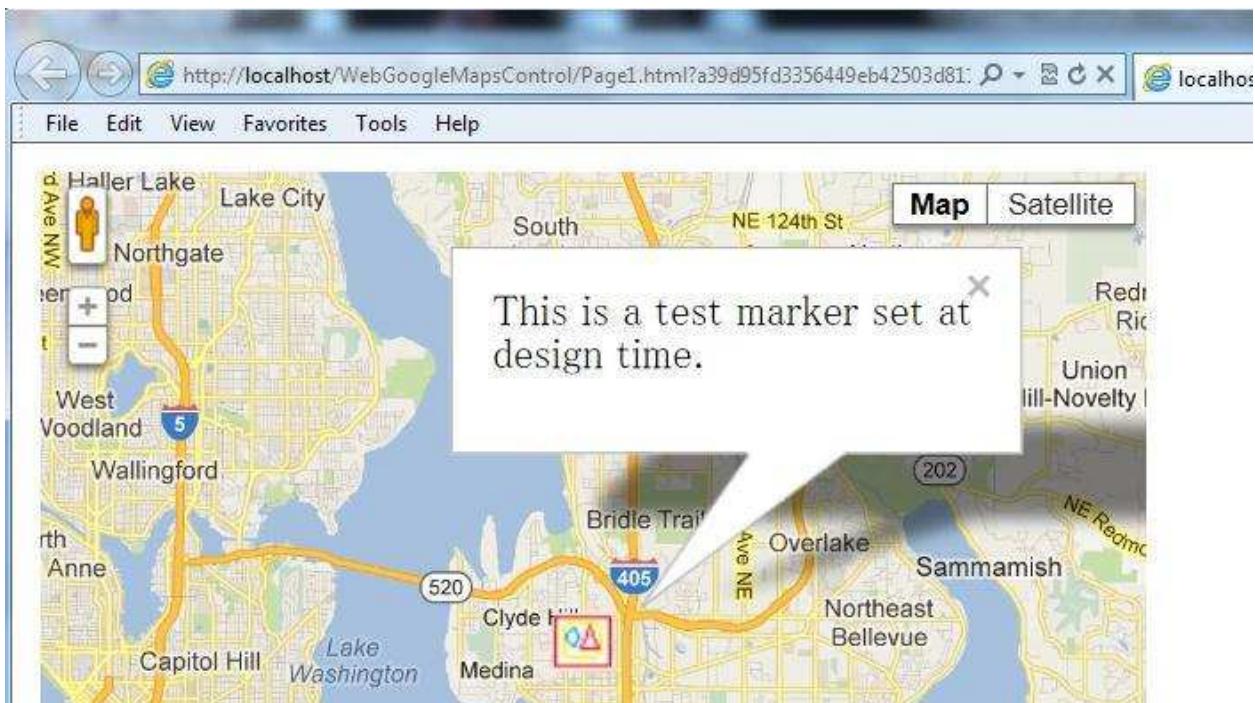
We may test the web page:



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:

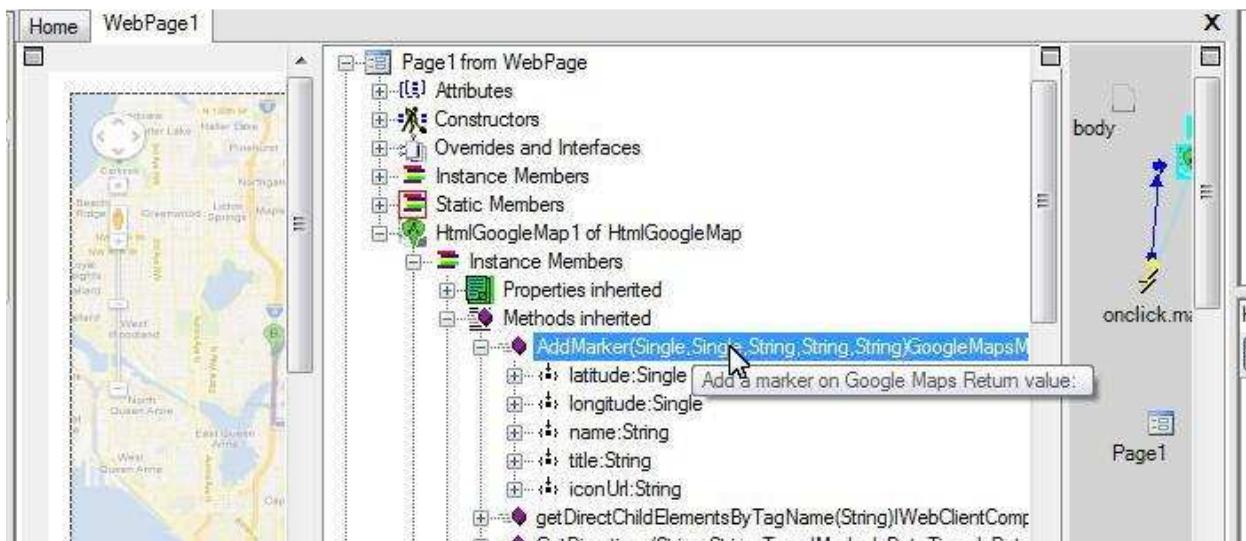


Click the marker, an information window appears:

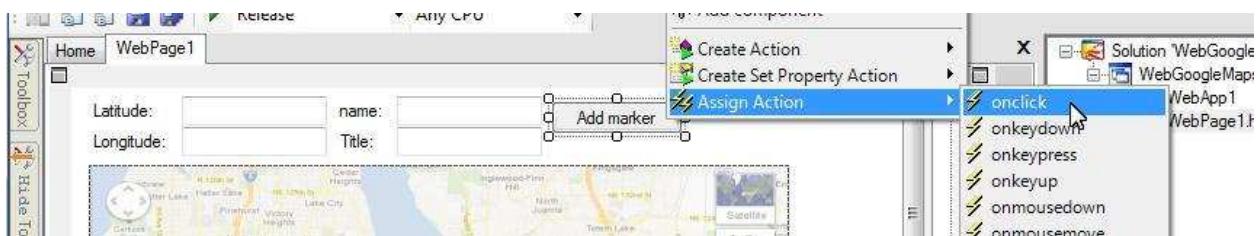


Add markers at runtime

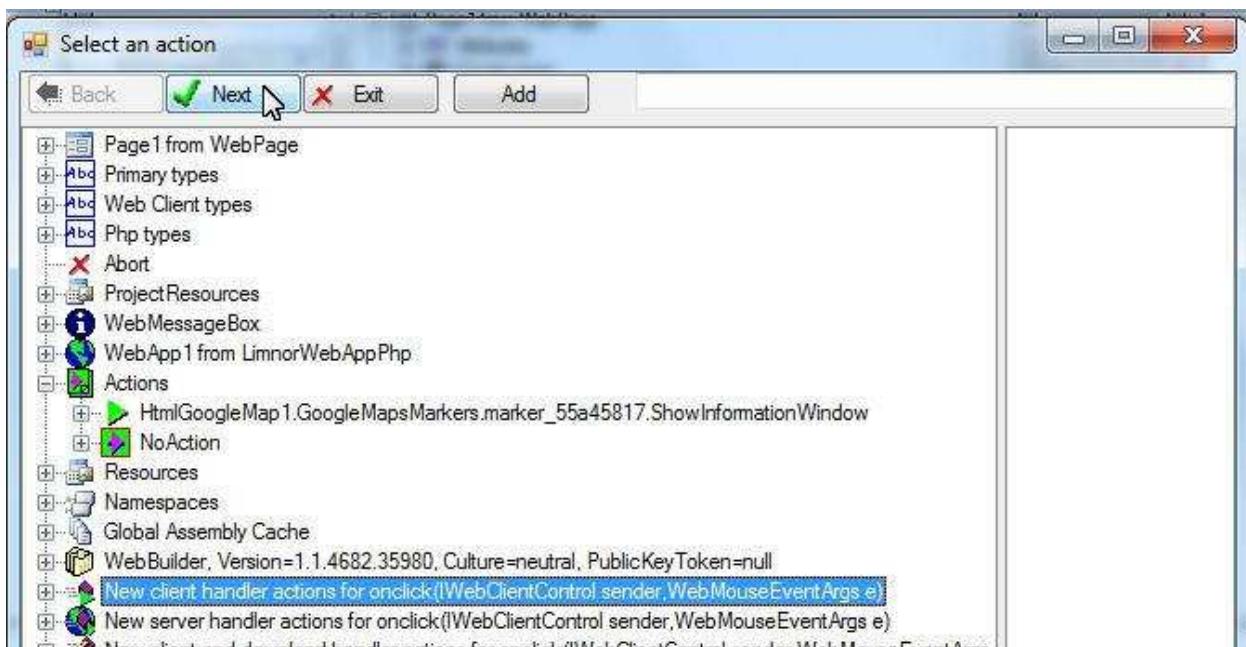
HtmlGoogleMap has an AddMarker for adding markers at runtime.



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

The screenshot shows the 'Method Designer' interface for handling an event. The top bar displays the title 'Method Designer - Handle event onclick by onbtAddMarkeronClick1'. Below the title is a toolbar with various icons for editing and navigating. The main area shows the object class 'Linner.WebBuilder.HtmlGoogleMap' and the method 'Main'. A context menu is open over the object 'HtmlGoogleMap1', with the path 'Create Action' > 'AddMarker(Single,Single,String,String)GoogleMapsMarker' highlighted.

Action Properties

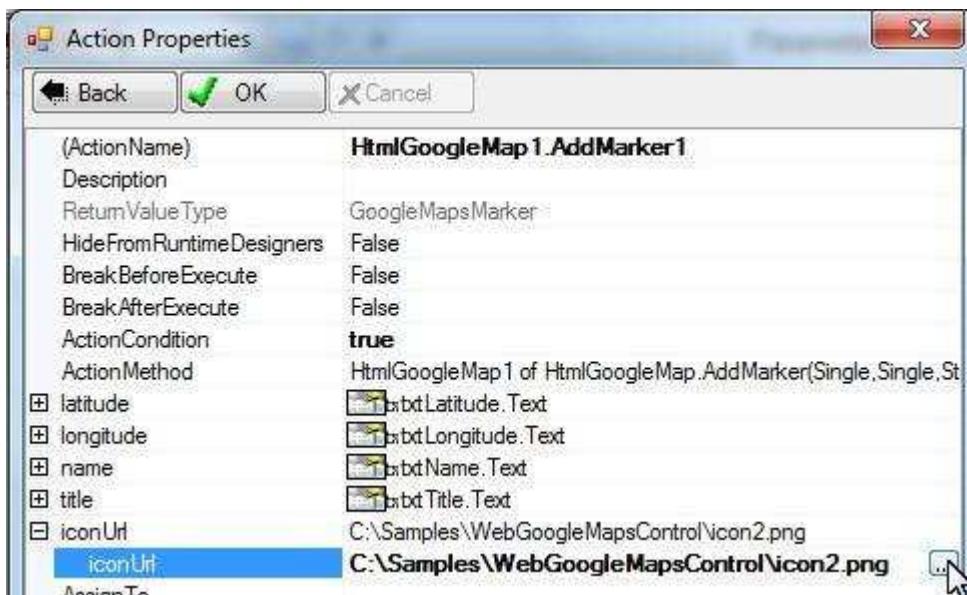
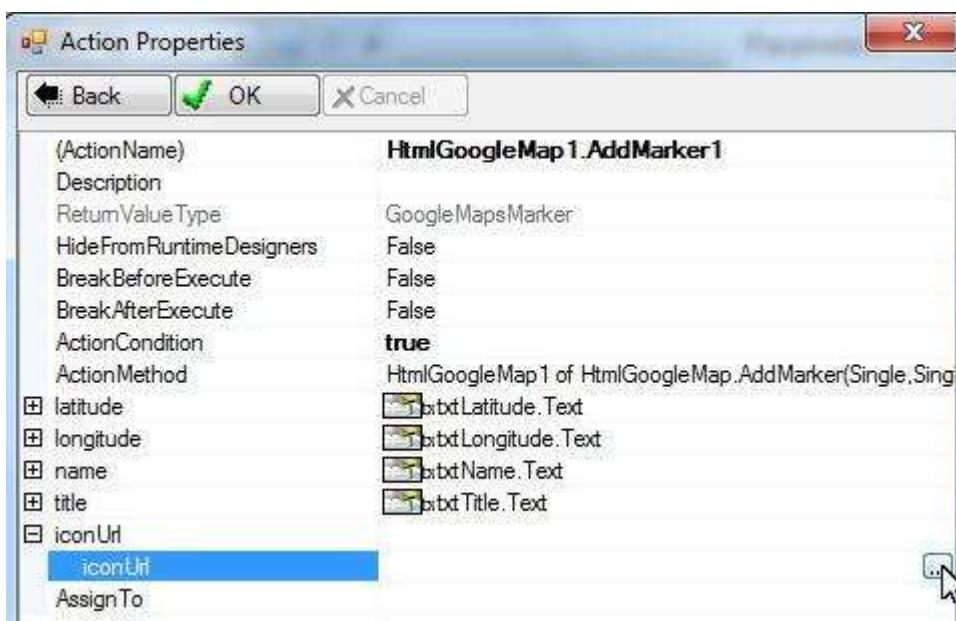
(ActionName)	
HtmlGoogleMap1.AddMarker1	
Description	GoogleMapsMarker
ReturnValueType	False
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	HtmlGoogleMap1 of HtmlGoogleMap.AddMarker(Single,Sing
+ latitude	0
+ longitude	abc ConstantValue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

Object picker

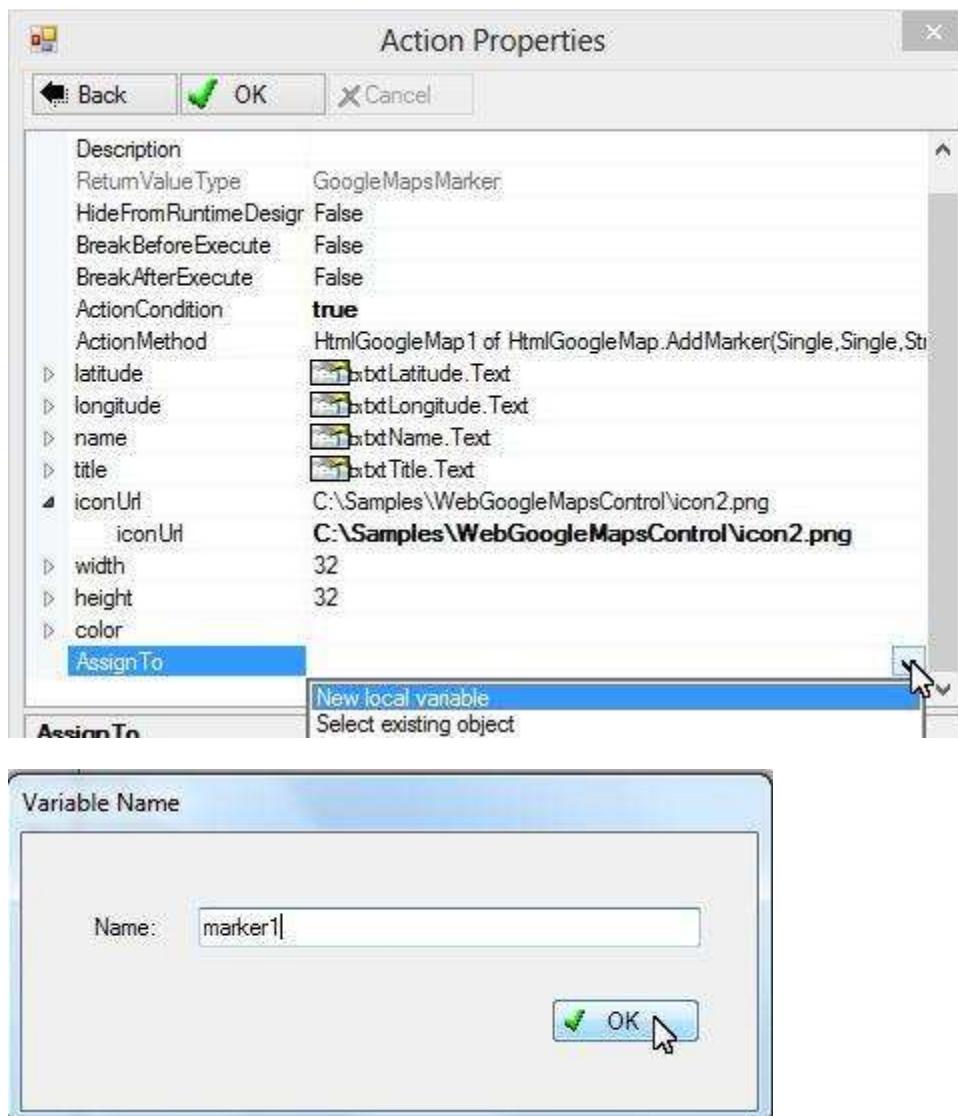
The 'Object picker' window displays the page members. The tree view shows 'Page1 from WebPage' expanded, revealing 'Constructors', 'Overrides and Interfaces', 'Instance Members', 'Static Members', and 'btAddMarker of HtmlButton'. Under 'Instance Members', 'HtmlGoogleMap1 of HtmlGoogleMap' is selected. Other members listed include 'HtmlLabel1 of HtmlLabel', 'HtmlLabel2 of HtmlLabel', 'HtmlLabel3 of HtmlLabel', 'HtmlLabel4 of HtmlLabel', and 'txtLatitude of HtmlTextBox'. The 'Properties inherited' section for 'txtLatitude' lists 'BackColor:Color' and 'DataBind:DataBind'.



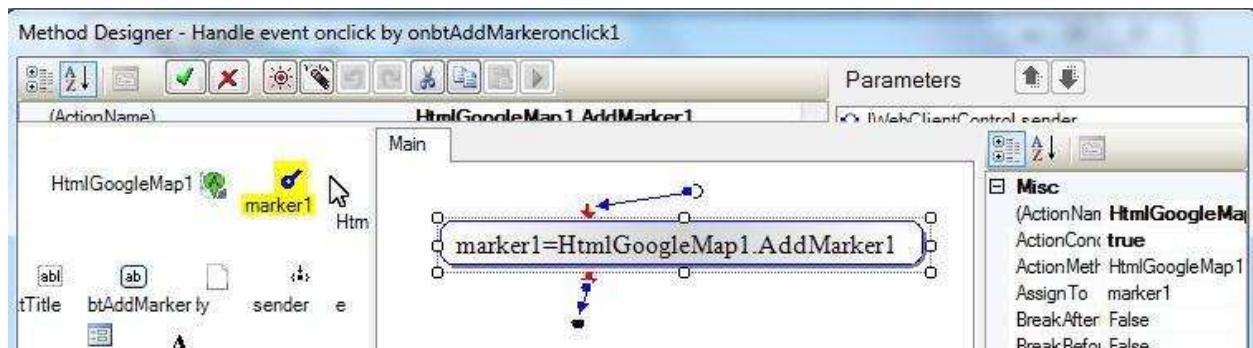
We may pass other text box values to the action. We may also select an image file for iconUrl:



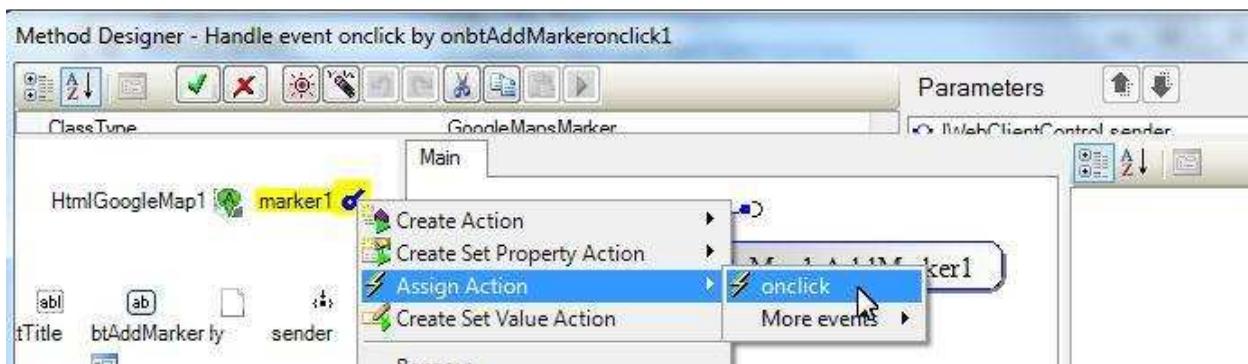
Set "AssignTo" to create a new local variable:



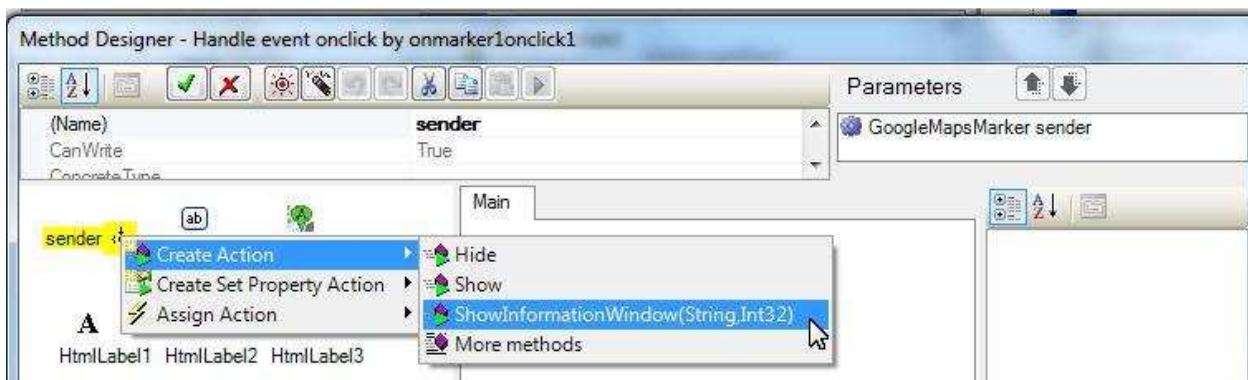
Click OK. The action is created. Note that variable marker1 is the marker added to web page:



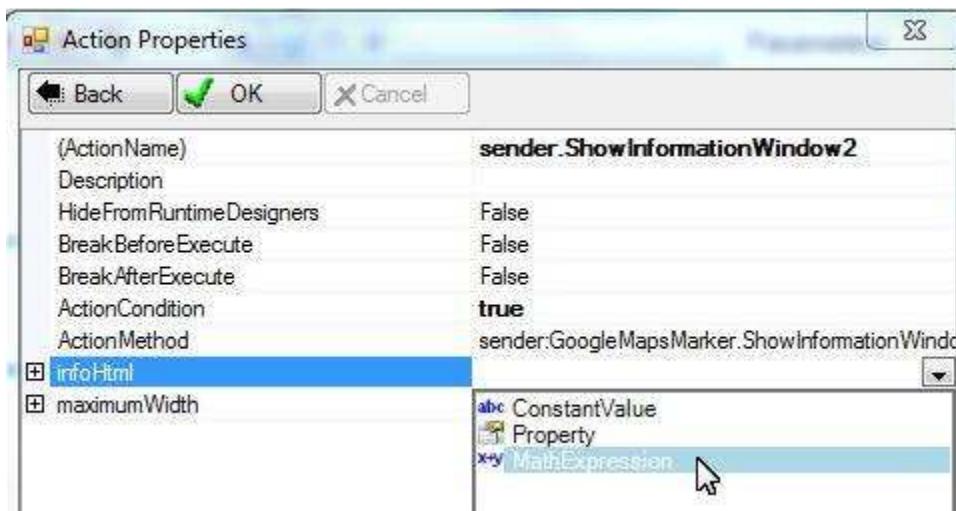
We may assign actions to onclick event of the newly created marker:

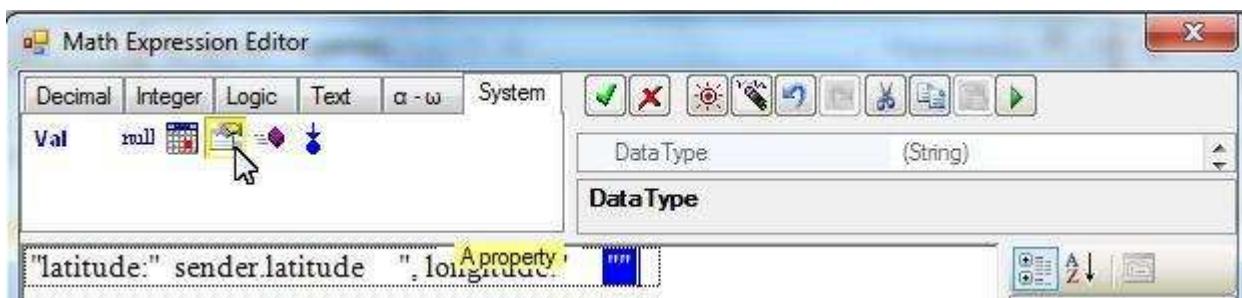
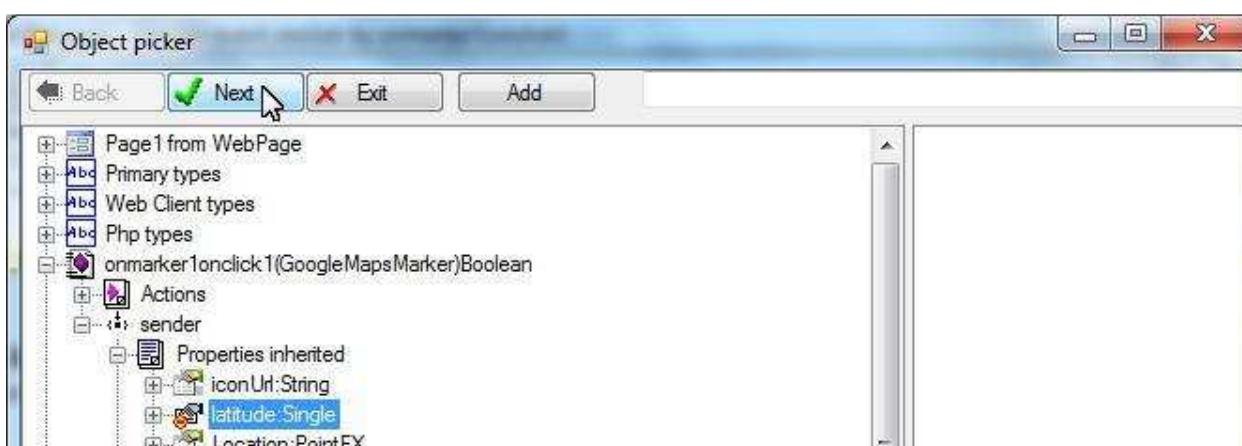
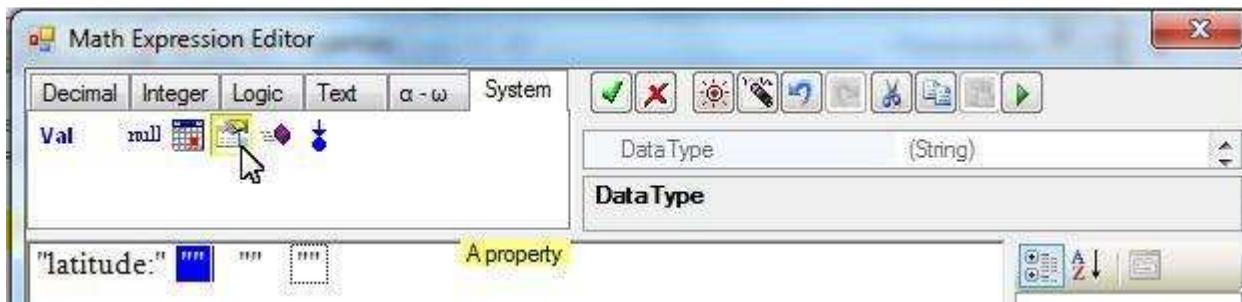


A method editor appears to let you add actions to be executed when onclick event occurs. Note that "sender" is the marker object which generated the onclick event. Suppose we want to show an information window at this event:



Suppose we want to show the location of the marker:





Object picker

```

+ Page1 from WebPage
+ Abc Primary types
+ Abc Web Client types
+ Abc Php types
- onmarker1onclick1(GoogleMapsMarker)Boolean
  Actions
  sender
    Properties inherited
      iconUrl:String
      latitude:Single
      Location:PointFX
      longitude:Single
      name:String
  
```

Object picker

```

+ Page1 from WebPage
+ Abc Primary types
+ Abc Web Client types
+ Abc Php types
- onmarker1onclick1(GoogleMapsMarker)Boolean
  Actions
  sender
    Properties inherited
      iconUrl:String
      latitude:Single
      Location:PointFX
      longitude:Single
      name:String
  
```

Math Expression Editor

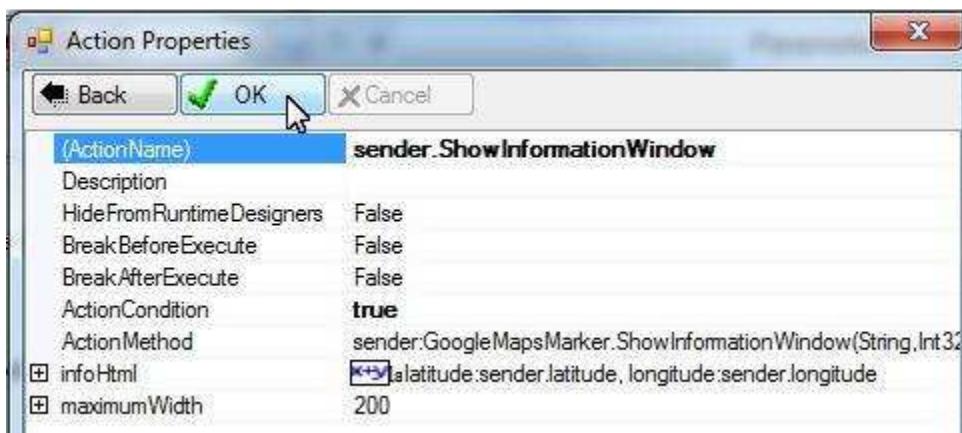
Finish the editing.

Val null

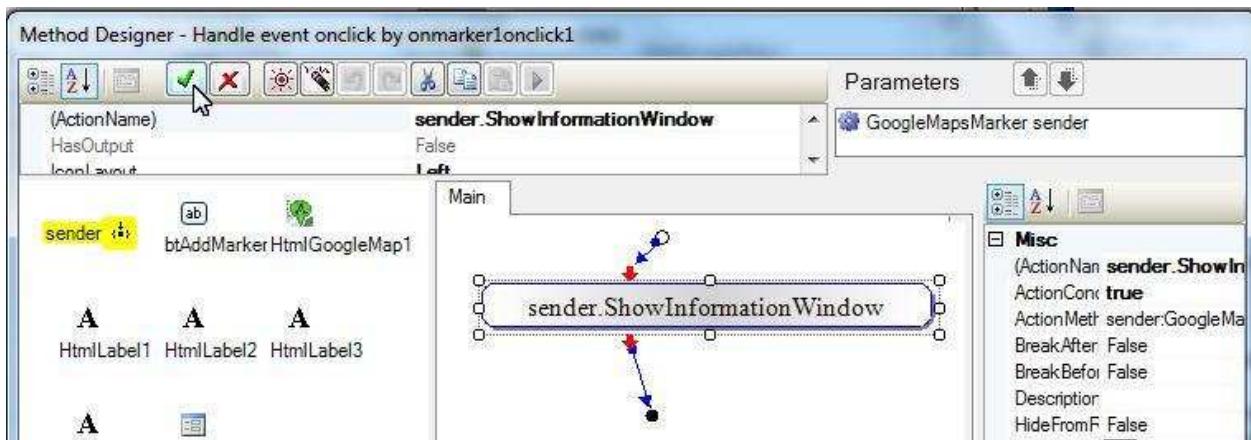
Data Type (Double)

DataType

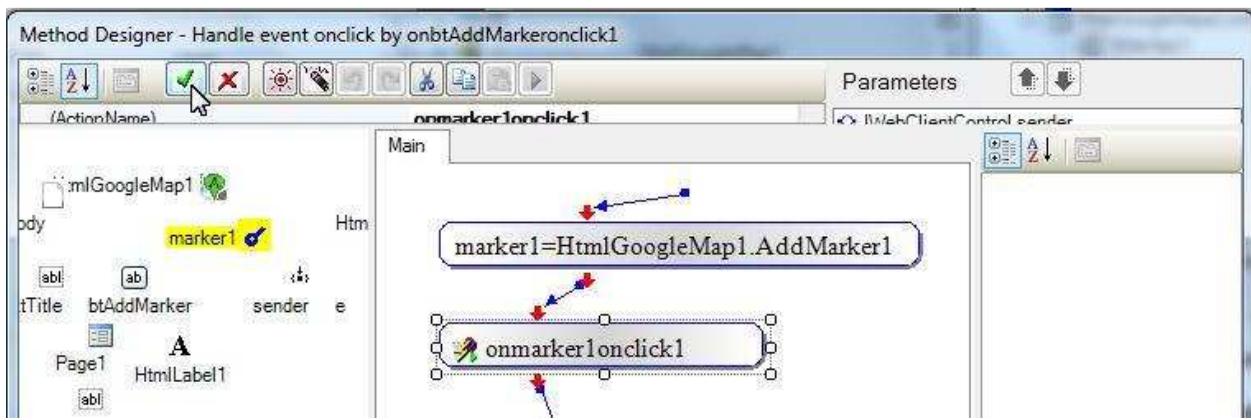
"latitude:" sender.latitude ",longitude:" **sender.longitude**



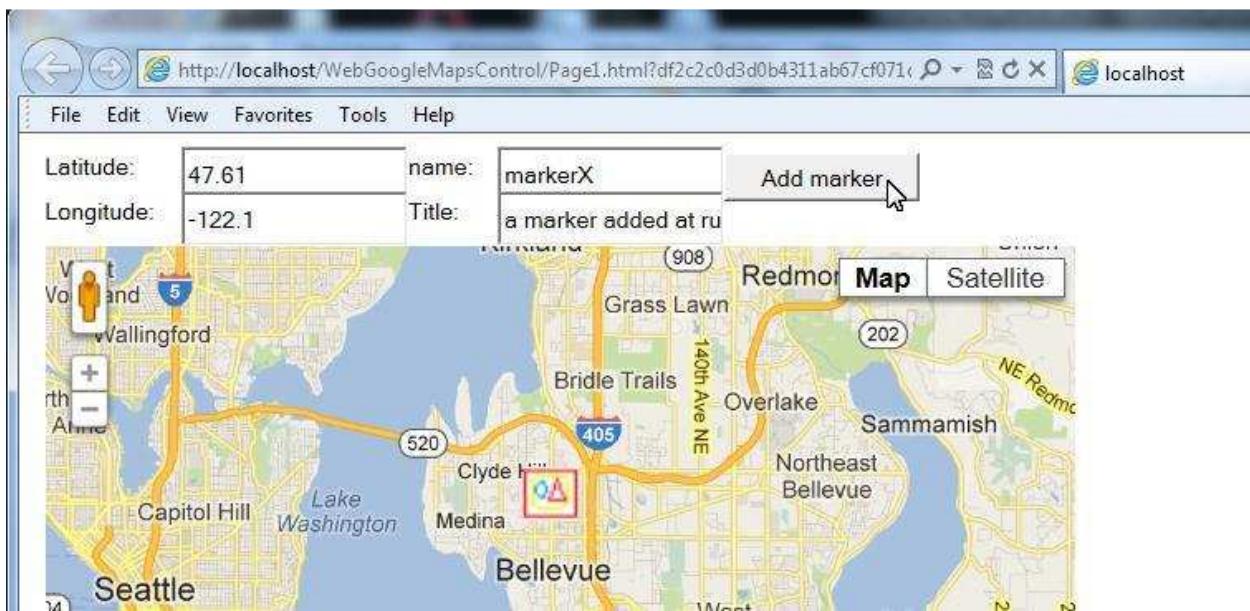
For this sample, we just use this one action:



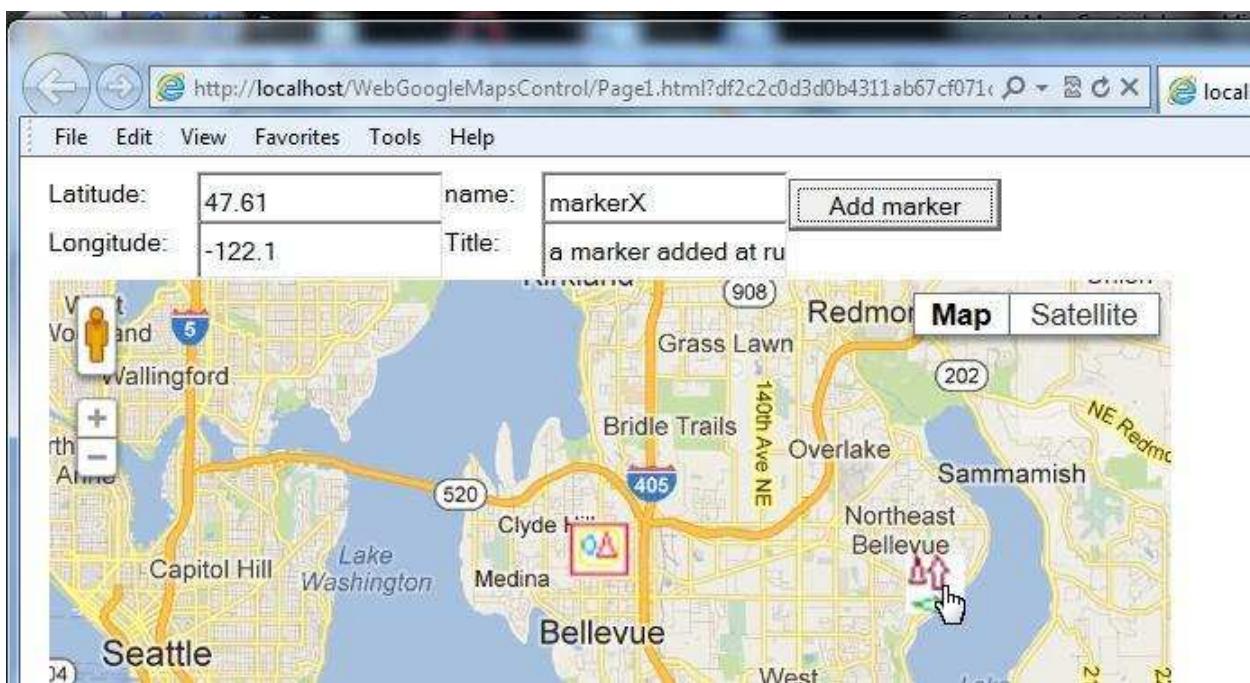
For this sample, that is all we need to manipulate the newly created marker:



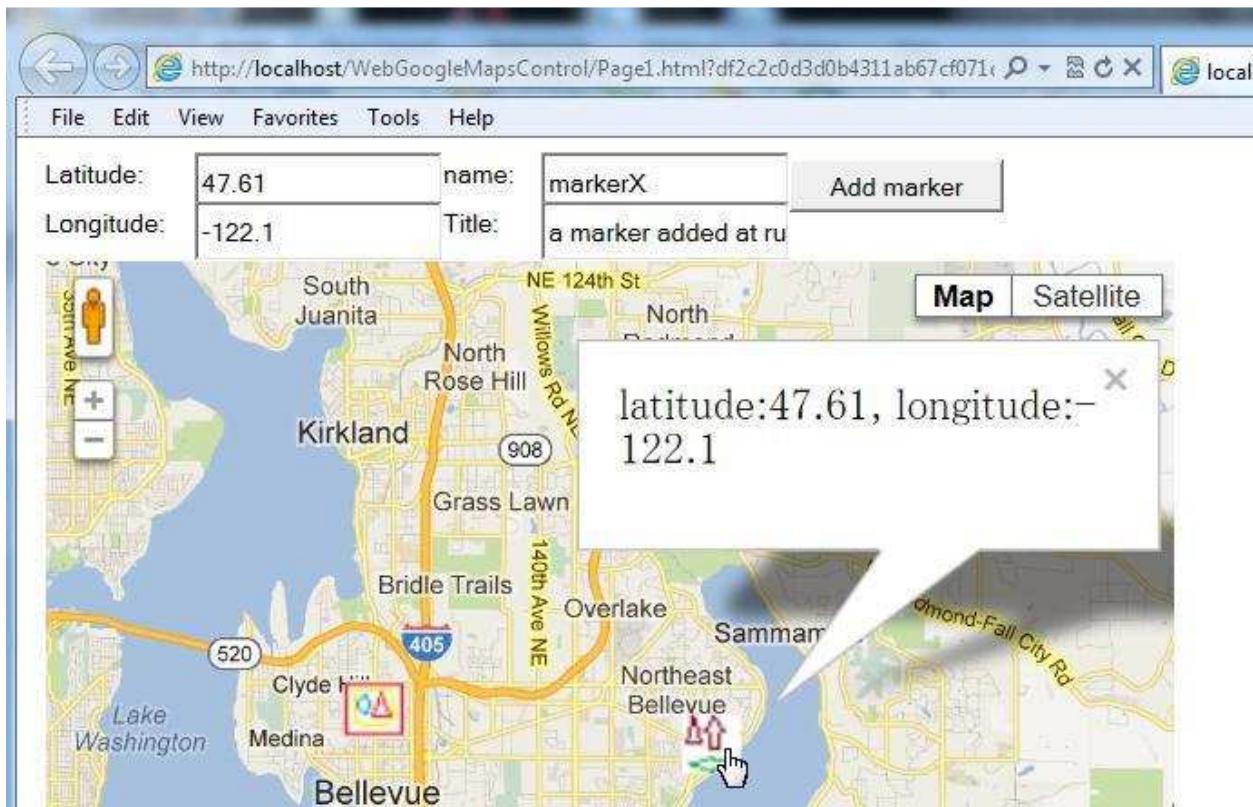
Let's launch the web page to try it. Enter parameters for a new marker. Click "Add marker":



A new marker appears. Click on the new marker:



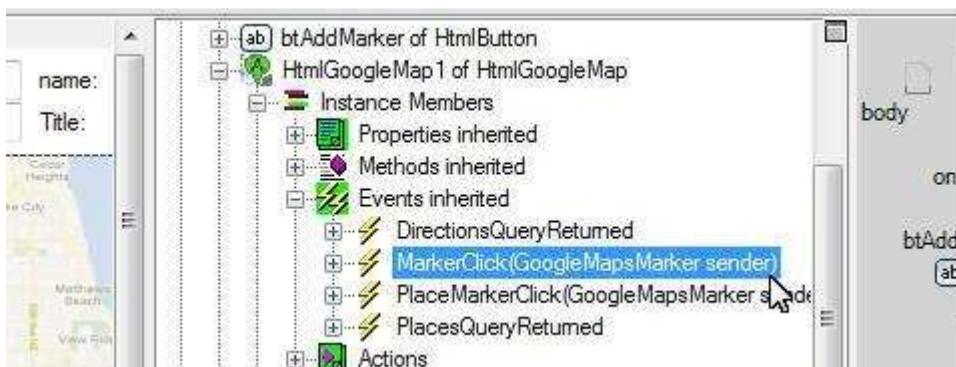
An information window appears showing marker location:



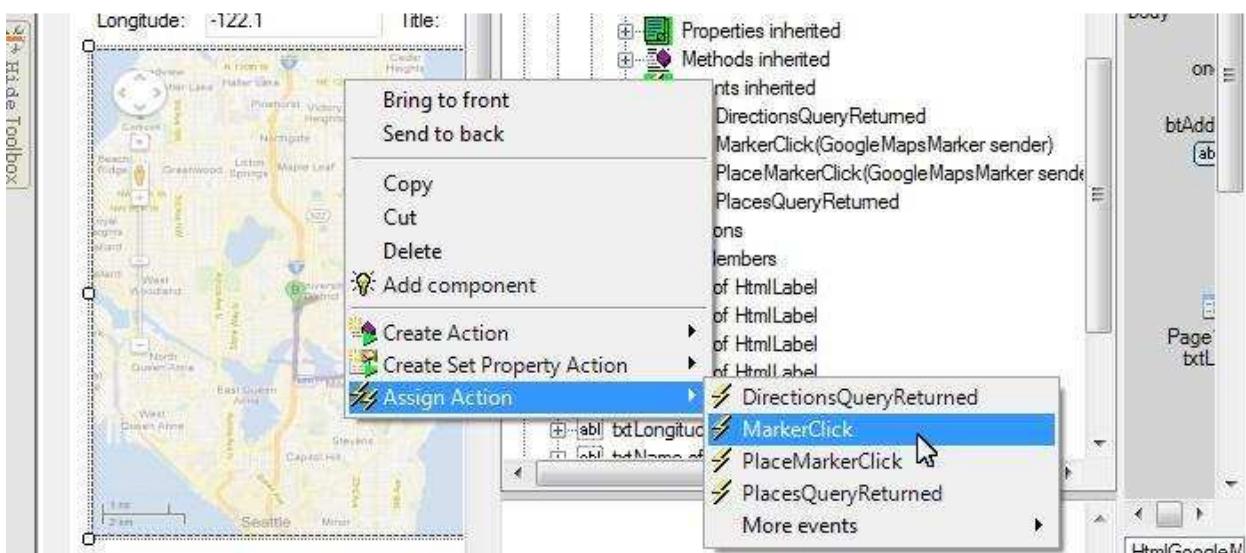
Handle event for all markers

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

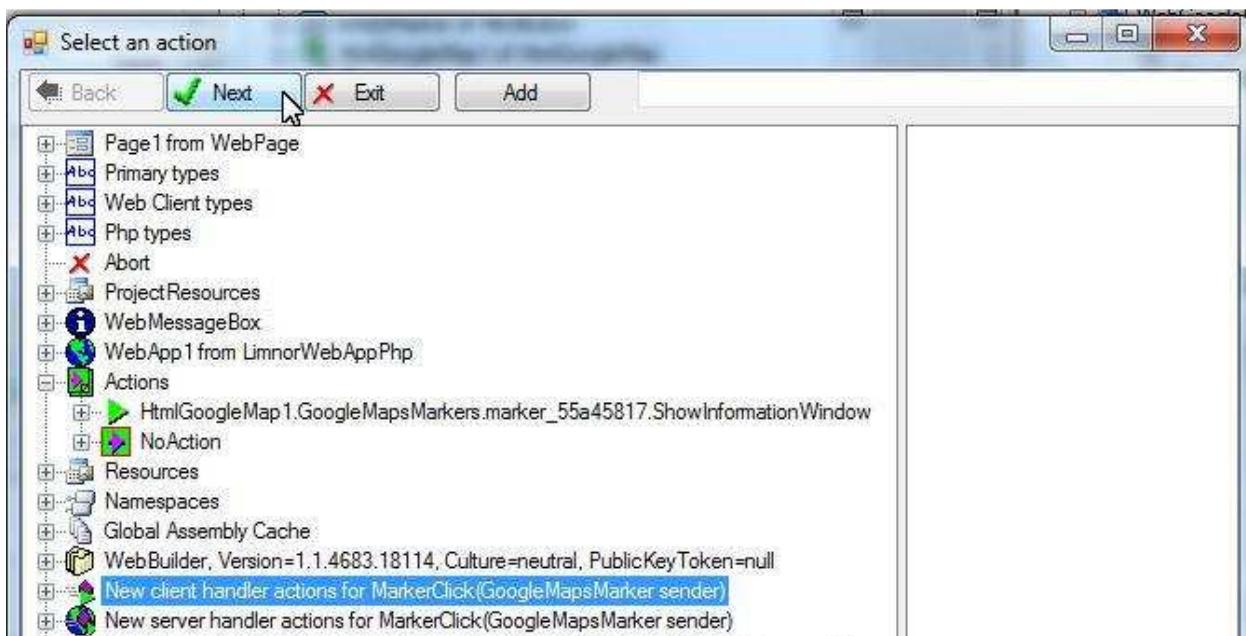
If your event handling logic is similar for all markers then you do not need to add event handling logic to each marker one by one. `HtmlGoogleMap` has an event, `MarkerClick`, which occurs when any marker is clicked on.



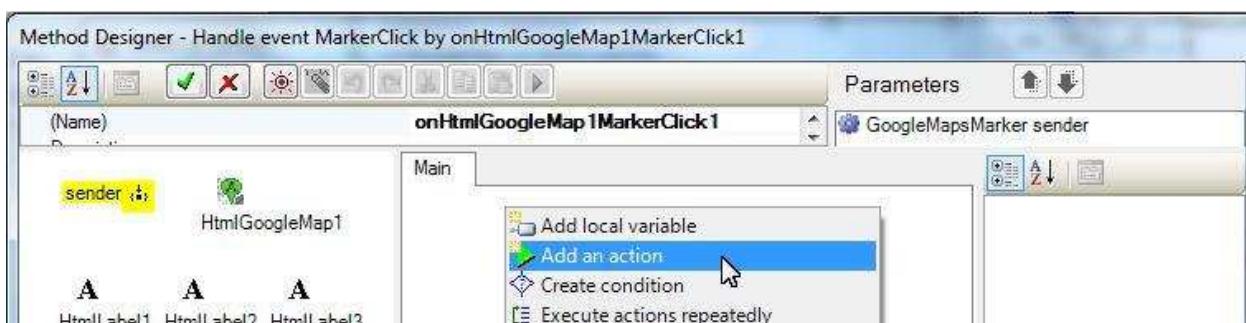
Suppose we want to show a message box displaying marker name and title when any marker is clicked:



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
- Web MessageBox
 - 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	<input type="button" value="abc ConstantValue"/> <input type="button" value="Property"/> <input style="background-color: #0070C0; color: white; font-weight: bold;" type="button" value="MathExpression"/>

Math Expression Editor

Decimal Integer Logic Text α - ω System

0x LF CR CL ↵ null () A? ?A > ≥ <
≤ ≠ ≠ !!! var Val null A+

Concatenate two strings together to form a new string

Math Expression Editor

Decimal Integer Logic Text α - ω System

Val null

DataType (String)

"name:" A property

The screenshot shows two overlapping windows. The top window is titled "Object picker" and displays a tree view of objects. It includes a toolbar with Back, Next, Exit, and Add buttons. The tree view shows:

- Page1 from WebPage
- Primary types
- Web Client types
- PHP types
- onHtmlGoogleMap1MarkerClick 1(GoogleMapsMarker)Boolean
 - Actions
 - sender
 - Properties inherited
 - iconUrl:String
 - latitude:Single
 - Location:PointFX
 - longitude:Single
 - name String**
 - place:GoogleMapsPlace
 - title:String

The "name String" property is highlighted in blue.

The bottom window is titled "Math Expression Editor" and shows a toolbar with buttons for Decimal, Integer, Logic, Text, α - w, System, and a set of icons. Below the toolbar, there are dropdown menus for "Val" and "null". A text input field contains the expression: "name:" sender.name ", title:". A tooltip "A property" points to the "name" part of the expression. To the right of the input field is a "DataType" dropdown set to "(String)".

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

The screenshot shows the "Object picker" window again. The left pane lists objects: Page1 from WebPage, Primary types, and Web Client types. The right pane shows a list of recent selections, with "GoogleMapsMarker sender.name" highlighted and a cursor pointing at it.

Object picker

The Object picker dialog shows the following structure:

- Page1 from WebPage
- Primary types
- Web Client types
- Php types
- onHtmlGoogleMap1MarkerClick (GoogleMapsMarker) Boolean
- Actions
- sender
- Properties inherited
 - iconUrl: String
 - latitude: Single
 - Location: PointFX
 - longitude: Single
 - name: String
 - place: GoogleMapsPlace
 - title: String
 - uid: String

Math Expression Editor

The Math Expression Editor displays the expression:

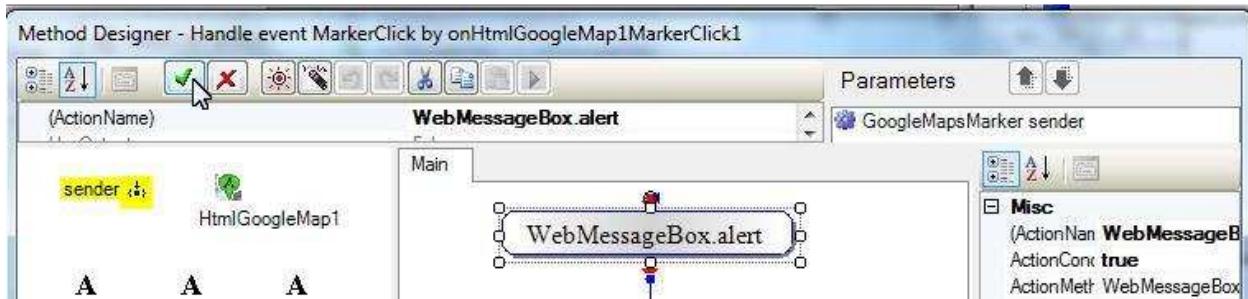
```
"name:" sender.name ", title:" sender.title
```

Action Properties

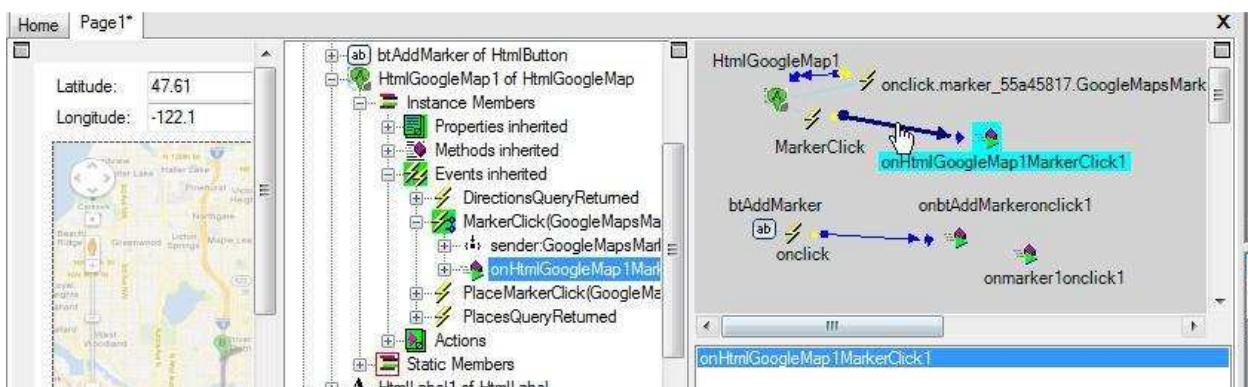
The Action Properties dialog shows the following settings for the 'WebMessageBox.alert' action:

(ActionName)	WebMessageBox.alert
Description	
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	WebMessageBox.alert(String)
message	a:name:sender.name,title:sender.title

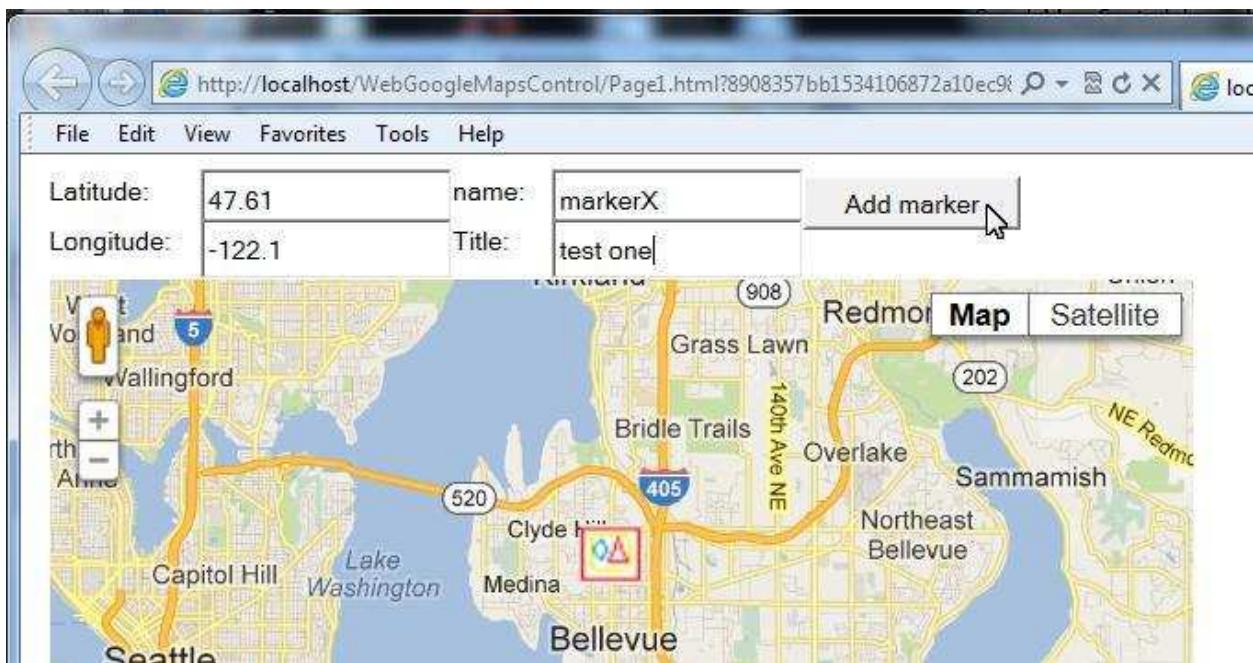
For this sample, we just use this one action on this event:



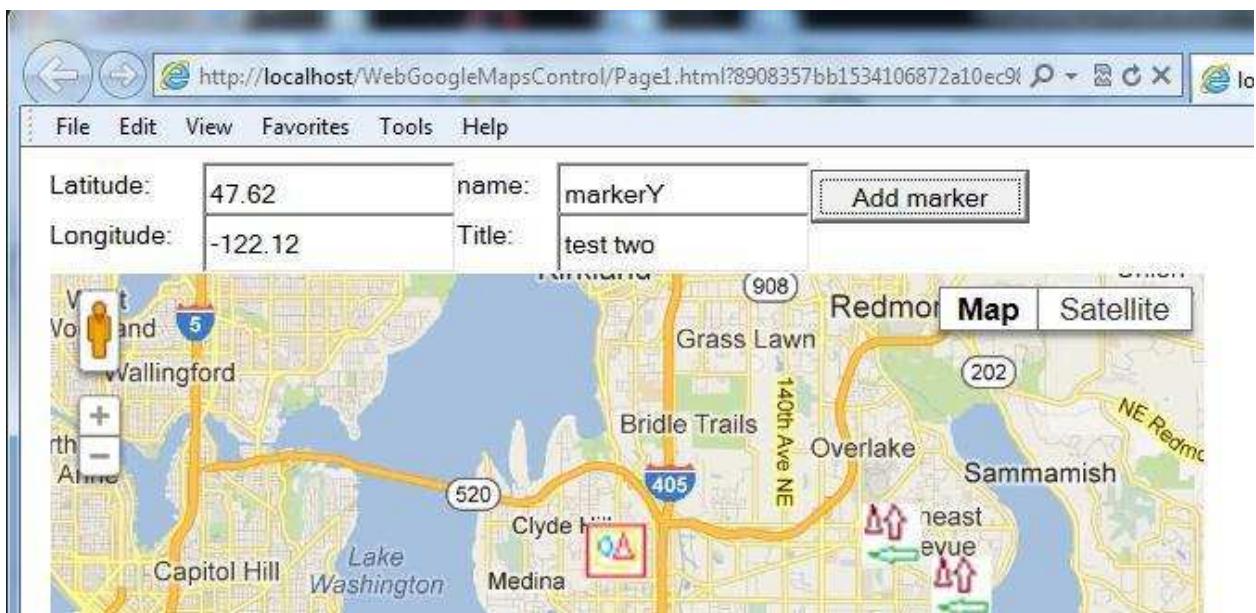
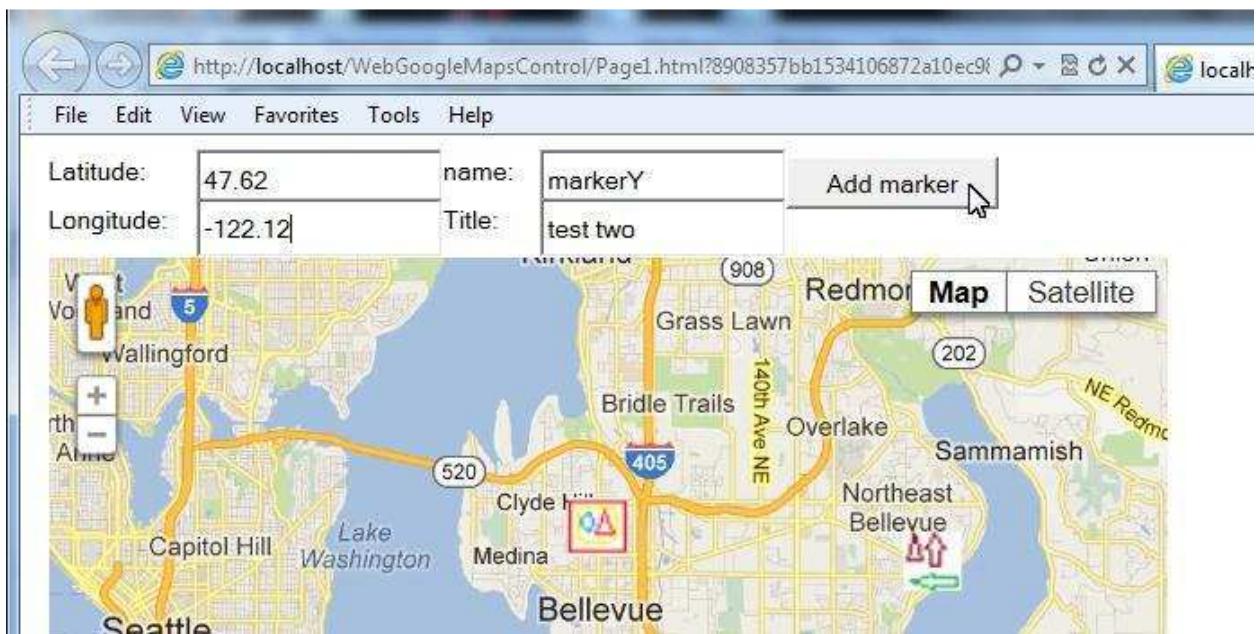
The event handler method is created:



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:

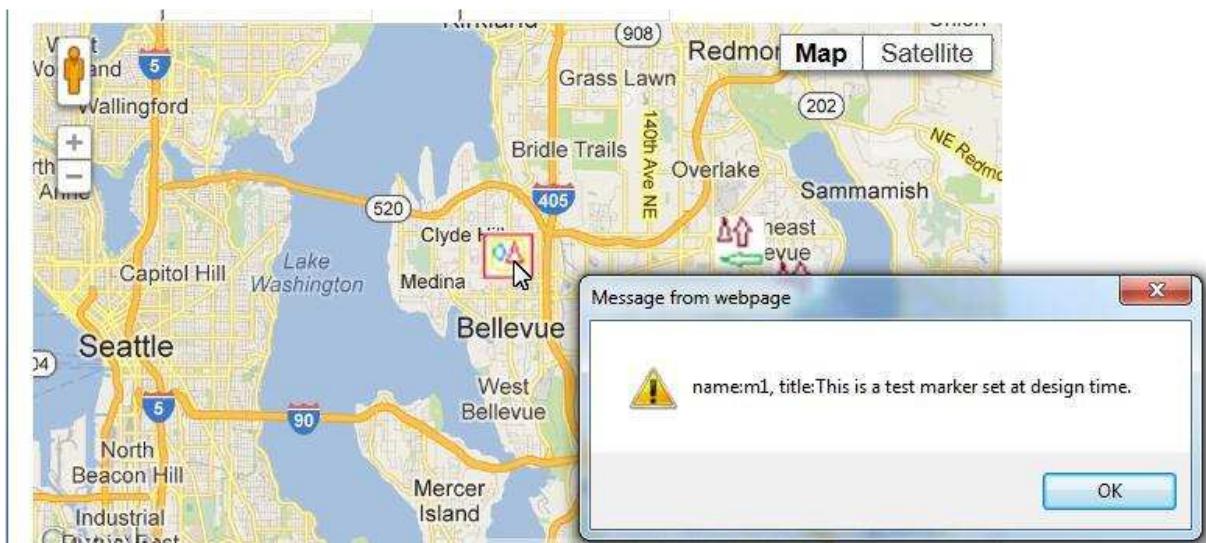


Google Maps Control | 2012



Click on each marker to see what will happen.

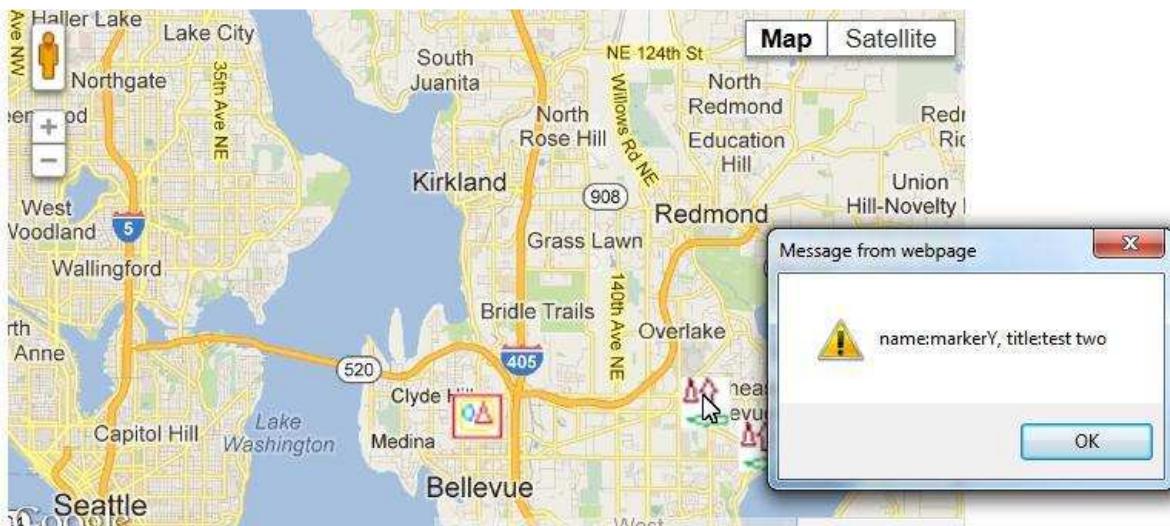
Click on the marker added at design time. A message box appears showing name and title of the marker, as the result of handling MarkerClick event of HtmlGoogleMap control:



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

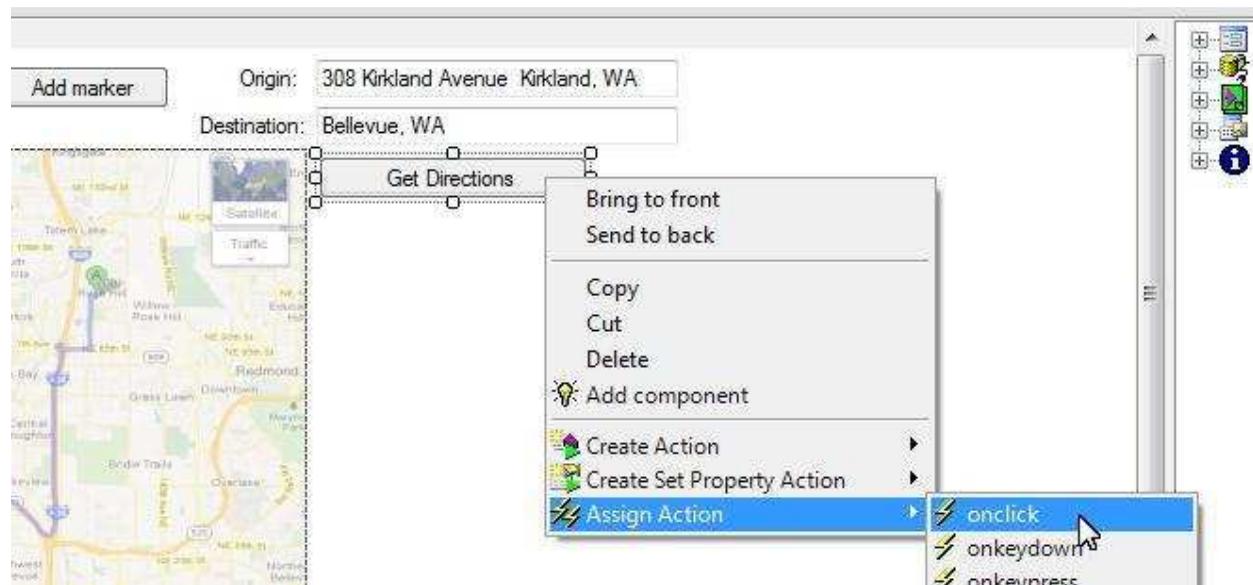
Get Directions

An HtmlGoogleMap control uses following procedure to get directions.

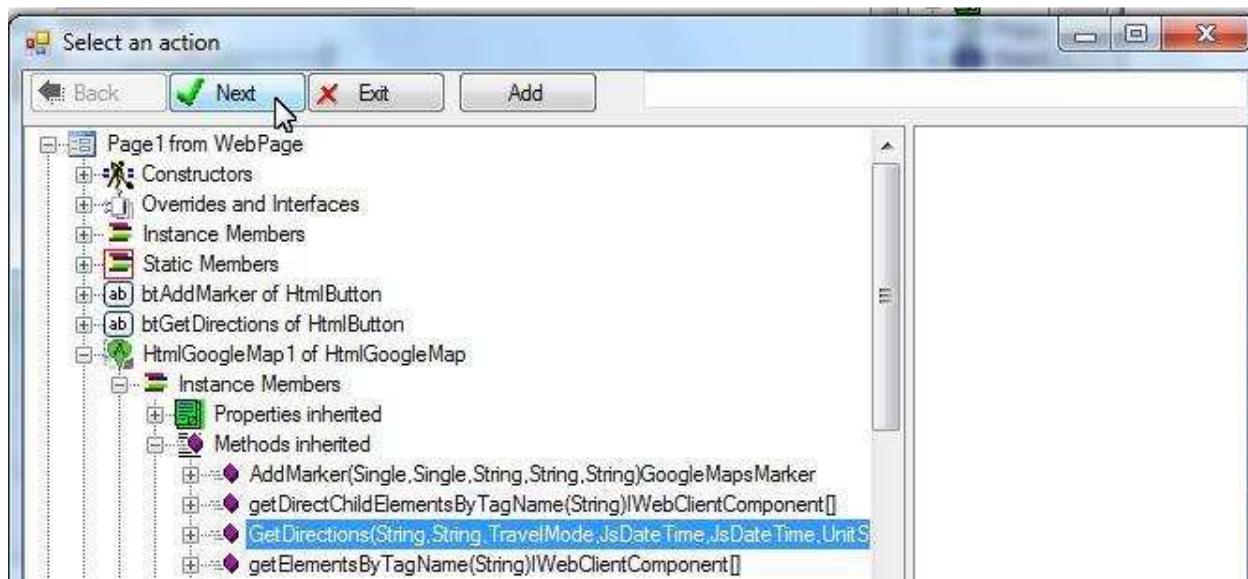
- A `GetDirections` action requests directions from Google Maps
- `DirectionsQueryReturned` event occurs when Google Maps responds to the request.
- `GoogleMapsDirectionsQuerytStatus` property indicates how Google Maps responded to the request.
- `GoogleMapsDirectionsResult` property contains all routes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

Create GetDirections action

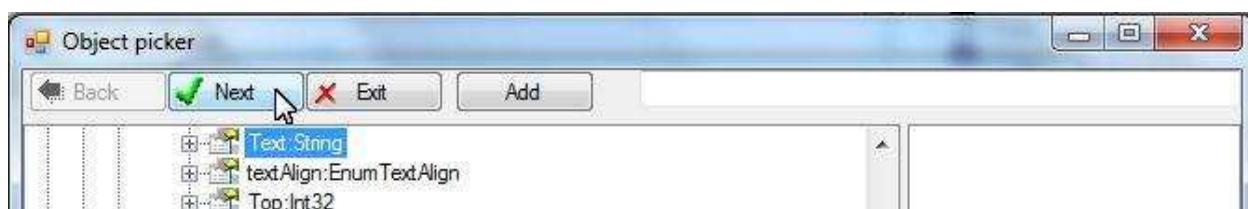
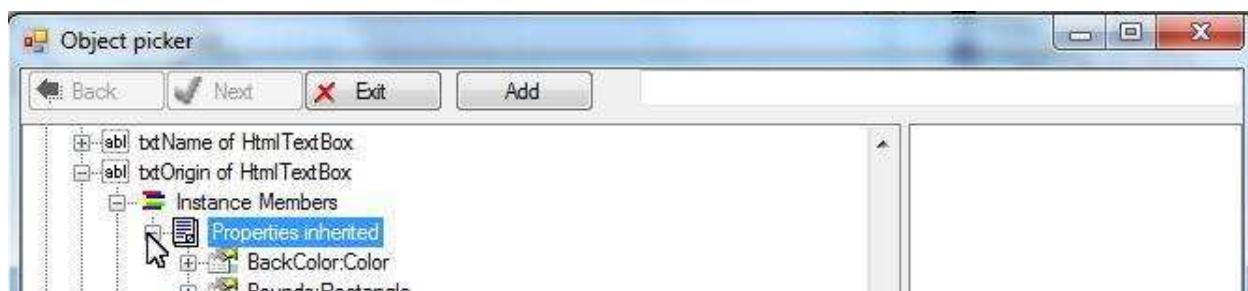
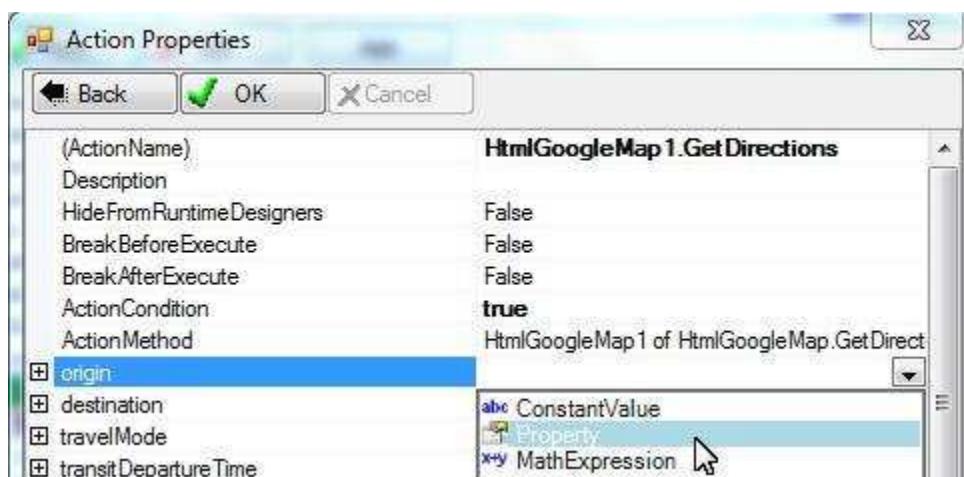
We use a button to initiate a get-directions action:



Select `GetDirections` method:



Use text box values for origin and destination:



Action Properties

Action Properties	
<input type="button" value="Back"/>	<input checked="" type="button" value="OK"/>
<input type="button" value="Cancel"/>	
HtmlGoogleMap1.GetDirections	
(ActionName)	
Description	
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	HtmlGoogleMap1 of HtmlGoogleMap.GetDirect
origin	<input type="button" value="txtOrigin.Text"/>
destination	<input checked="" type="button" value="abc ConstantValue"/>
travelMode	<input type="button" value="Property"/>
transitDepartureTime	<input type="button" value="MathExpression"/>
transitArrivalTime	

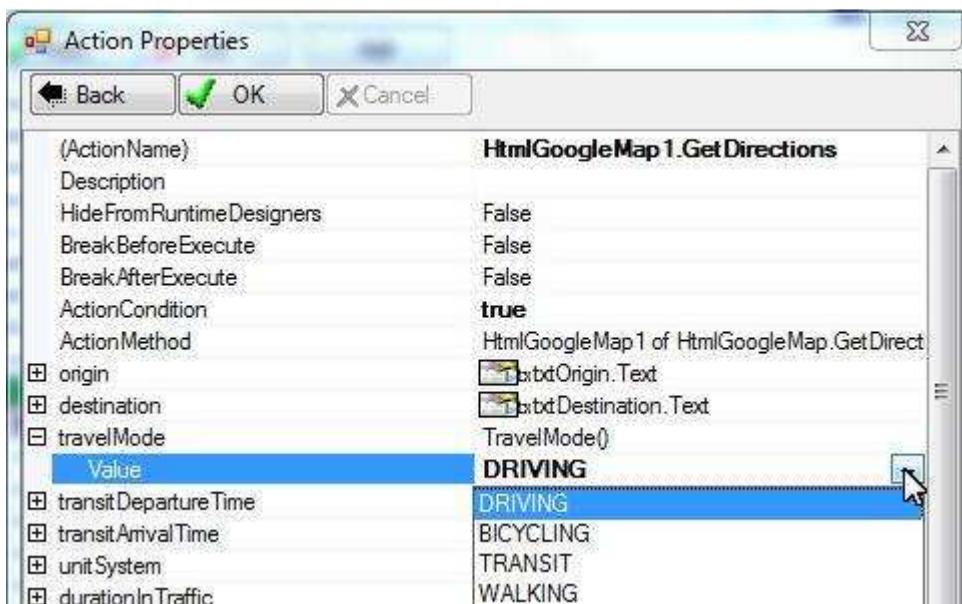
Object picker

Object picker									
<input type="button" value="Back"/>	<input checked="" type="button" value="Next"/>								
<input type="button" value="Exit"/>	<input type="button" value="Add"/>								
<table border="1"> <tr> <td><input type="checkbox"/></td><td>txtDestination of HtmlTextBox</td></tr> <tr> <td><input type="checkbox"/></td><td>Instance Members</td></tr> <tr> <td><input checked="" type="checkbox"/></td><td>Properties inherited</td></tr> <tr> <td><input type="checkbox"/></td><td>BackColor:Color</td></tr> </table>	<input type="checkbox"/>	txtDestination of HtmlTextBox	<input type="checkbox"/>	Instance Members	<input checked="" type="checkbox"/>	Properties inherited	<input type="checkbox"/>	BackColor:Color	<input type="button" value="txtOrigin of HtmlTextBox.Text"/>
<input type="checkbox"/>	txtDestination of HtmlTextBox								
<input type="checkbox"/>	Instance Members								
<input checked="" type="checkbox"/>	Properties inherited								
<input type="checkbox"/>	BackColor:Color								

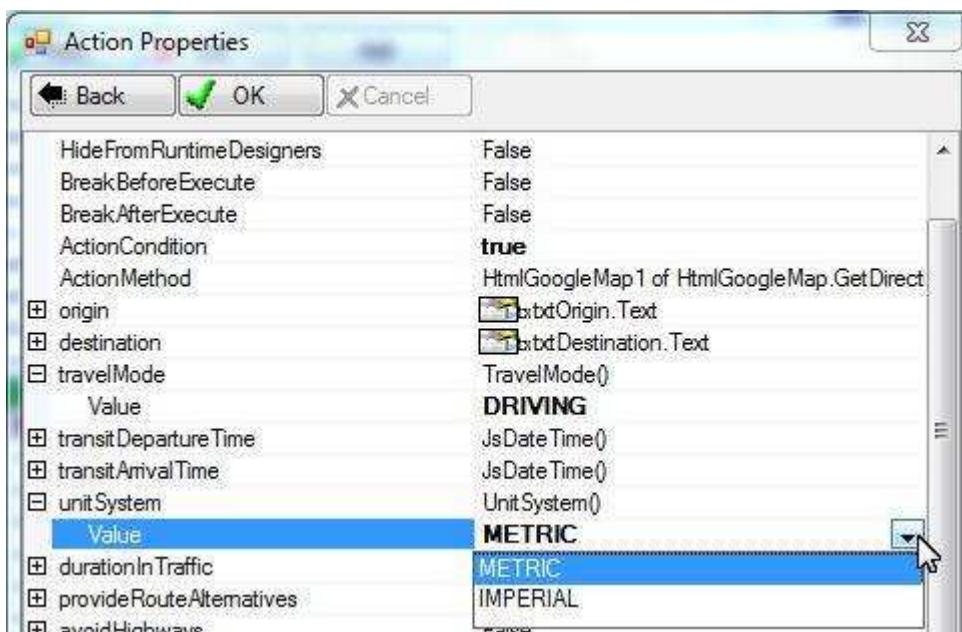
Object picker

Object picker							
<input type="button" value="Back"/>	<input checked="" type="button" value="Next"/>						
<input type="button" value="Exit"/>	<input type="button" value="Add"/>						
<table border="1"> <tr> <td><input type="checkbox"/></td><td>tagName:String</td></tr> <tr> <td><input checked="" type="checkbox"/></td><td>Text:String</td></tr> <tr> <td><input type="checkbox"/></td><td>textAlign:Enum TextAlign</td></tr> </table>	<input type="checkbox"/>	tagName:String	<input checked="" type="checkbox"/>	Text:String	<input type="checkbox"/>	textAlign:Enum TextAlign	<input type="button" value="txtOrigin of HtmlTextBox.Text"/>
<input type="checkbox"/>	tagName:String						
<input checked="" type="checkbox"/>	Text:String						
<input type="checkbox"/>	textAlign:Enum TextAlign						

Choose travel mode:



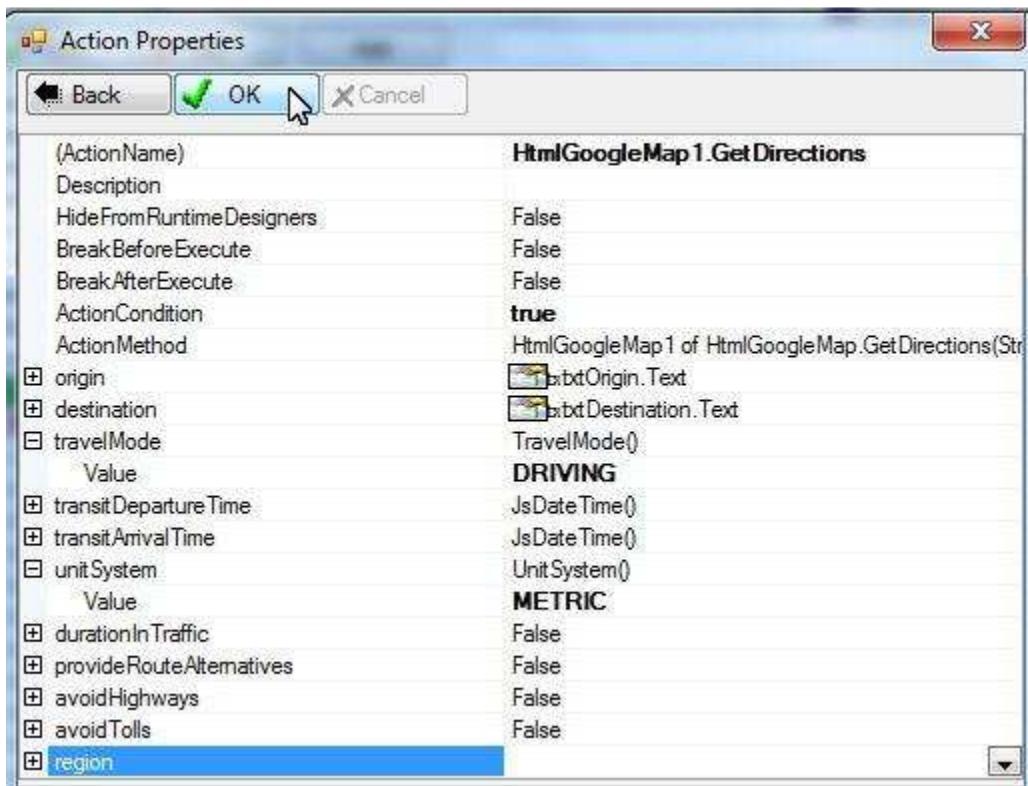
Choose unit system:



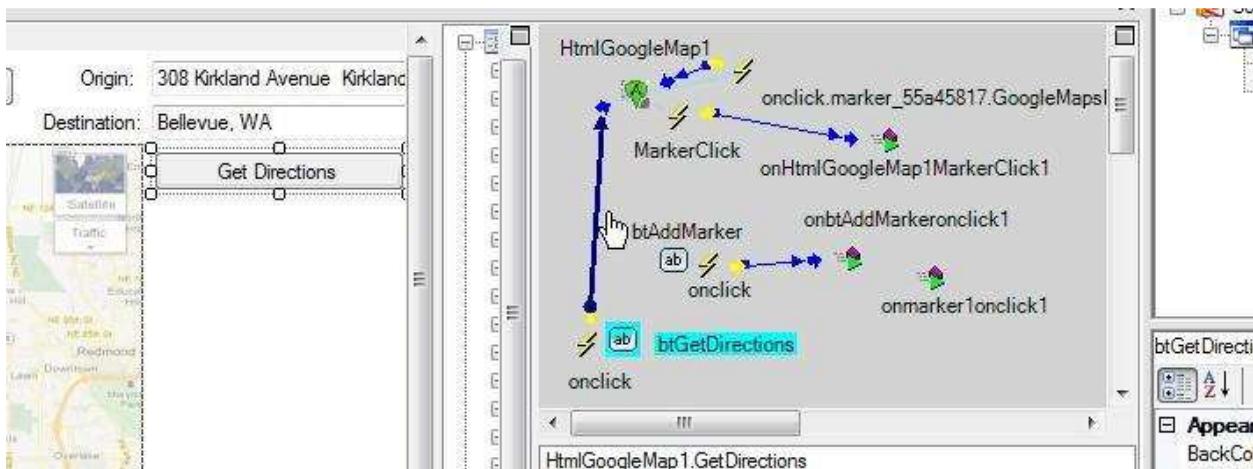
We may specify a region. For more information, see

<https://developers.google.com/maps/documentation/javascript/directions#DirectionsRegionBiasing>

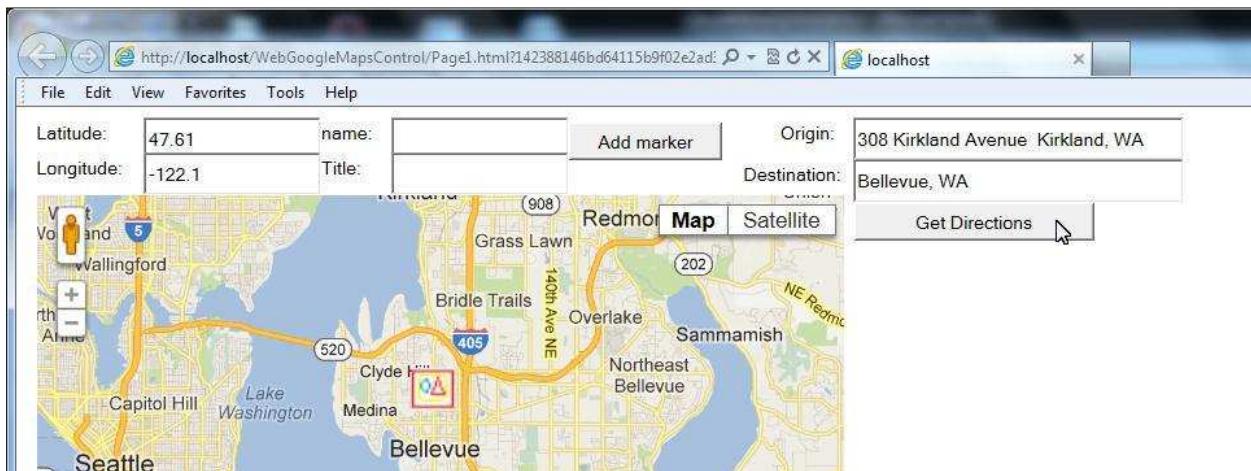
Click OK to finish creating the action:



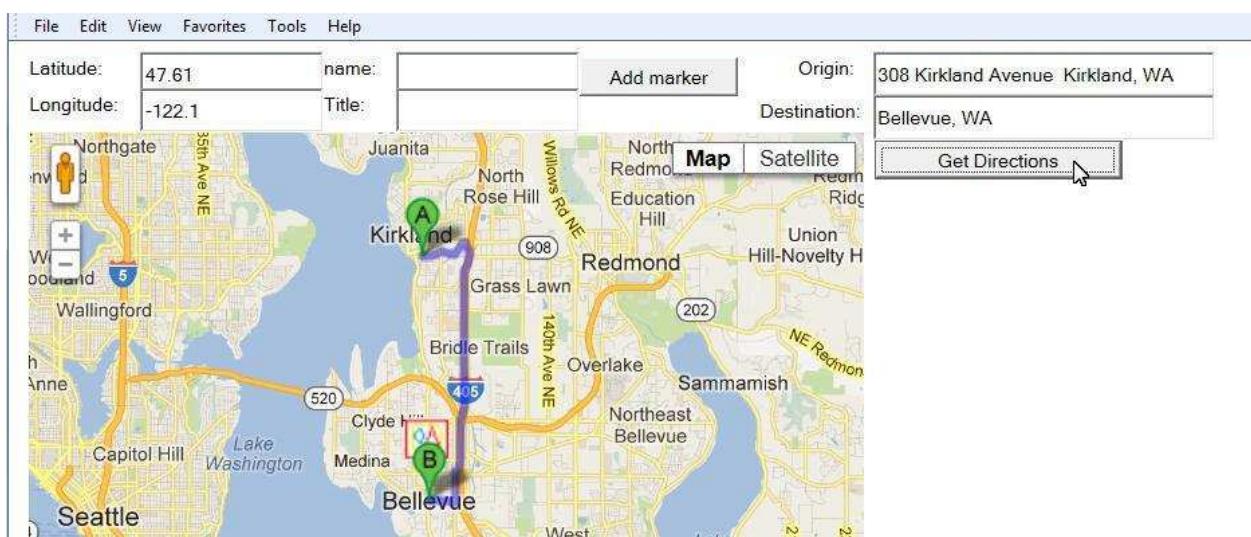
The action is created and assigned to the button:



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`:





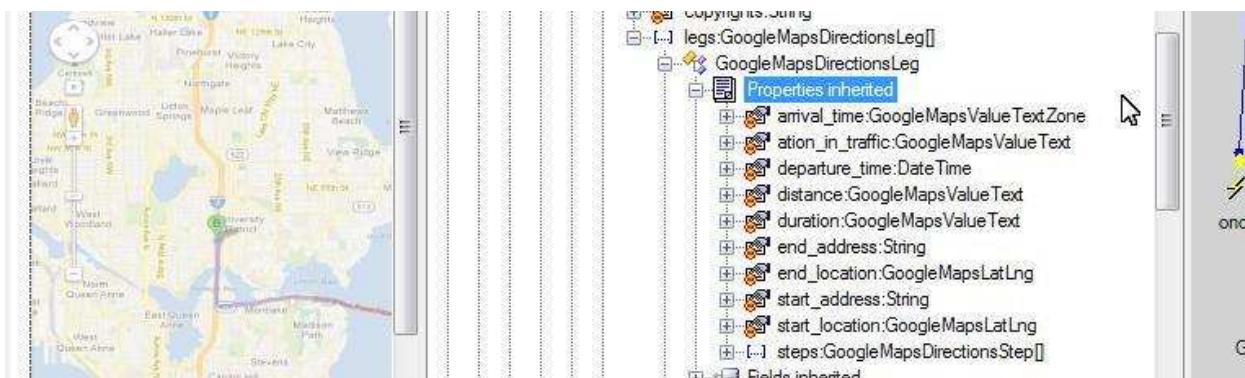
All information can be found under GoogleMapsDirectionsResult:



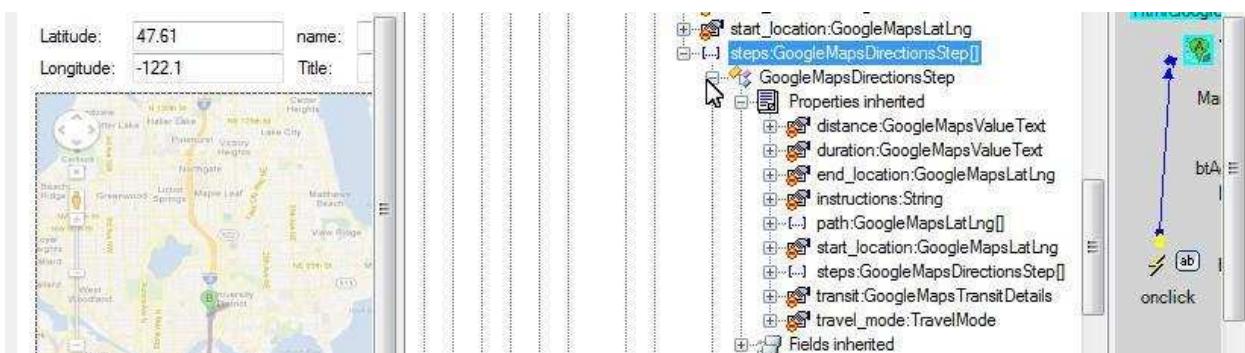
For convenience, the first route contained in routes can be found at FirstRoute property:



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>

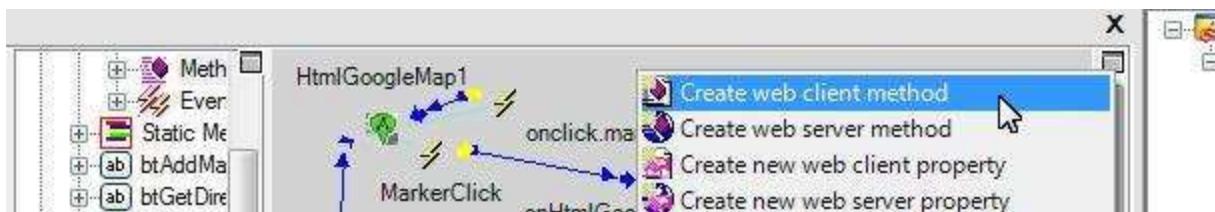
We create a root tree node to represent a route. For each leg in a route, create a child node for the route node. For each step in a leg create a child node for the leg node. For each step in a step create a child node for the step node. The above programming is done in the following way.

- 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



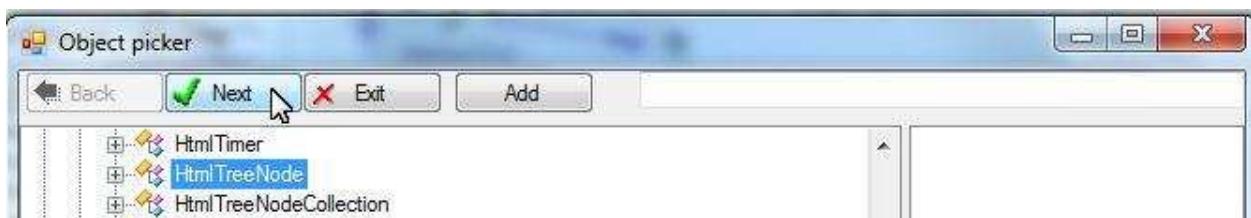
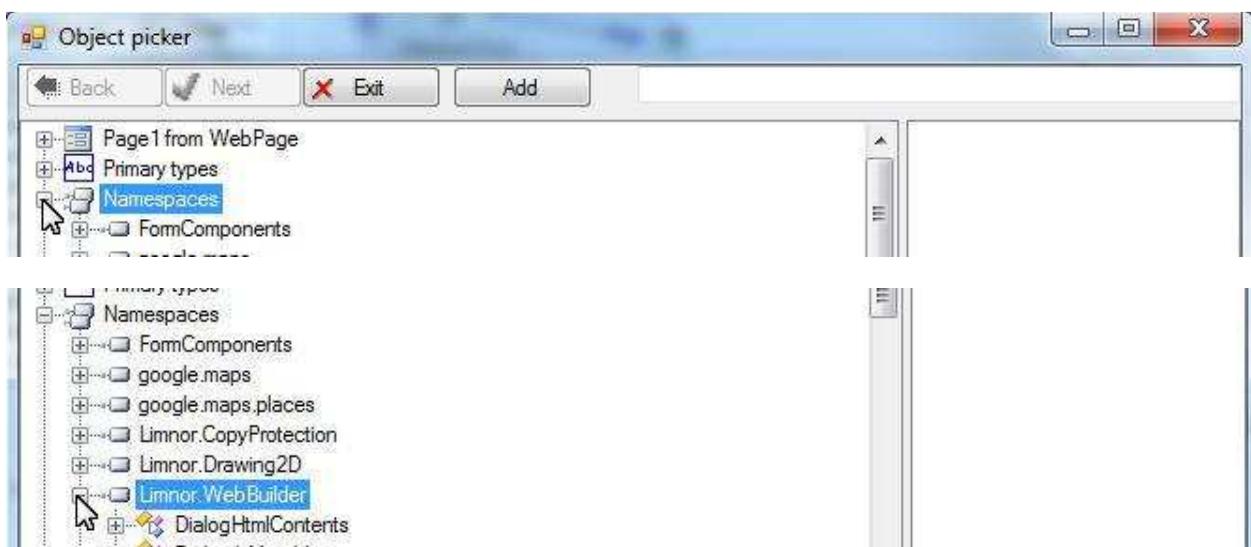
Rename the method:



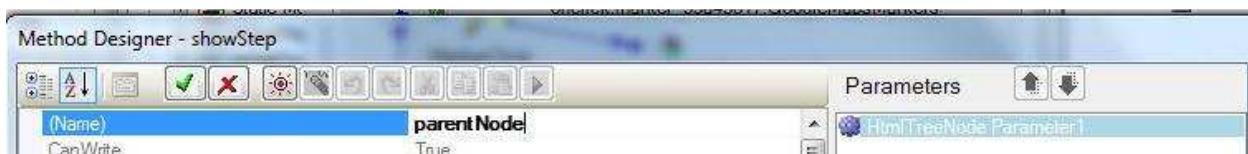
Add parent tree node parameter:



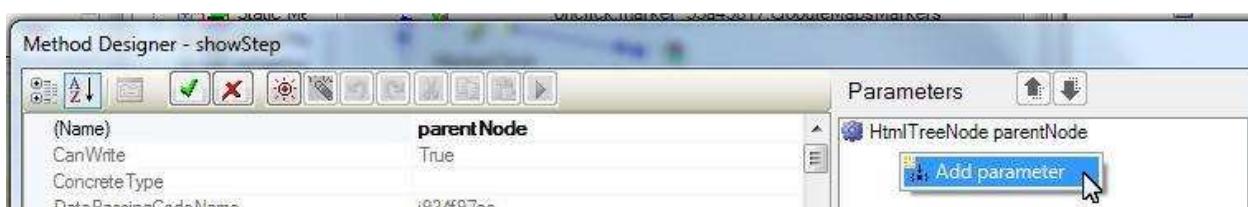
Select tree node:



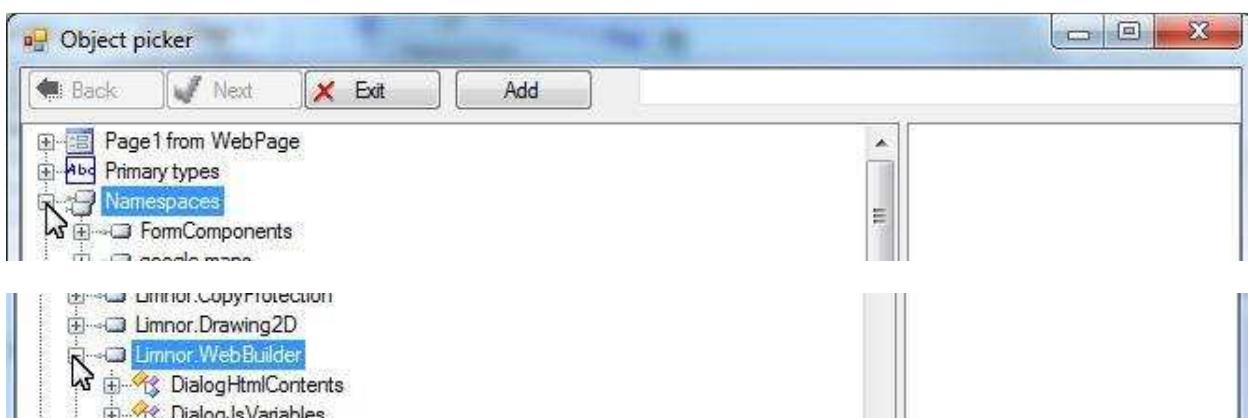
Rename the parameter:



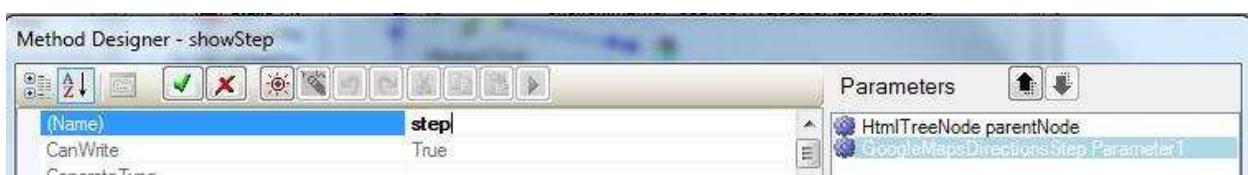
Add a parameter for step:



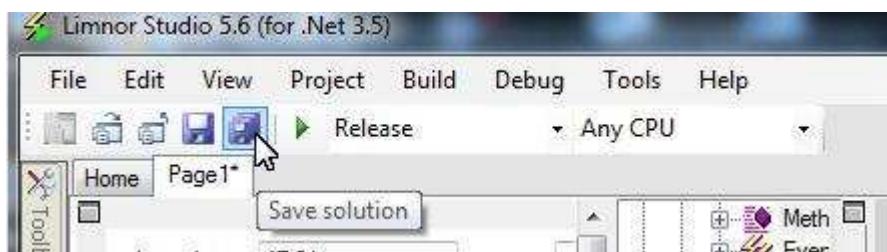
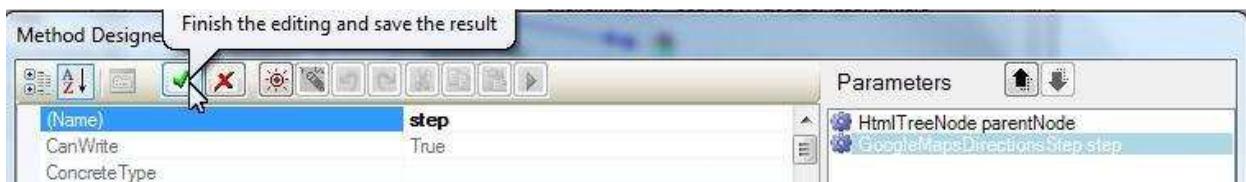
Select step object:



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:

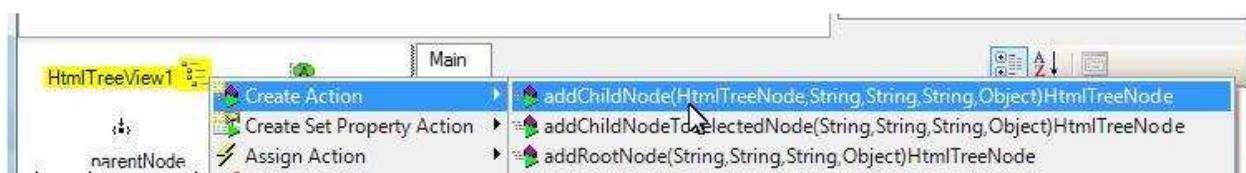


Display step

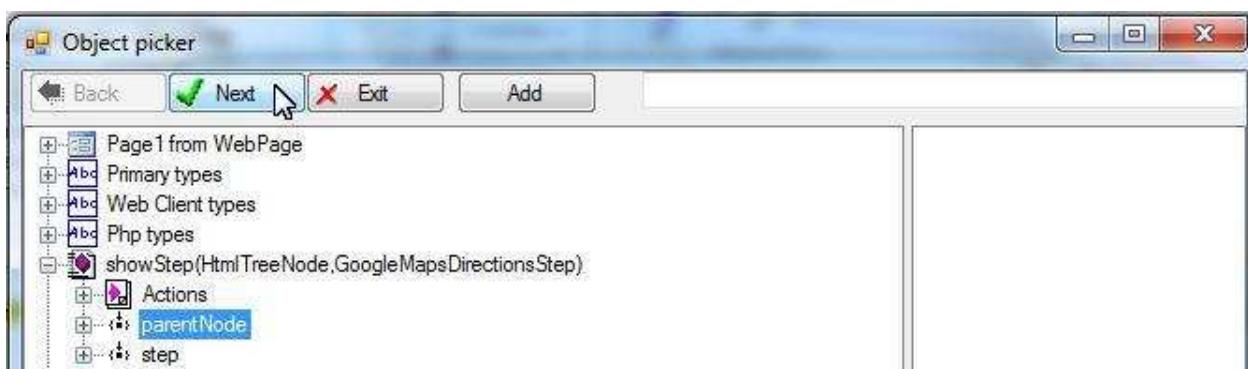
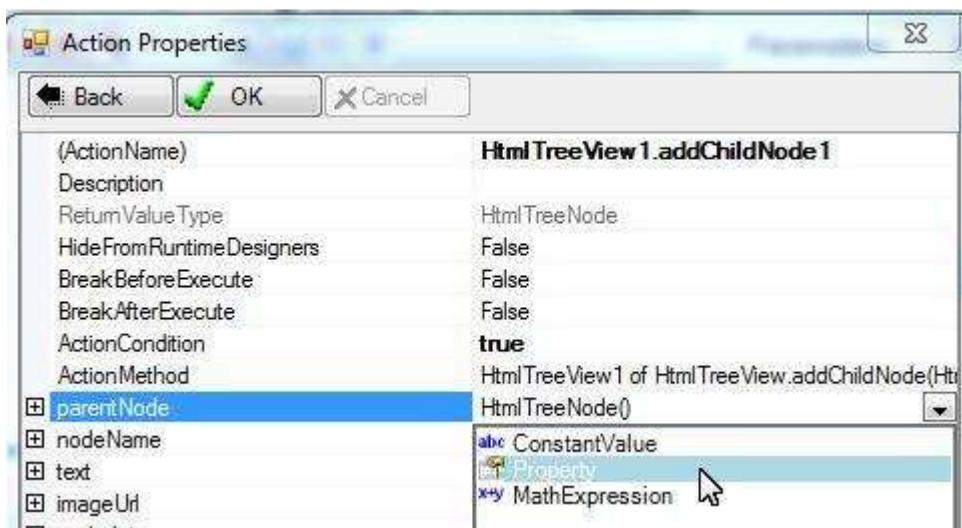
We may edit showStep to display the step:



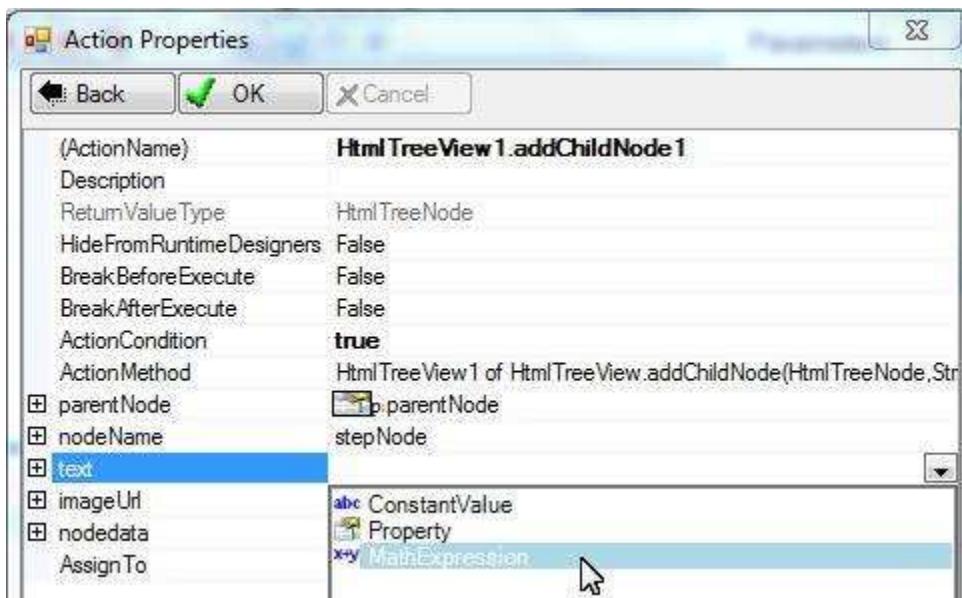
Create an “addChildNode” action to display the step:



Use the method parameter for the parent node:



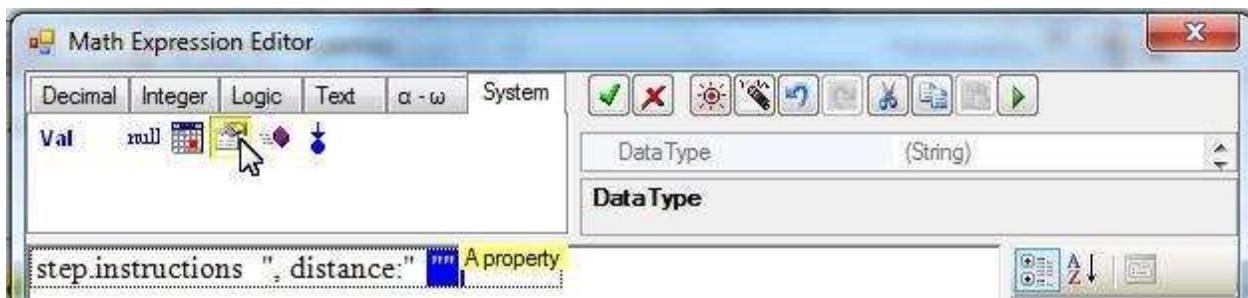
Use an expression to form step display:



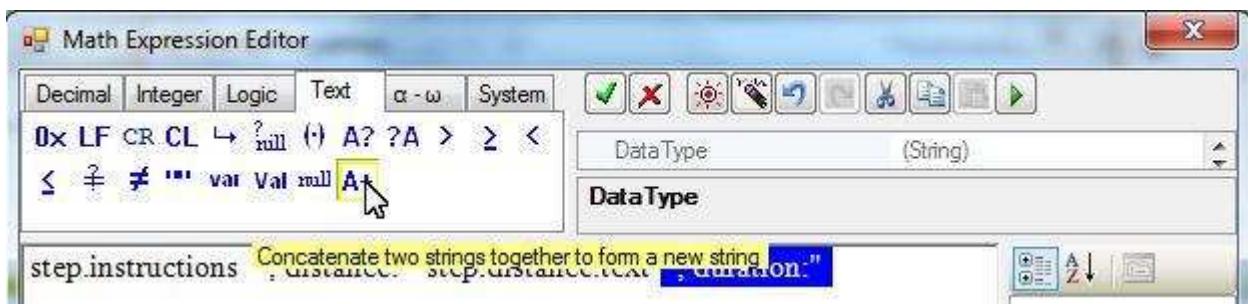
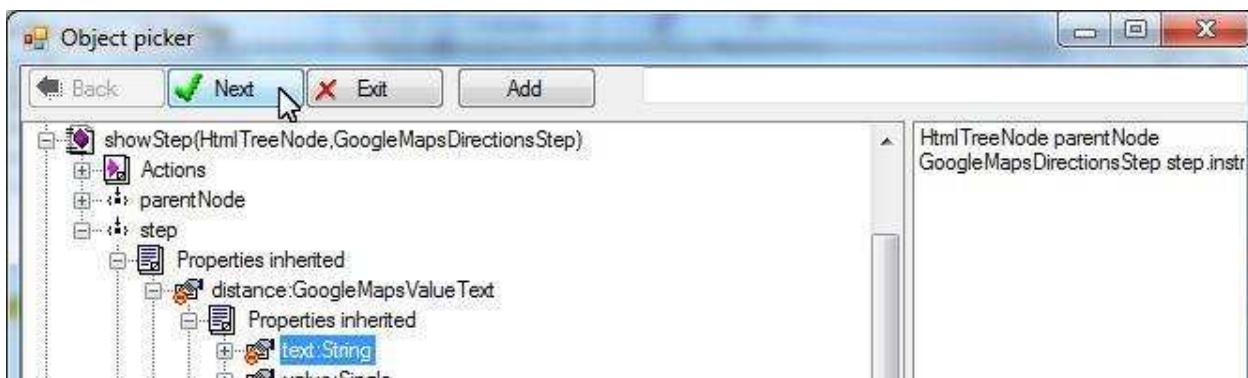
Use "A+" to concatenate step properties we want to display:

The image consists of four separate windows from the Google Maps Control interface:

- Math Expression Editor (Top Left):** Shows a toolbar with buttons for Decimal, Integer, Logic, Text, α-ω, System, and various operators like <, >, ≤, ≥, ≠, and ==. A status bar at the bottom says "Concatenate two strings together to form a new string".
- Math Expression Editor (Top Right):** Shows a toolbar with buttons for Checkmark, Cross, Undo, Redo, and others. A dropdown menu shows "DataType (String)".
- Object picker (Middle Left):** A tree view showing the structure of a page. It includes "Page1 from WebPage", "Primary types", "Web Client types", "Php types", and a specific node "showStep(HtmlTreeNode,GoogleMapsDirectionsStep)" which has "Actions", "parentNode", and "step" children. The "step" node has "Properties inherited" which include "distance:GoogleMapsValueText", "duration:GoogleMapsValueText", "end_location:GoogleMapsLatLn", "instructions:String", and "path:GoogleMapsLatLnq[]".
- Math Expression Editor (Bottom Left):** Shows a toolbar with buttons for Decimal, Integer, Logic, Text, α-ω, System, and various operators like <, >, ≤, ≥, ≠, and ==. A status bar at the bottom says "Concatenate two strings together to form a new string".
- Math Expression Editor (Bottom Right):** Shows a toolbar with buttons for Checkmark, Cross, Undo, Redo, and others. A dropdown menu shows "DataType (String)".



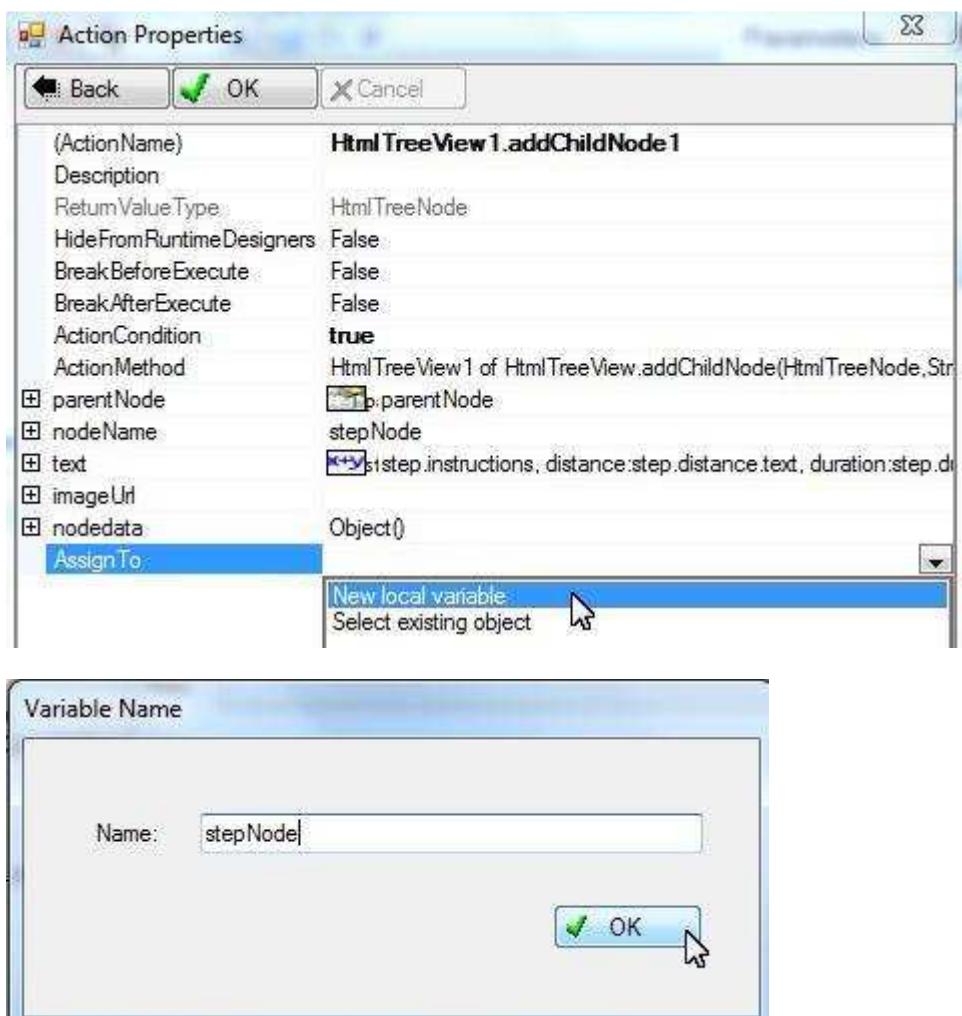
Note that we use "text" property of "distance":



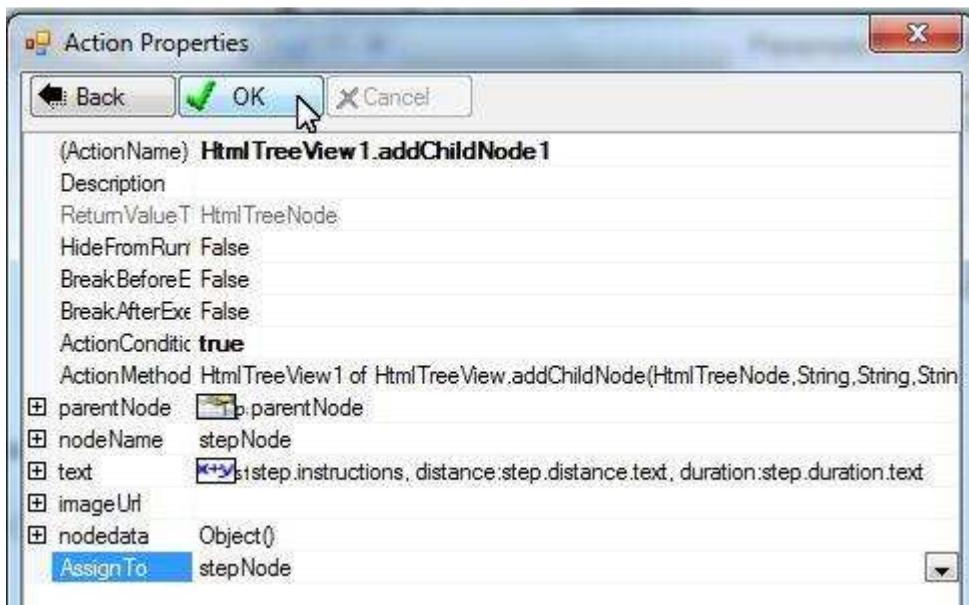
The image consists of three vertically stacked windows from the Google Maps Control interface:

- Top Window:** A "Math Expression Editor" window. The toolbar includes buttons for Decimal, Integer, Logic, Text, and System. The status bar shows "Val null". The main area contains the expression "step.instructions ", distance:" step.distance.text ", duration:". The "DataType" dropdown is set to "(String)".
- Middle Window:** An "Object picker" window. It shows a tree structure under "showStep(HtmlTreeNode, GoogleMapsDirectionsStep)": "Actions", "parentNode", "step". Under "step", there are "Properties inherited": "distance:GoogleMapsValueText" and "duration:GoogleMapsValueText". Below these are "Properties inherited": "text:String" and "value:Sindle". To the right, a tooltip displays "HtmlTreeNode parentNode" and "GoogleMapsDirectionsStep step.step.instr" and "GoogleMapsDirectionsStep step.step.dist".
- Bottom Window:** Another "Math Expression Editor" window. The toolbar includes buttons for Decimal, Integer, Logic, Text, and System. The status bar shows "Val null". The main area contains the expression "step.instructions ", distance:" step.distance.text ", duration:". The "DataType" dropdown is set to "(Double)".

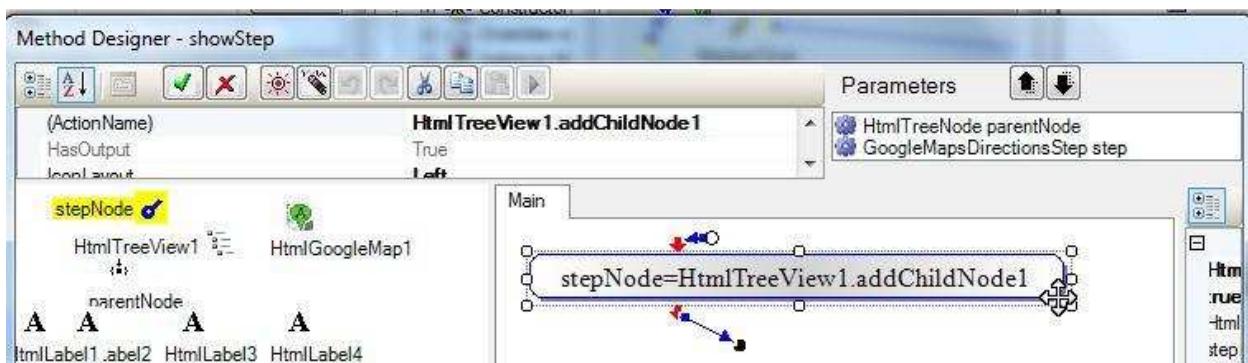
Use a new variable for the new tree node:



Click OK to finish creating the action:

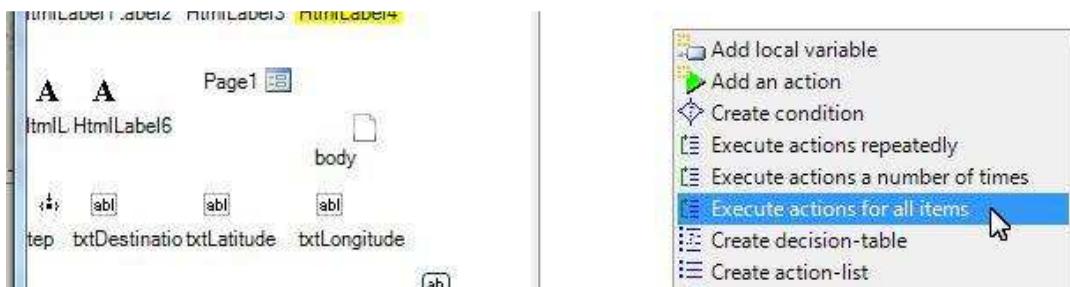


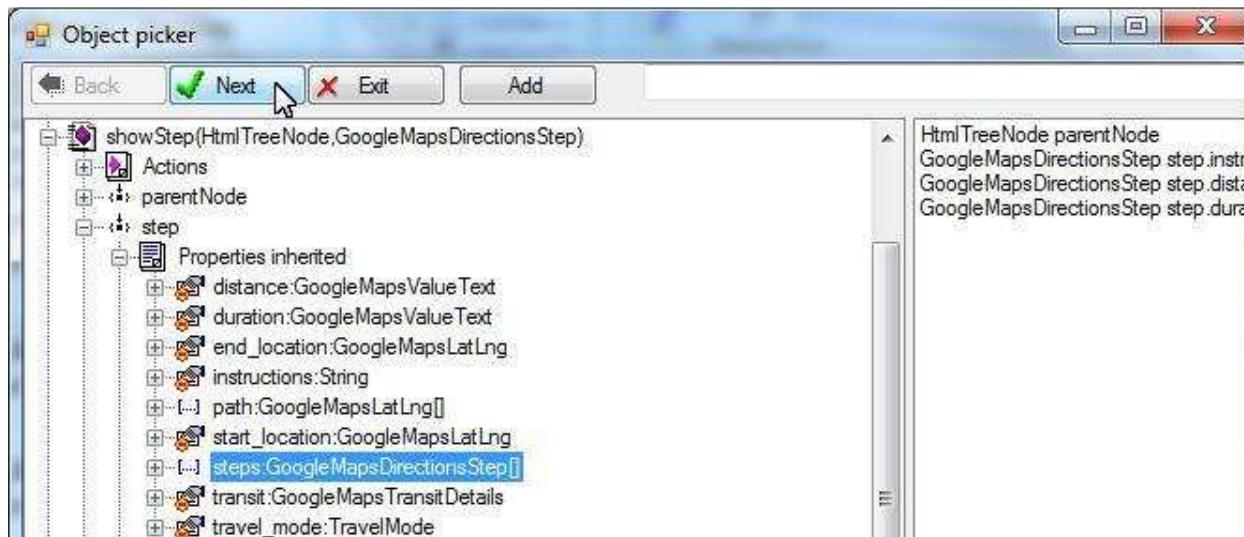
The new step node and the new action appear:



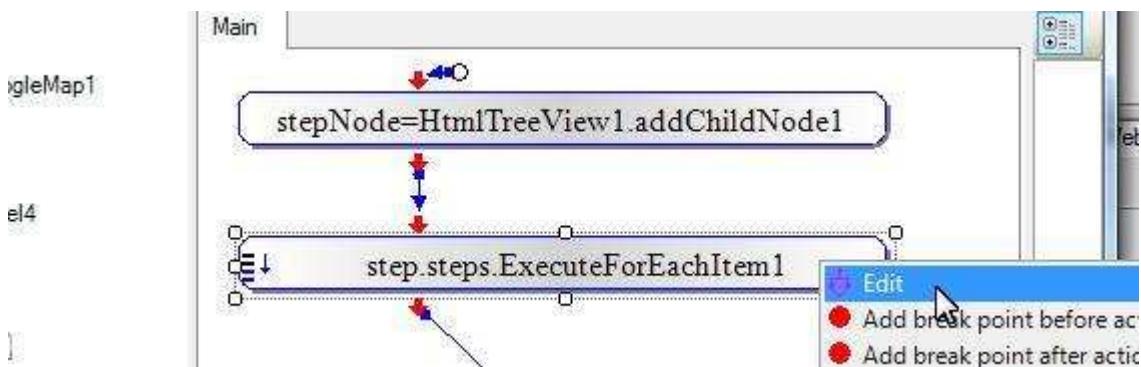
Display sub-steps

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

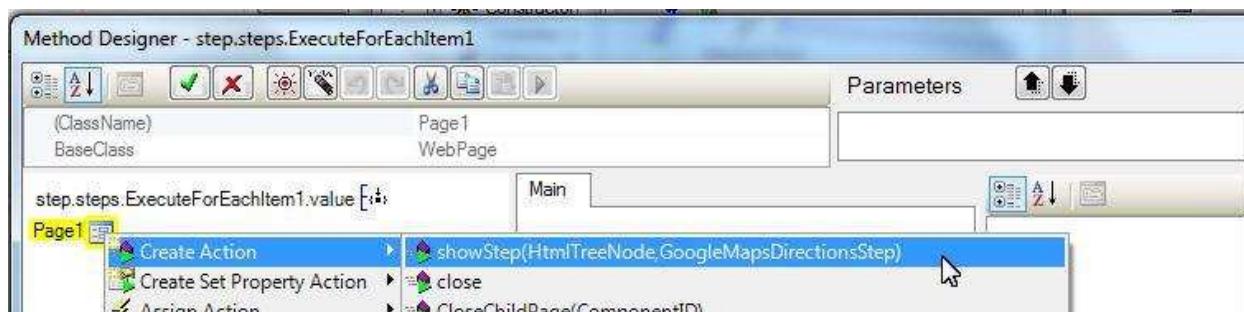




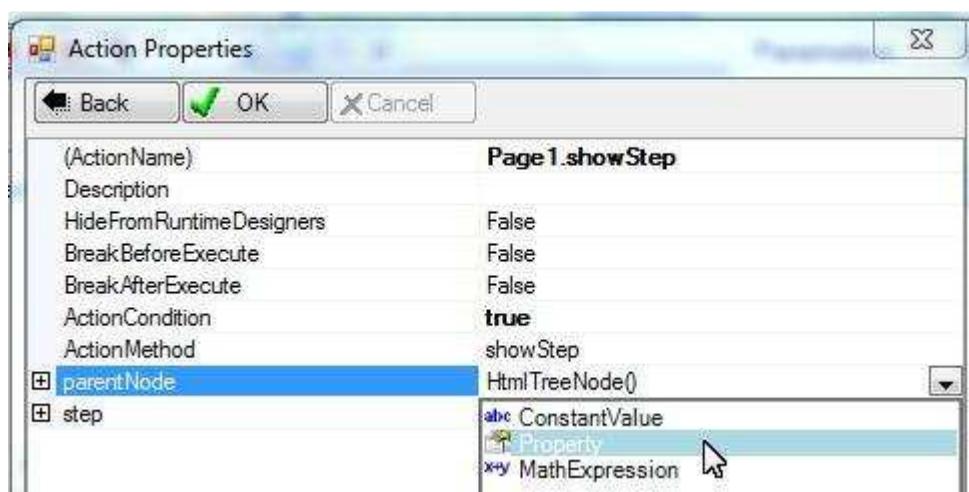
Edit the action to perform actions on each step:



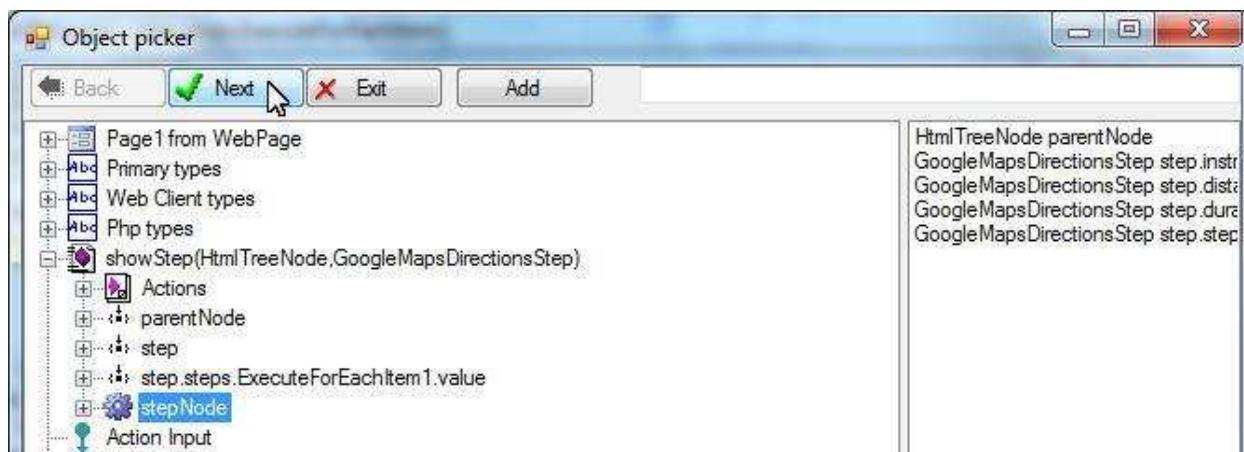
A method editor appears for adding actions. Note that “value” icon is the step to be displayed. Create a “showStep” action to display the step:



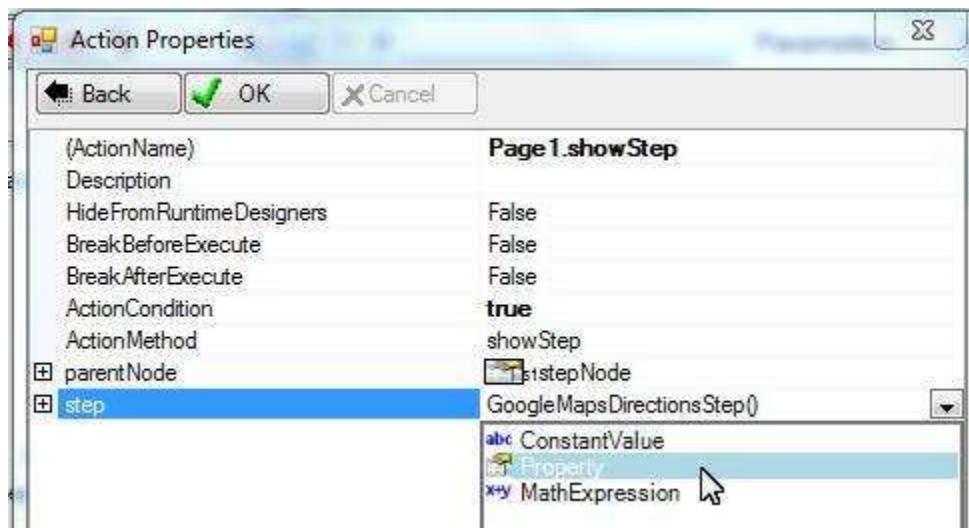
Select parent node:

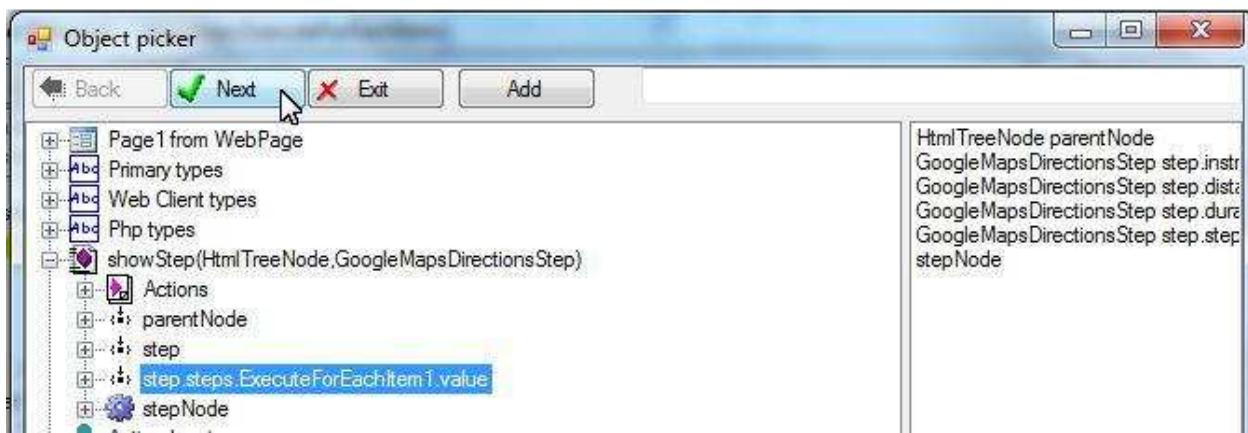


Note that we must use the stepNode variable as the parent node:

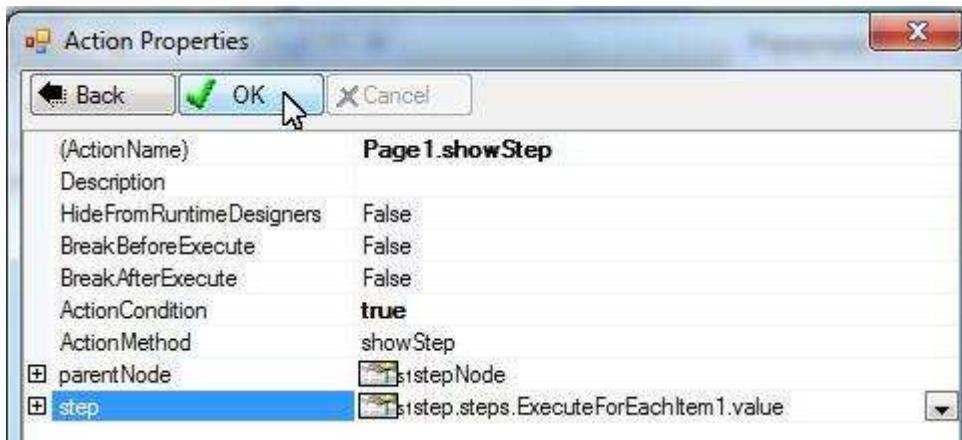


Select "value" for step:

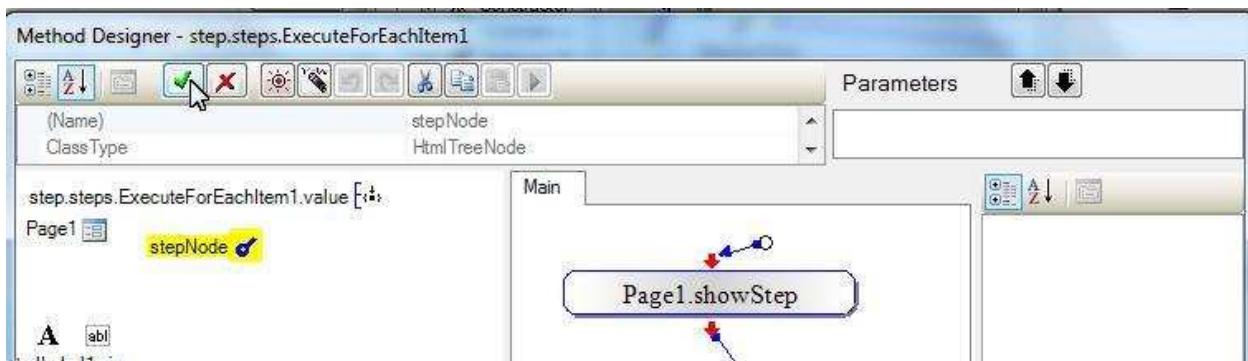




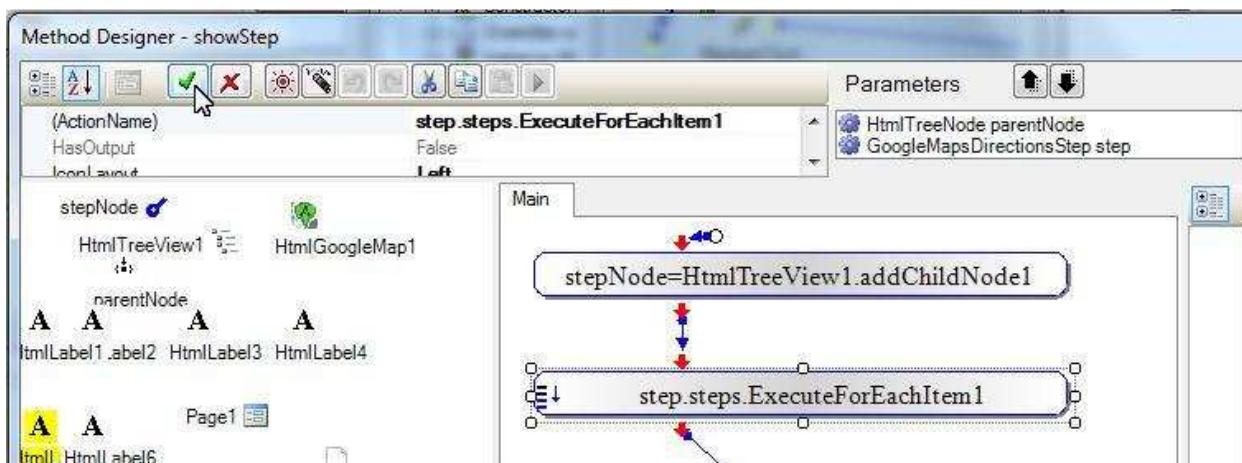
Click OK:



That is all we need:



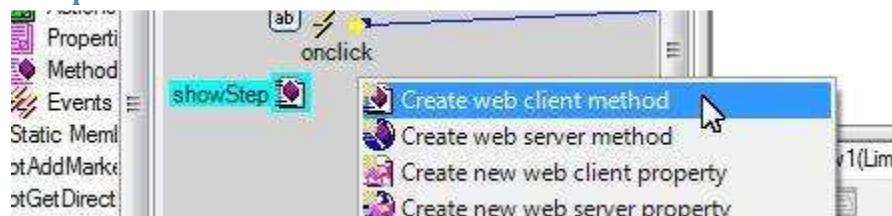
We are done creating showStep:



Show Leg

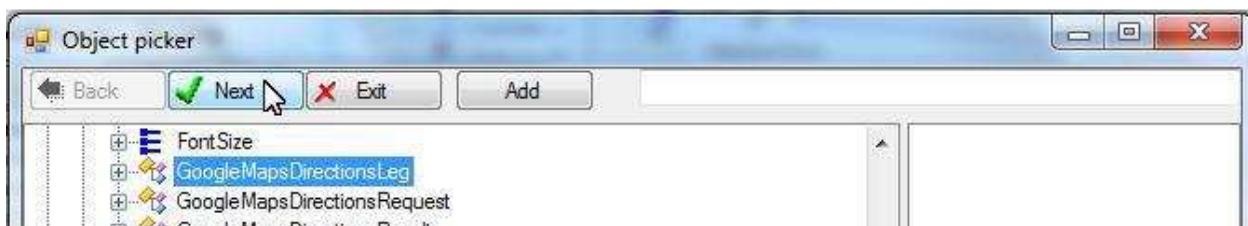
We may create a showLeg method.

Add parameters



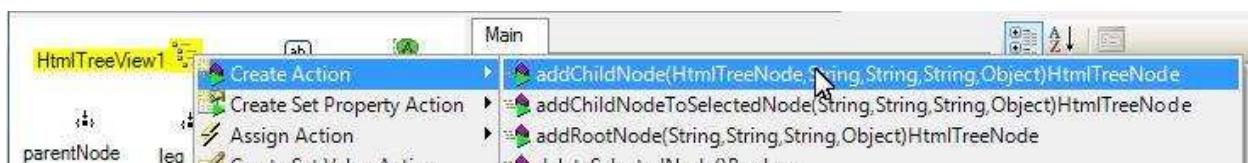
Rename the method and add a parent tree node parameter. Add a parameter for leg:

The screenshots show the 'Object picker' dialog being used to select components. The first dialog shows the 'Namespaces' section with 'Limnor.Drawing2D' and 'Limnor.WebBuilder' expanded. The second dialog shows 'Limnor.WebBuilder' selected. The third dialog shows 'DialogHtmlContents' and 'DialogJsVariables' expanded under 'Limnor.WebBuilder'.

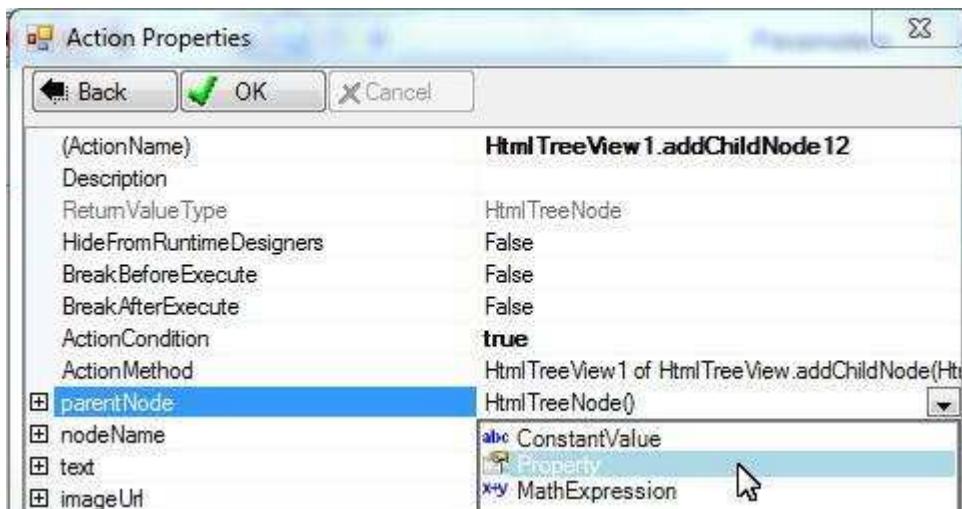


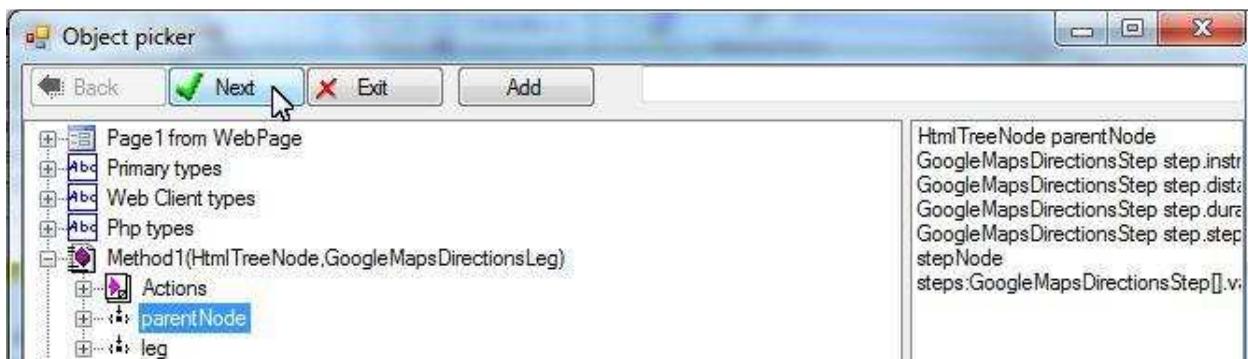
Display leg

Add a child node to display the leg.

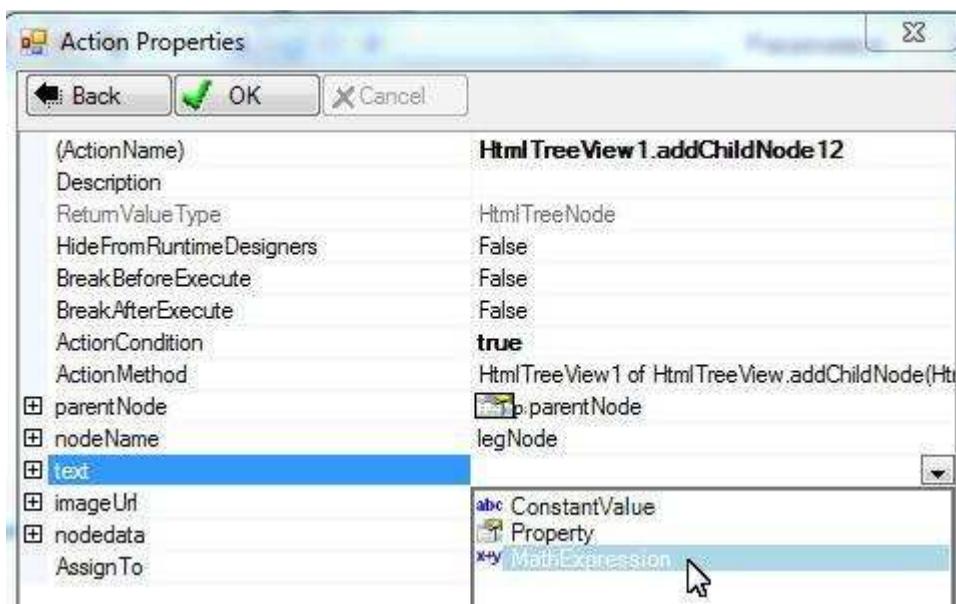


Set parent node:





Set node text to display the leg:



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

Object picker

Back Next Exit Add

- Page1 from WebPage
- Primary types
- Web Client types
- Php types
- Method1(HtmlTreeNode,GoogleMapsDirectionsLeg)
 - Actions
 - parentNode
 - leg
 - Properties inherited
 - arrival_time:GoogleMapsValueTextZone
 - action_in_traffic:GoogleMapsValueText
 - departure_time:DateTime
 - distance:GoogleMapsValueText
 - duration:GoogleMapsValueText
 - end_address:String
 - end_location:GoogleMapsLatLng
 - start_address:String

HtmlTreeNode parentNode
 GoogleMapsDirectionsStep step instr
 GoogleMapsDirectionsStep step.dist
 GoogleMapsDirectionsStep step.dura
 GoogleMapsDirectionsStep step.step
 stepNode
 steps:GoogleMapsDirectionsStep[] v:

Math Expression Editor

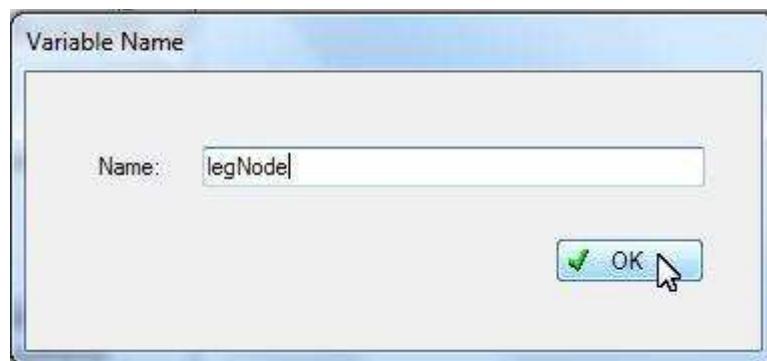
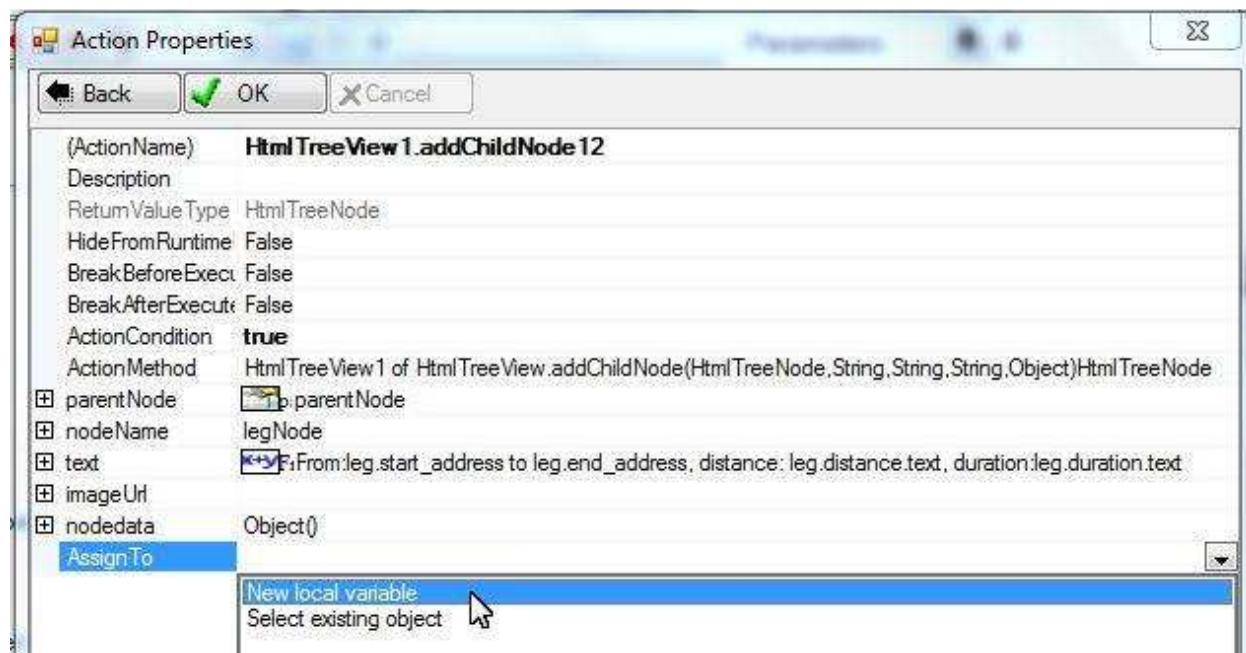
Finish the editing.

Decimal Integer Logic Text $\alpha - \omega$ System

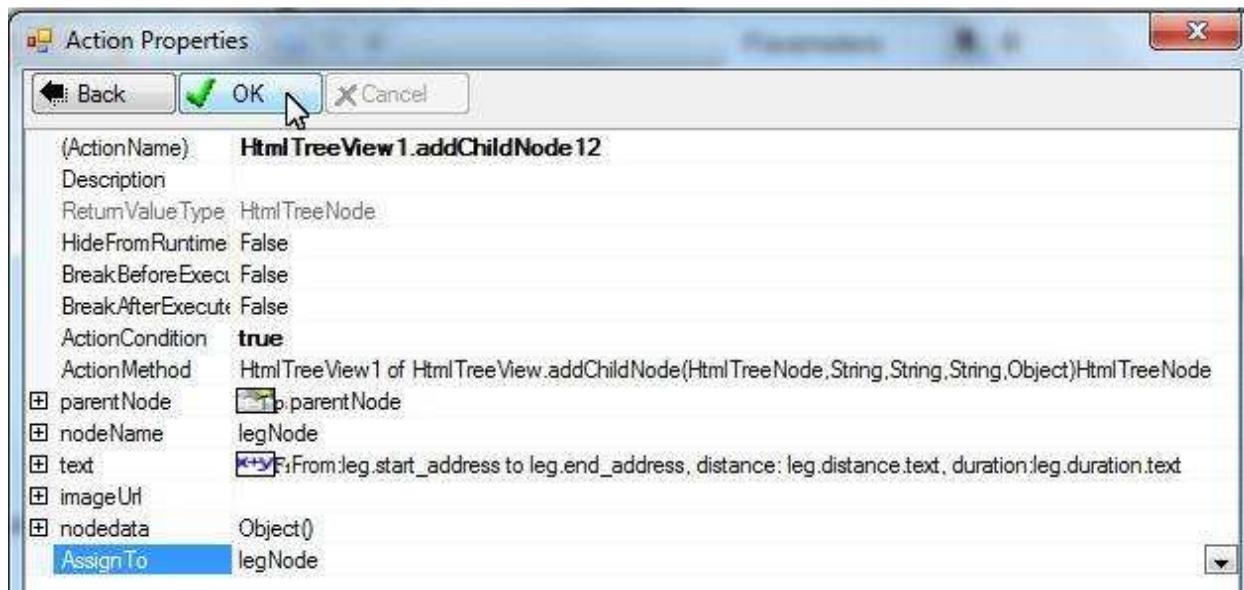
Name	text
Name	

```
"From:" leg.start_address " to " leg.end_address ", distance: "
leg.distance.text ", duration:" leg.duration.text
```

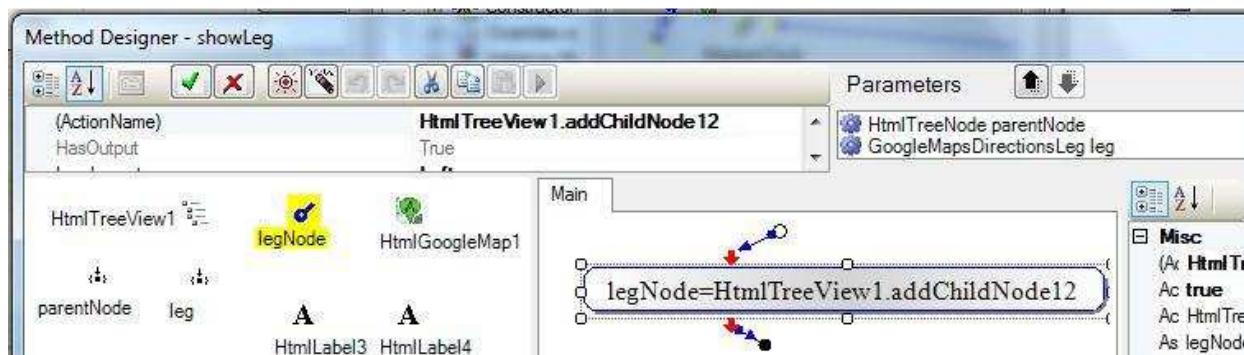
Use a variable for the new node:



Click OK:

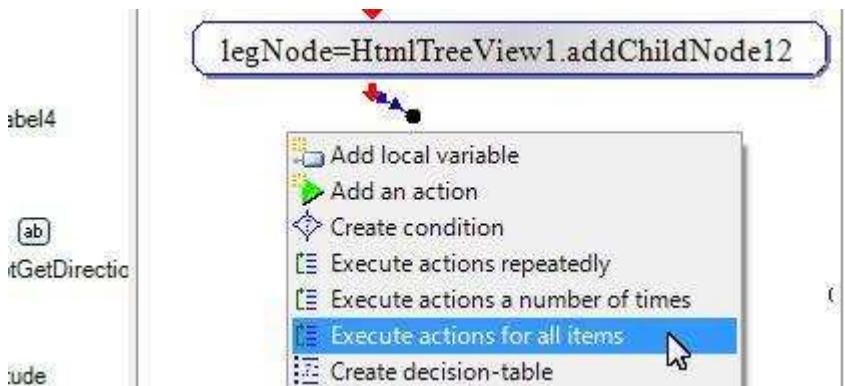


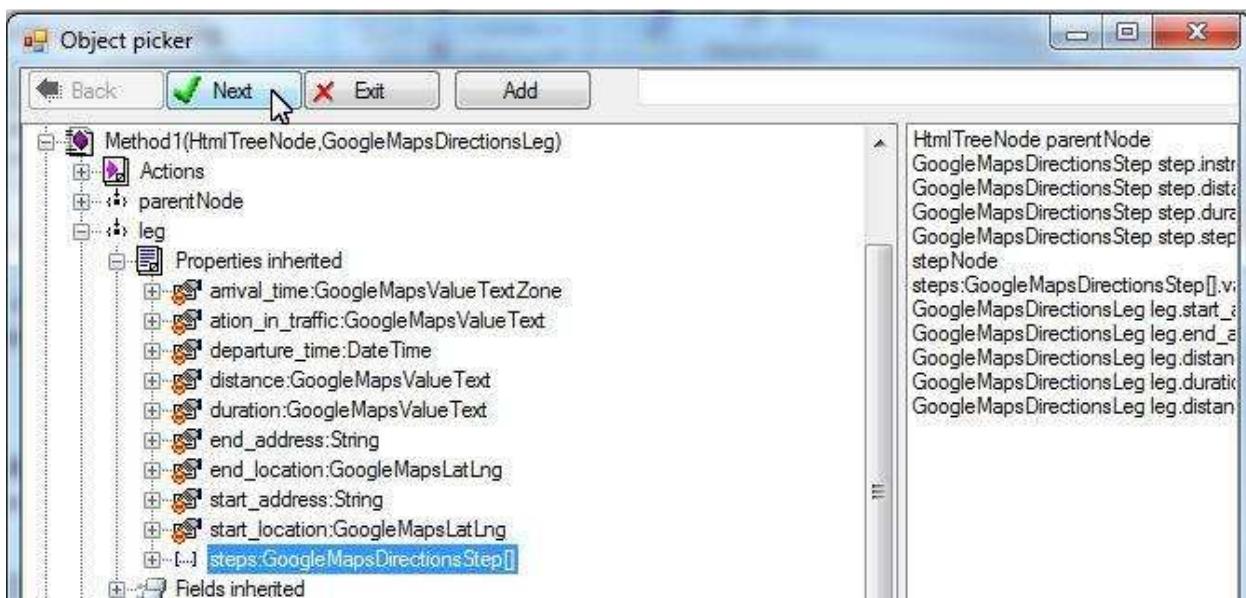
The action and variable legNode appear:



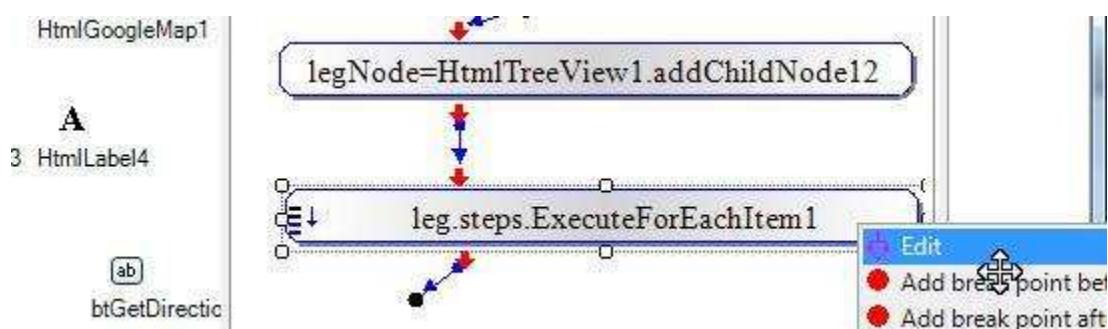
Display steps

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.

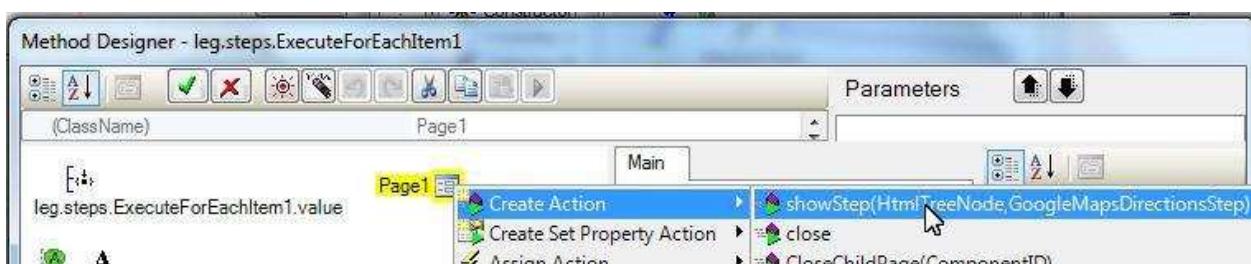




Edit the action to process each step:



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



Use the legNode variable as the parentNode:

Action Properties

<input type="button" value="Back"/>	<input checked="" type="button" value="OK"/>	<input type="button" value="Cancel"/>
(ActionName) Page1.showStep2		
Description	False	
HideFromRuntimeDesigners	False	
BreakBeforeExecute	False	
BreakAfterExecute	False	
ActionCondition	true	
ActionMethod	showStep	
parentNode	HtmlTreeNode()	
step	<input type="button" value="ConstantValue"/> <input checked="" type="button" value="Property"/> <input type="button" value="MathExpression"/>	

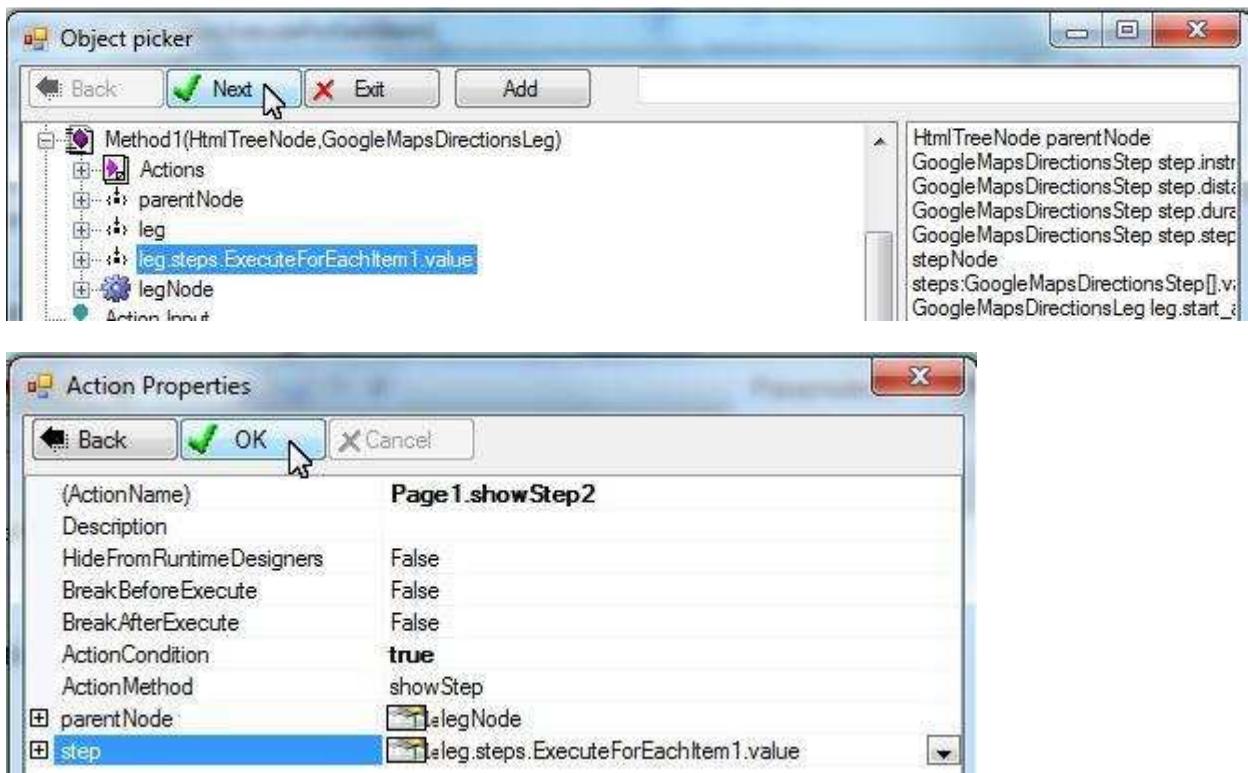
Object picker

<input type="button" value="Back"/>	<input checked="" type="button" value="Next"/>	<input type="button" value="Exit"/>	<input type="button" value="Add"/>
<ul style="list-style-type: none"> + Page1 from WebPage + Abc Primary types + Abc Web Client types + Abc Php types - Method1(HtmlTreeNode,GoogleMapsDirectionsLeg) <ul style="list-style-type: none"> + Actions + parentNode + leg + leg.steps.ExecuteForEachItem1.value legNode + Action Input 		<ul style="list-style-type: none"> HtmlTreeNode parentNode GoogleMapsDirectionsStep step.instr GoogleMapsDirectionsStep step.dist GoogleMapsDirectionsStep step.duration GoogleMapsDirectionsStep step.stepNode steps:GoogleMapsDirectionsStep[] step.steps GoogleMapsDirectionsLeg leg.start_address GoogleMapsDirectionsLeg leg.end_address GoogleMapsDirectionsLeg leg.distance GoogleMapsDirectionsLeg leg.duration GoogleMapsDirectionsLeg leg.distance GoogleMapsDirectionsLeg leg.steps 	

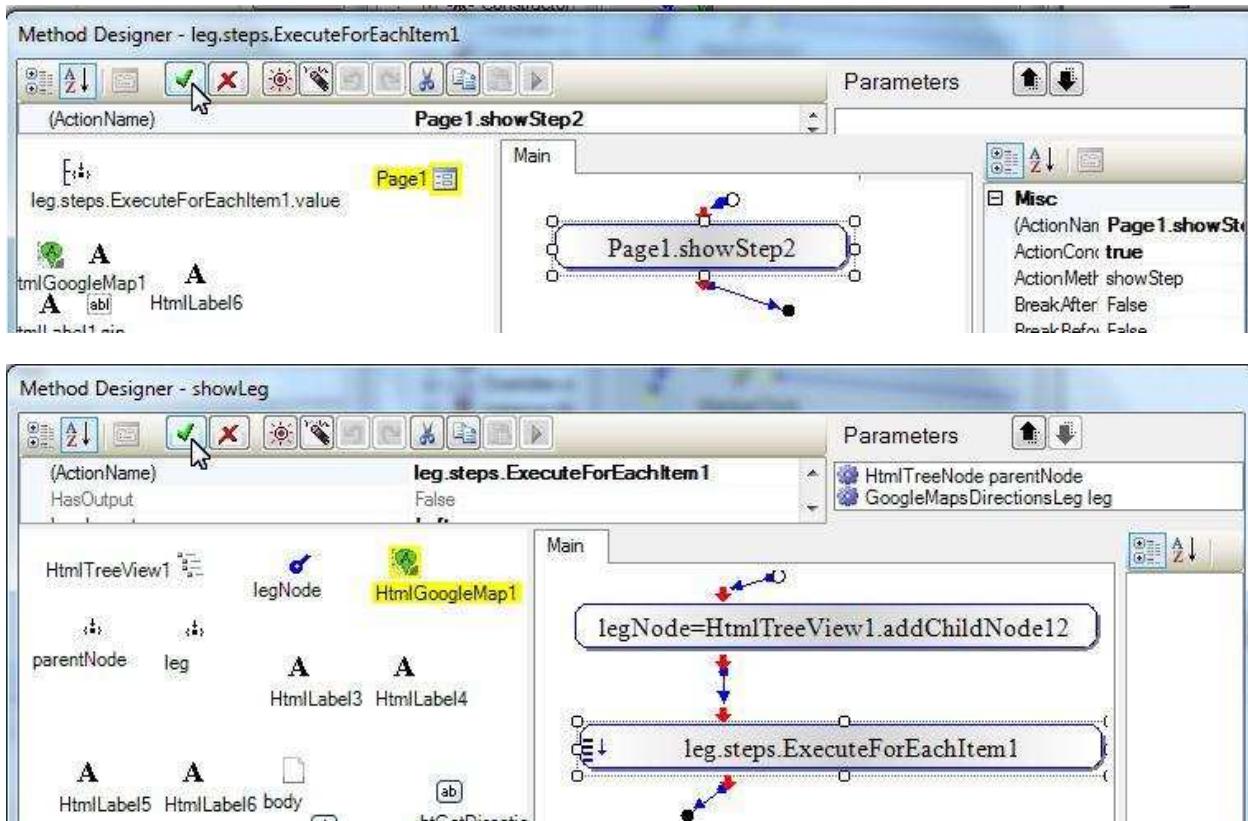
Select the item value for step:

Action Properties

<input type="button" value="Back"/>	<input checked="" type="button" value="OK"/>	<input type="button" value="Cancel"/>
(ActionName) Page1.showStep2		
Description	False	
HideFromRuntimeDesigners	False	
BreakBeforeExecute	False	
BreakAfterExecute	False	
ActionCondition	true	
ActionMethod	showStep	
parentNode	legNode	
step	<input type="button" value="ConstantValue"/> <input checked="" type="button" value="Property"/> <input type="button" value="MathExpression"/>	

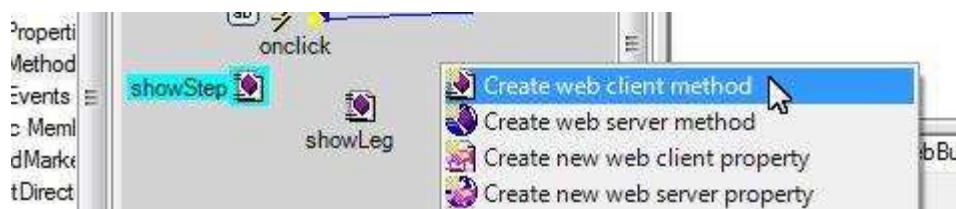


That is all for this sample.

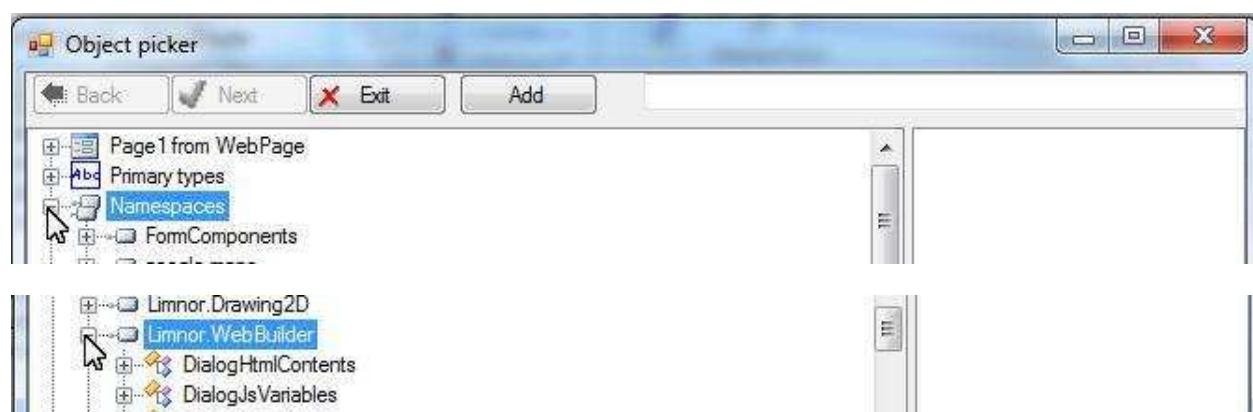
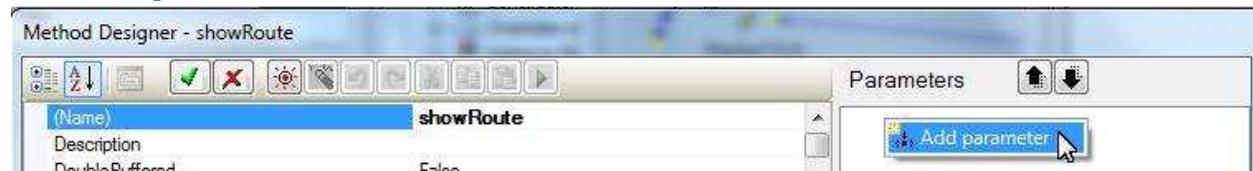


Show route

Add a method to display a route.

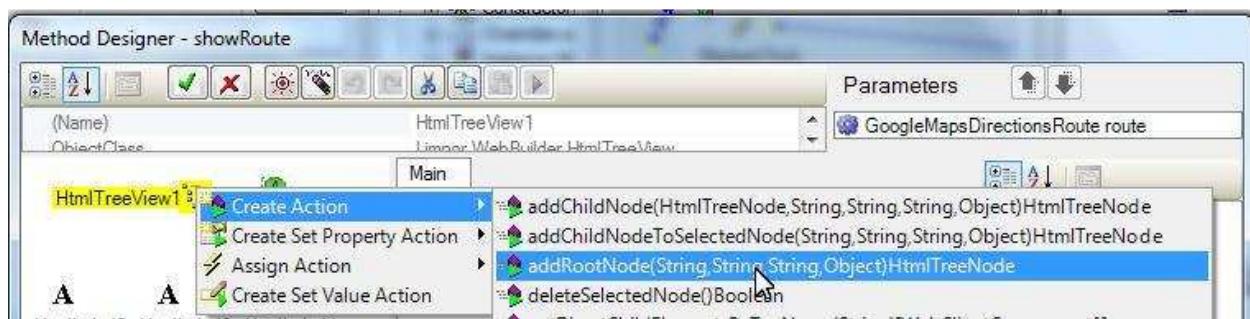


Add route parameter

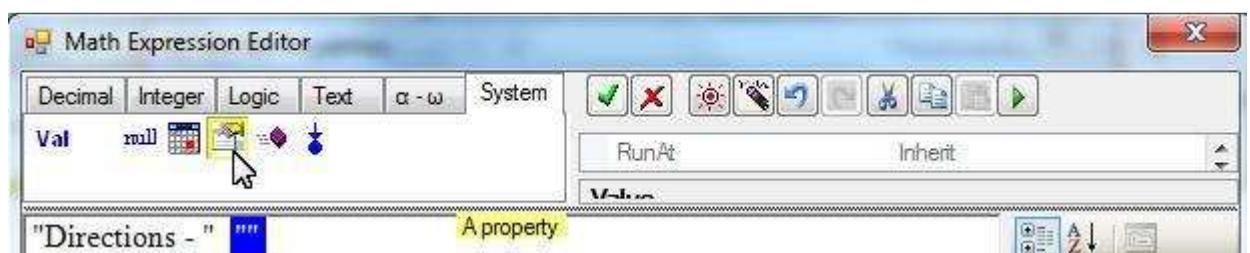
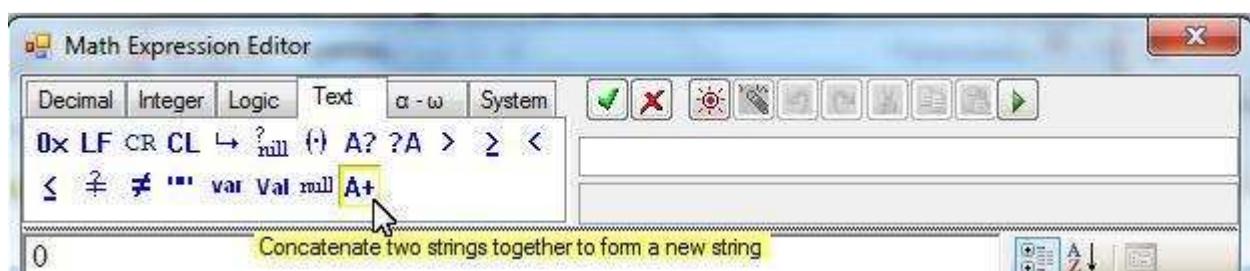
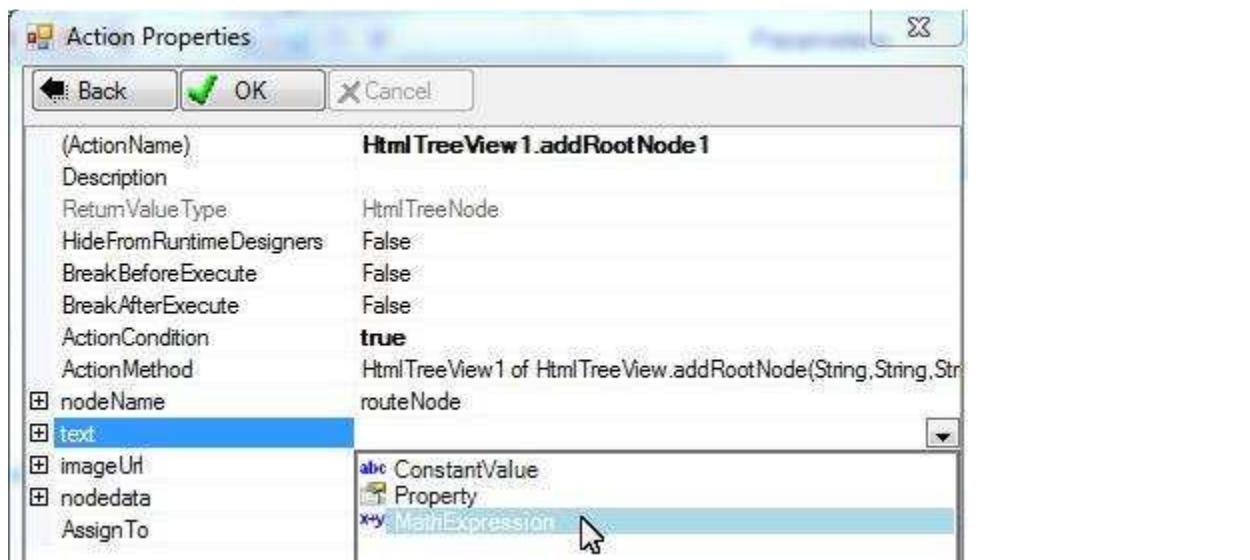


Create a root node

We may create a root node for the route:



Form display text using an expression:

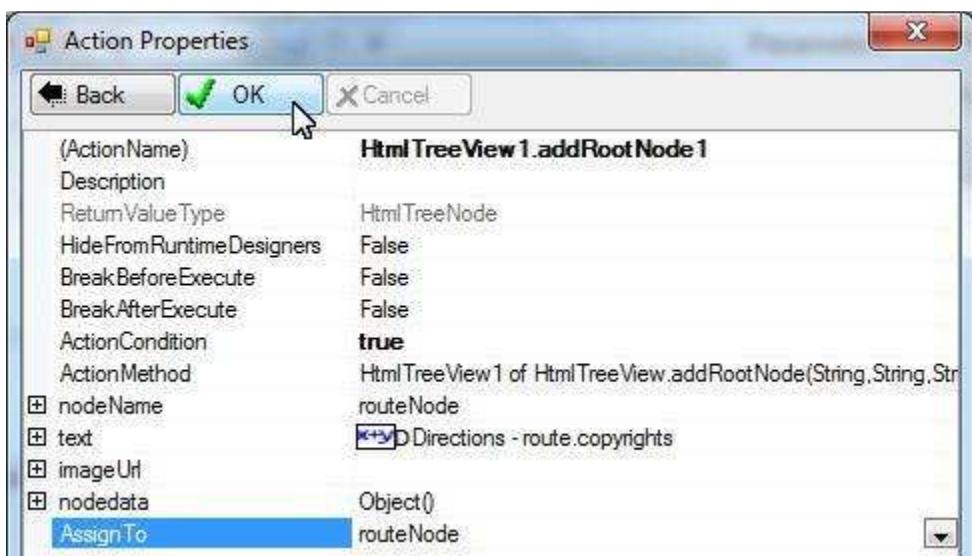
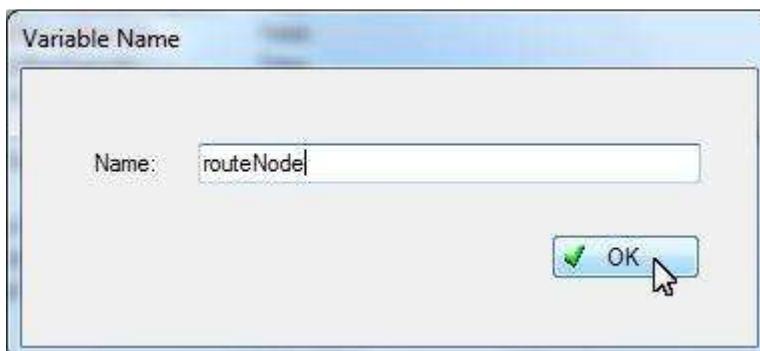


The screenshot shows two windows side-by-side. The top window is titled "Object picker" and displays a tree view of objects. The tree includes "Page1 from WebPage", "Primary types", "Web Client types", "Php types", "Method1(GoogleMapsDirectionsRoute)", "Actions", "route", "Properties inherited", and "copyrights:String". To the right of the tree, there is a list of properties for the selected "route" object, such as "HtmlTreeNode parentNode", "GoogleMapsDirectionsStep step.instr", etc. The bottom window is titled "Math Expression Editor" and shows a text input field containing the expression "Directions - route.copyrights". A tooltip above the input field says "Finish the editing.". Below the input field are buttons for Decimal, Integer, Logic, Text, and System, along with a toolbar and a status bar showing "CanWrite" and "True".

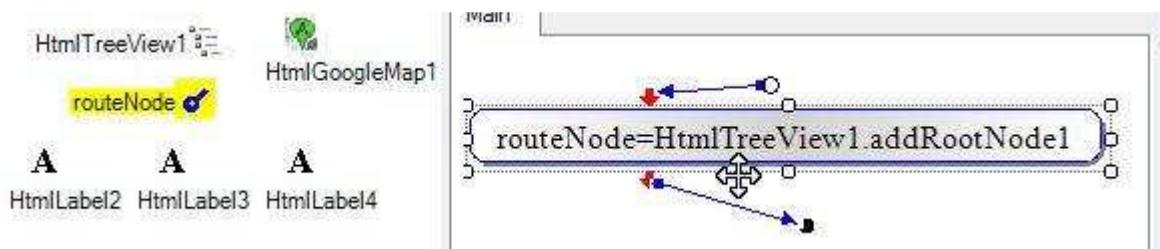
Use a variable for the new route node:

The screenshot shows the "Action Properties" dialog box. The "ActionName" is set to "HtmlTreeView1.addRootNode1". Under the "text" property, the value is "Directions - route.copyrights". In the "AssignTo" dropdown menu, the option "New local variable" is highlighted. Other options in the menu include "Select existing object".

ActionName	HtmlTreeView1.addRootNode1
Description	
ReturnValueType	HtmlTreeNode
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	HtmlTreeView1 of HtmlTreeView.addRootNode(String, String, String)
nodeName	routeNode
text	Directions - route.copyrights
imageUrl	
nodedata	Object()
AssignTo	New local variable

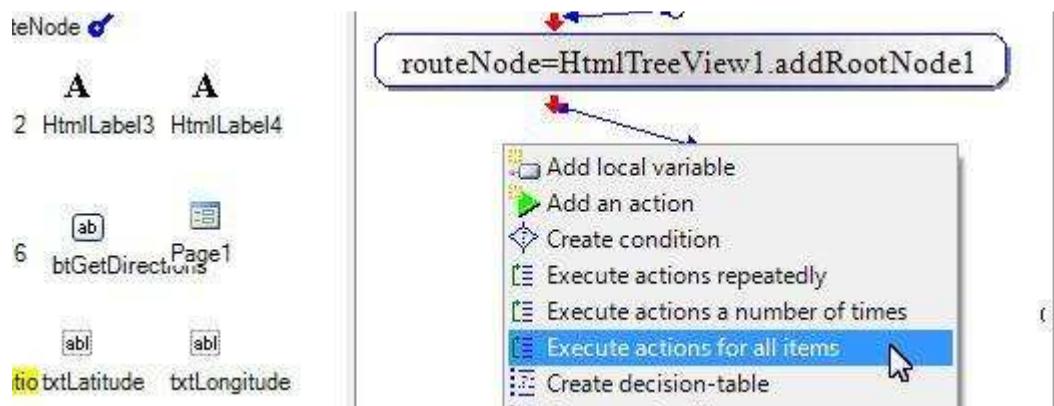


The action and the variable appear.

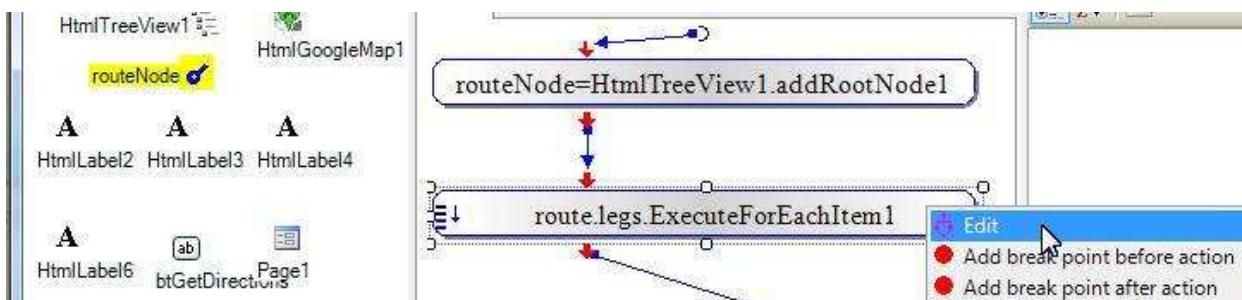


Display legs

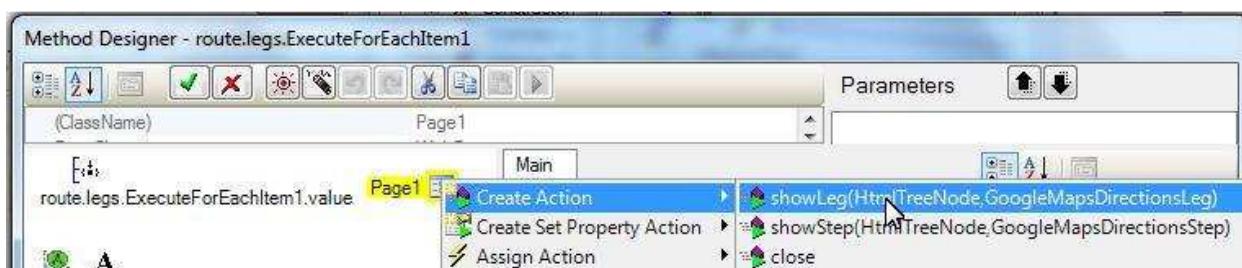
Property "legs" is an array of legs. We may use an "Execute actions for each item" to display all legs.



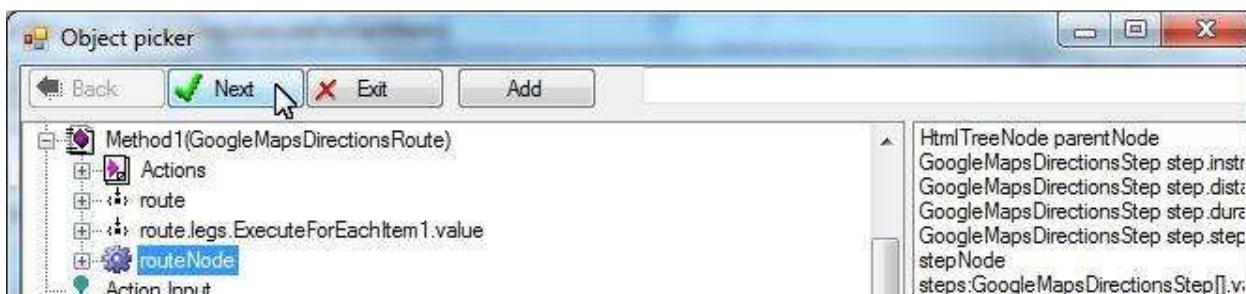
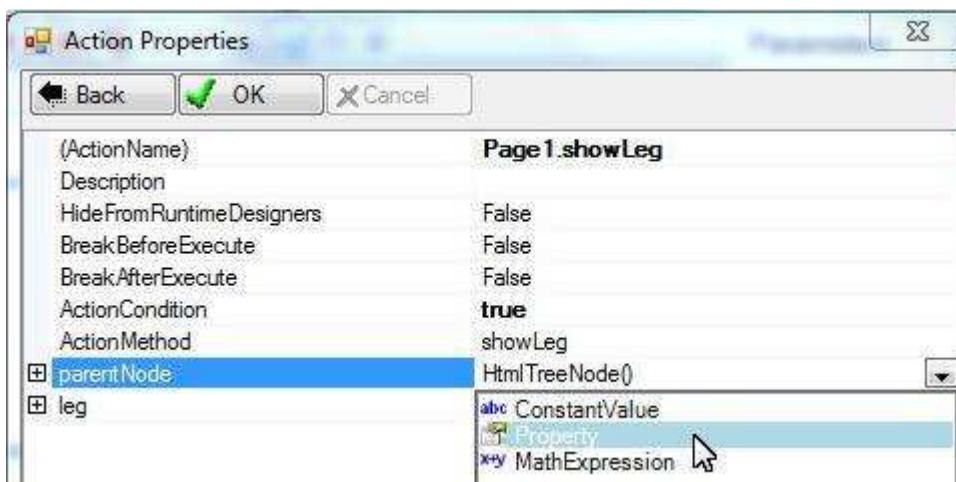
Edit the action to process each leg:



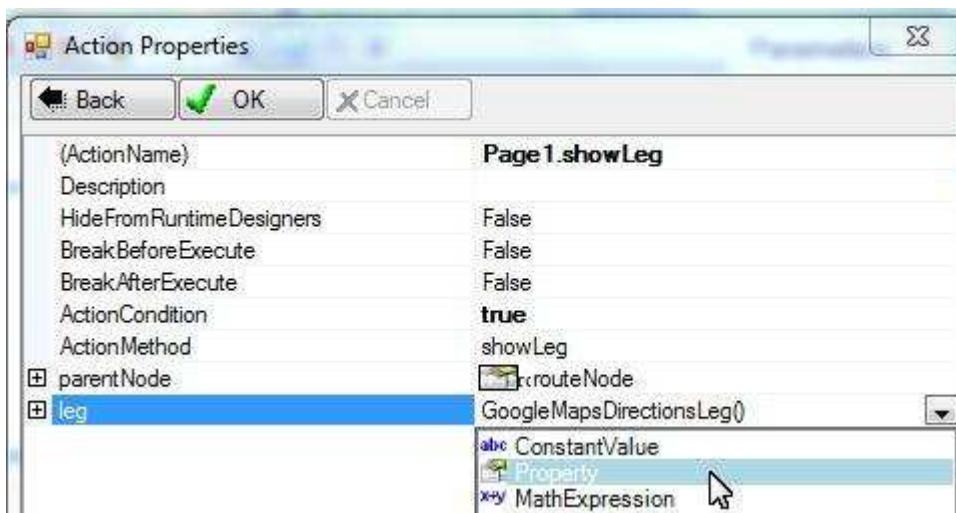
A method editor appears for adding actions. Create a showLeg action to display the leg:

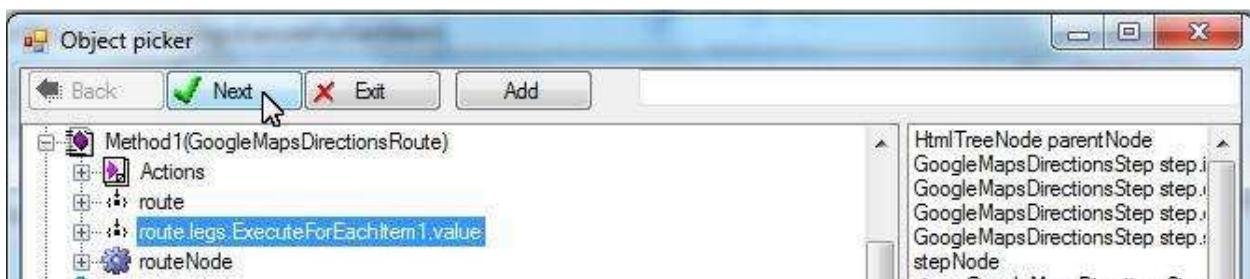


Use routeNode variable as the parent node:

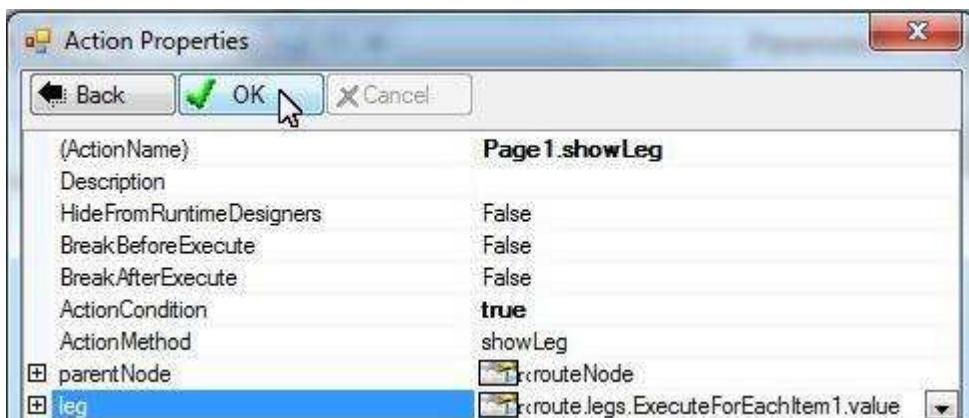


Select "value" for the leg:

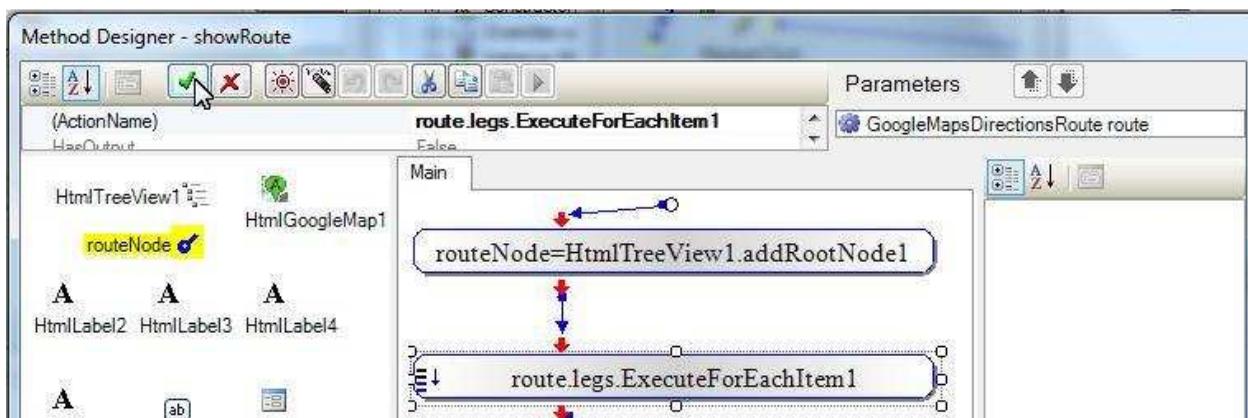
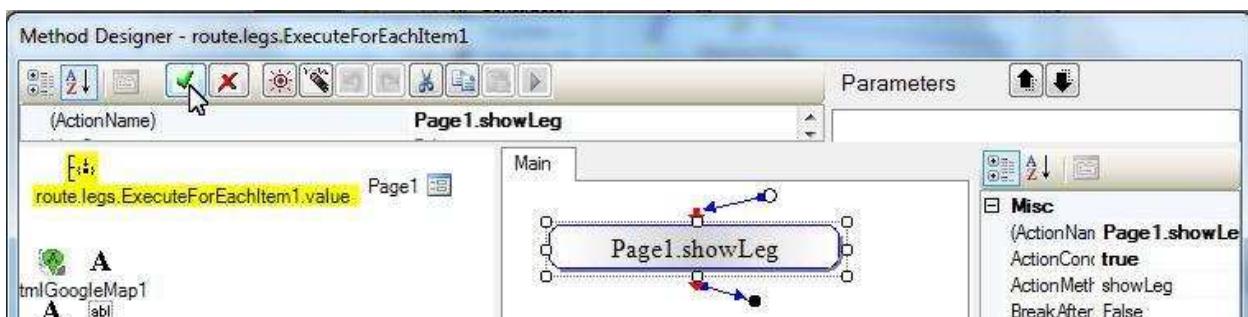




Click OK:



That is all for our sample:

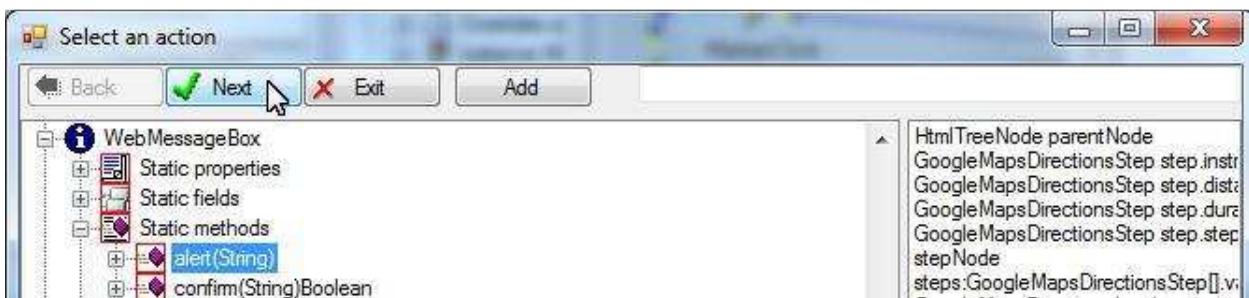
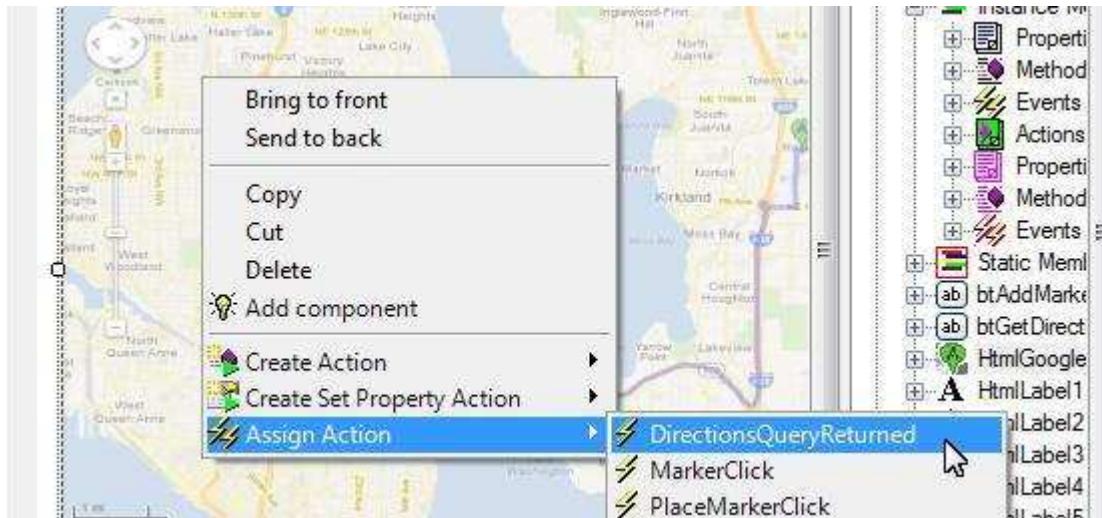


Handle Directions Query Event

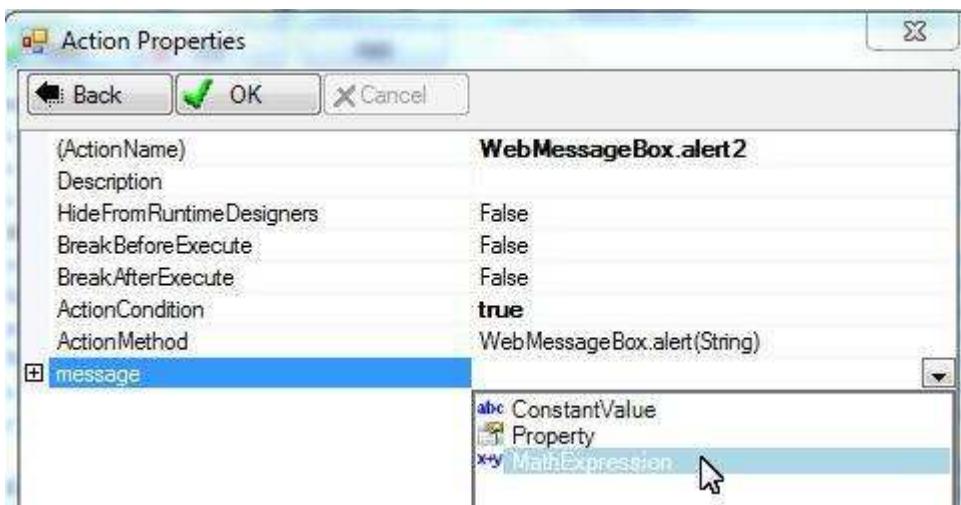
Event `DirectionsQueryReturned` occurs when Google Maps service responds to a `GetDirections` action. At this event we may display route.

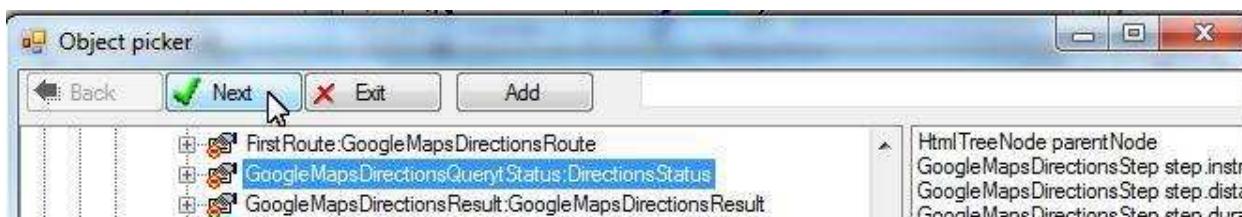
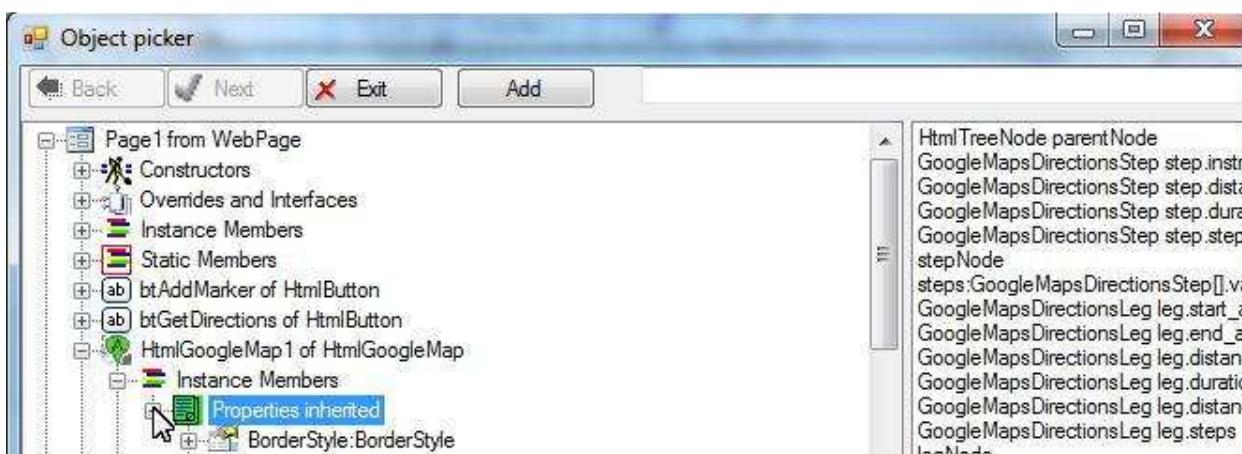
Show status

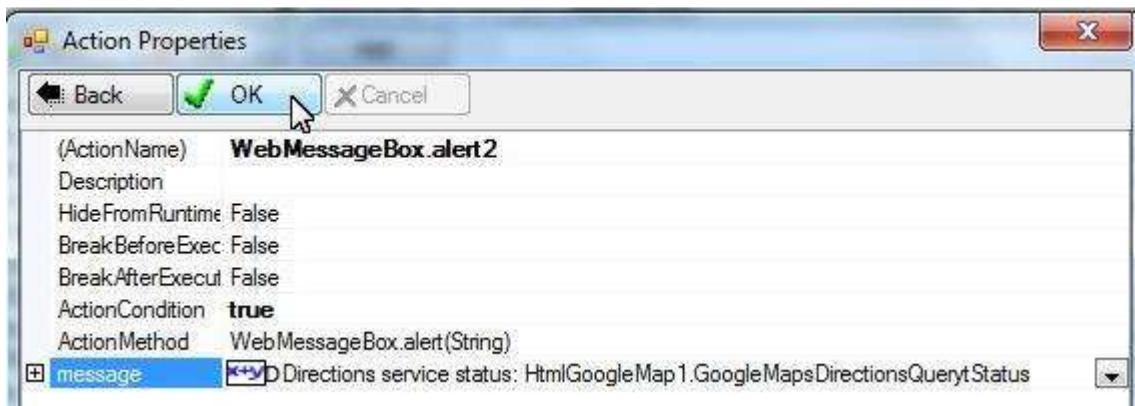
As a sample, we first use a message box to display directions service status:



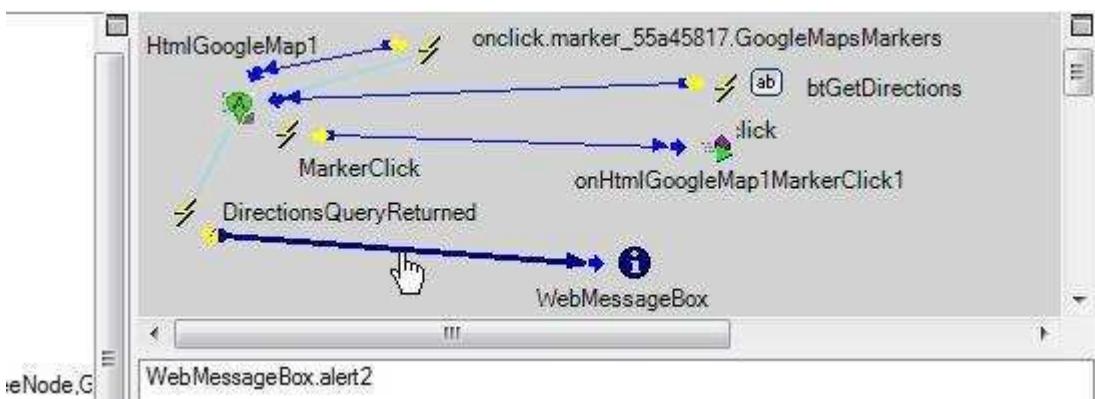
Form a display expression:





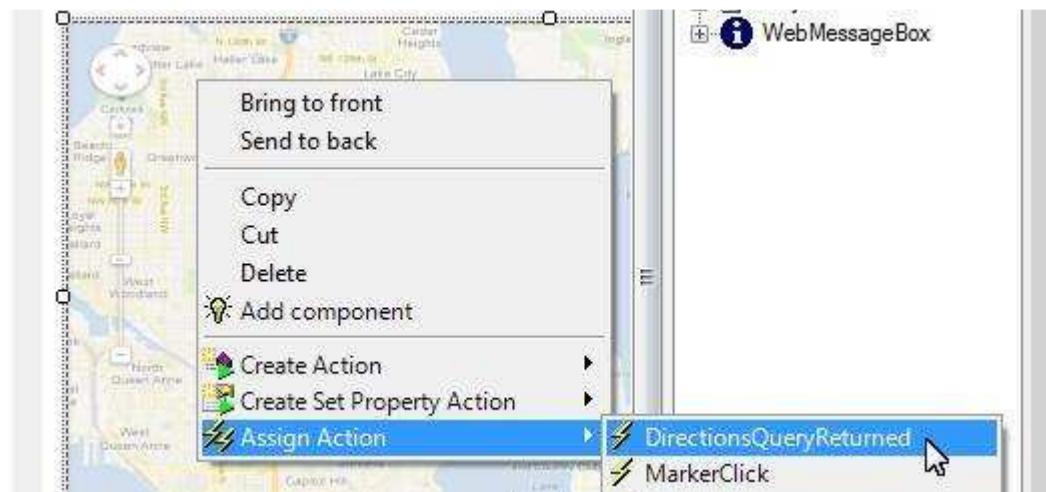


The action is created and assigned to the event:



Clear tree view

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



Select an action

Back Next Exit Add

HtmlTreeView1 of HtmlTreeView

Instance Members

- Properties inherited
- Methods inherited
- addChildNode(HtmlTreeNode, String, String, String, Object)HtmlTreeNode
- addChildNode To SelectedNode(String, String, String, Object)HtmlTreeNode
- addRootNode(String, String, String, Object)HtmlTreeNode
- clear
- deleteSelectedNode()Boolean

Action Properties

Back OK Cancel

(ActionName) **HtmlTreeView1.clear**

Description

HideFromRuntimeDesigners False

BreakBeforeExecute False

BreakAfterExecute False

ActionCondition **true**

ActionMethod **HtmlTreeView1 of HtmlTreeView.clear**

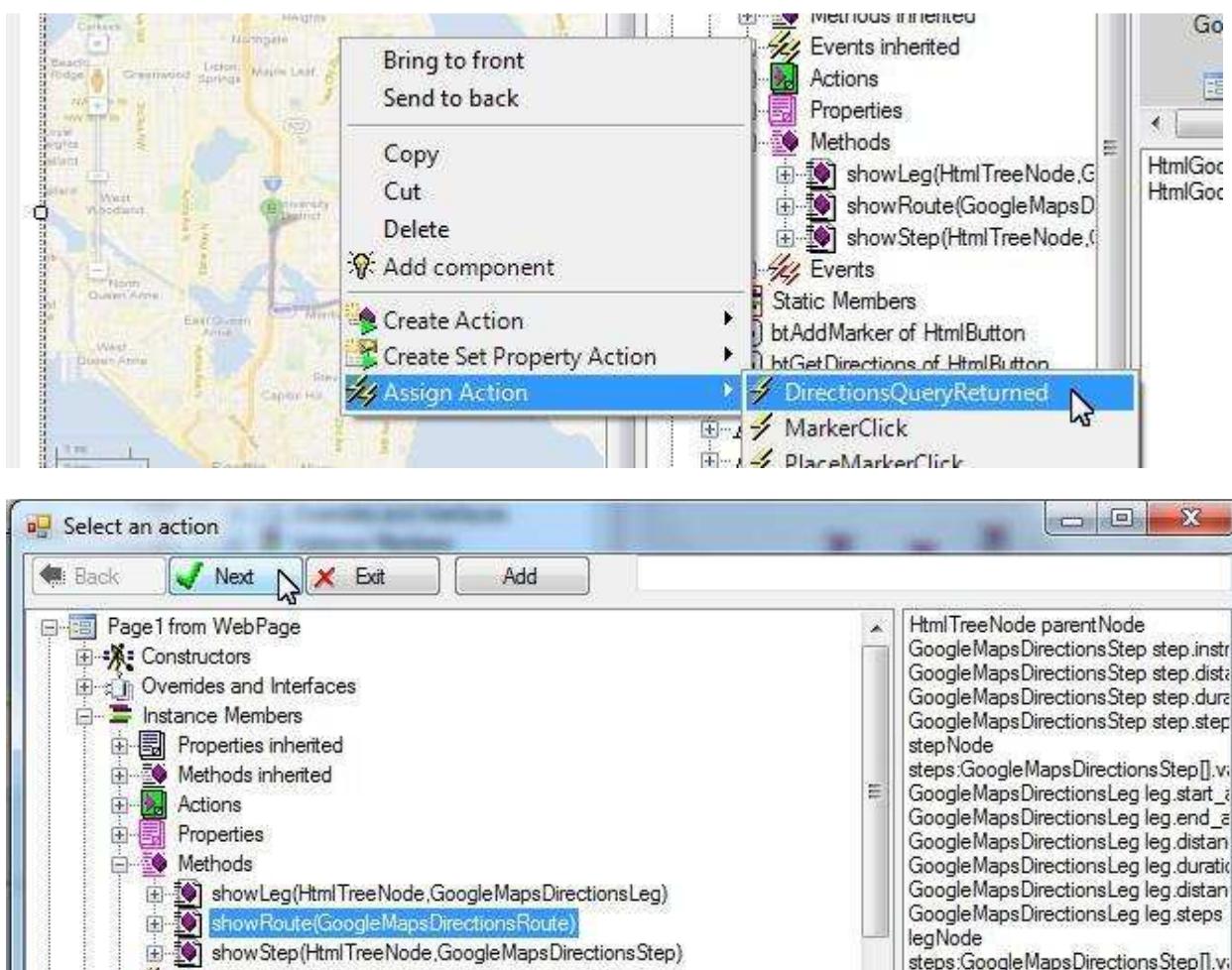
```

graph TD
    subgraph "Page"
        direction TB
        HGM1[HtmlGoogleMap1] --- OMC[MarkerClick]
        OMC --- DQR[DirectionsQueryReturned]
        DQR --- WM[WebMessageBox]
        WM --- HTV1[HtmlTreeView1]
        HTV1 --- C[clear]
    end
    subgraph "Interfaces"
        direction TB
        I1[btGetDir] --- OMC
        I2[click] --- OMC
        I3[onHtmlGoogleMap1MarkerClick1] --- DQR
        I4[DirectionsQueryReturned] --- WM
        I5[onbtAddMarkeronclick1] --- C
    end
    subgraph "Members"
        direction TB
        M1[btAddMarker] --- OMC
        M2[onbtAddMarkeronclick1] --- C
    end

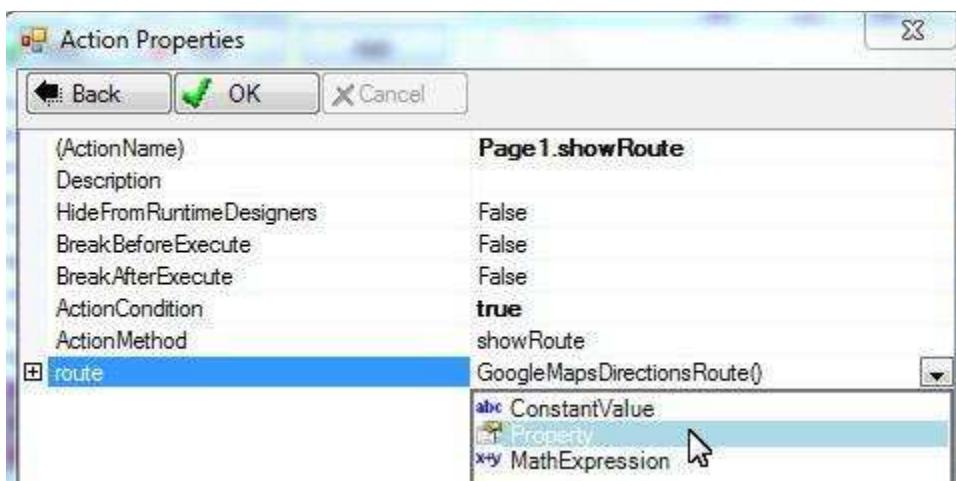
```

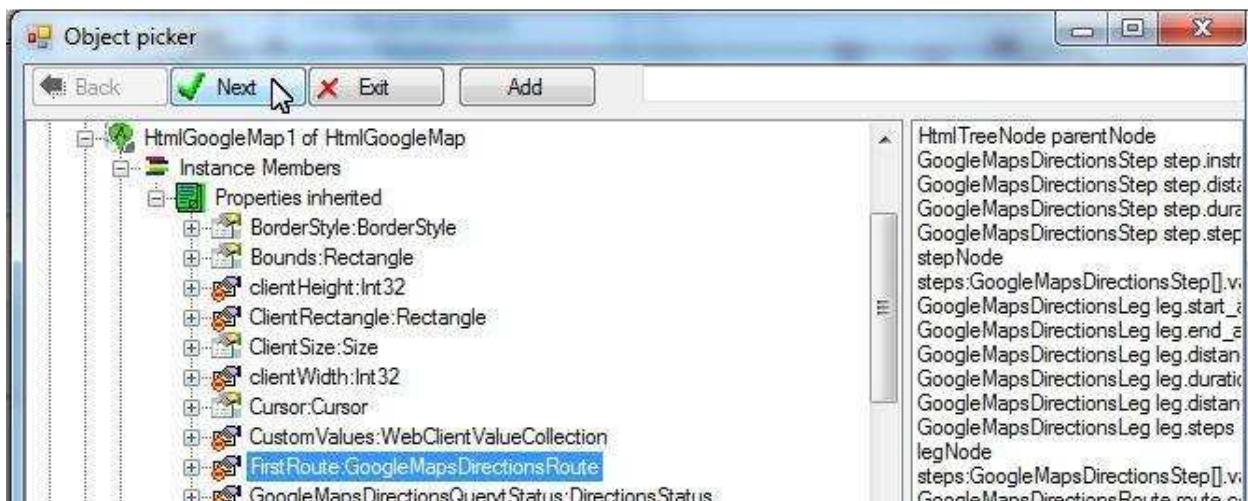
Show route in tree view

We create a showRoute action to display the first route:

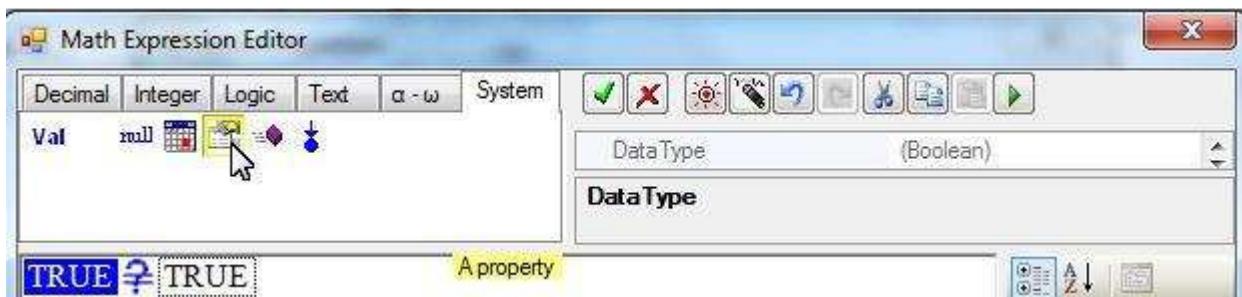
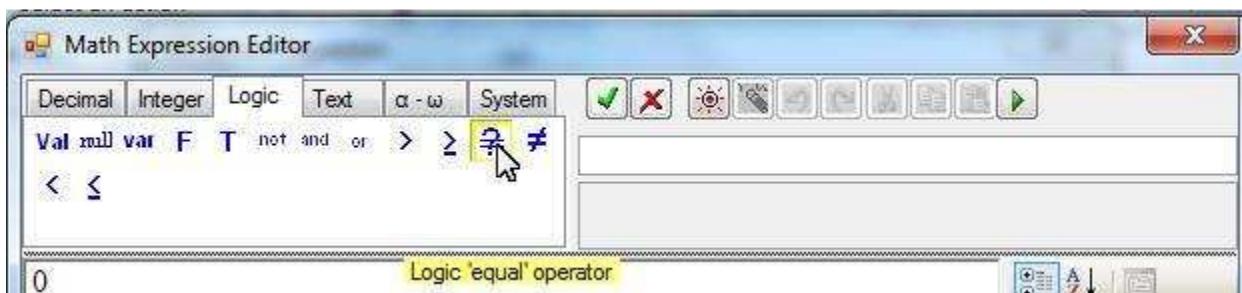
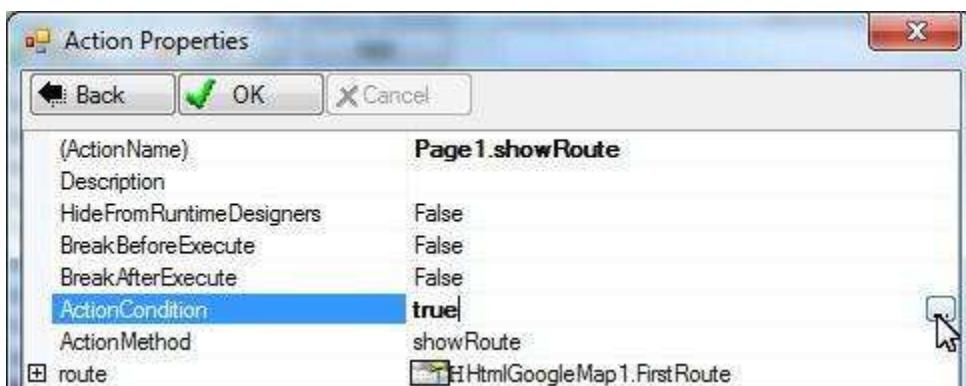


Select the first route:





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



Object picker

Math Expression Editor

Expr: `Html1GoogleMap1.GoogleMapsDirectionsQueryStatus` = TRUE

DataType: Boolean

Math Expression Editor

Expr: `Html1GoogleMap1.GoogleMapsDirectionsQueryStatus`

Finish the editing.

DataType: DirectionsStatus
RunAt: Inherit
Value: OK
Value Type: DirectionsStatus

Action Properties

ActionName: Page1.showRoute

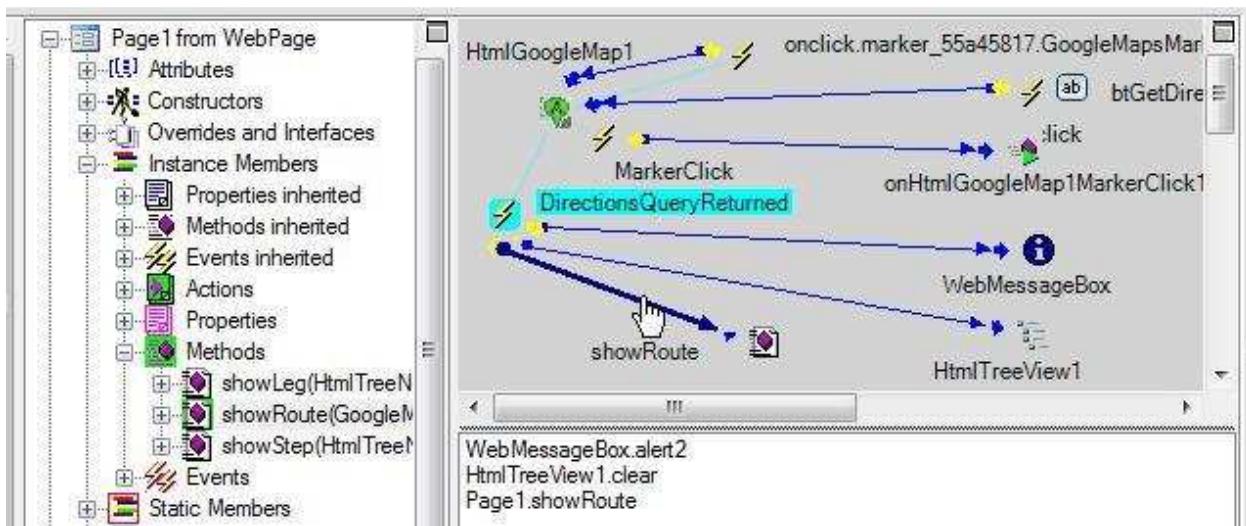
Description: HideFromRuntime: False
BreakBeforeExecute: False
BreakAfterExecute: False

ActionCondition: `Html1GoogleMap1.GoogleMapsDirectionsQueryStatus = OK`

ActionMethod: showRoute

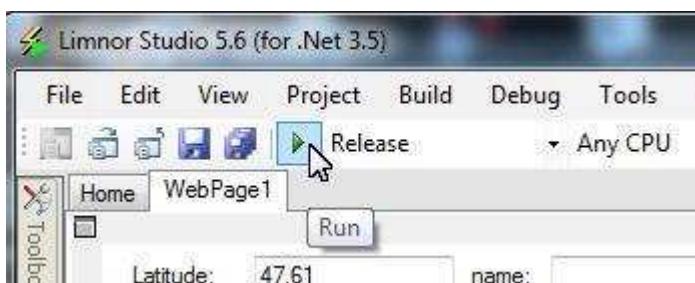
route: `Html1GoogleMap1.FirstRoute`

The action is created and assigned to the event.

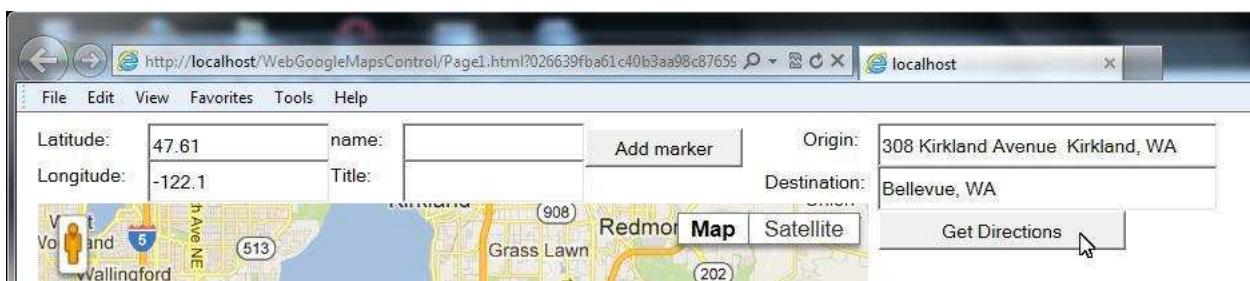


Test

We may test directions services now.



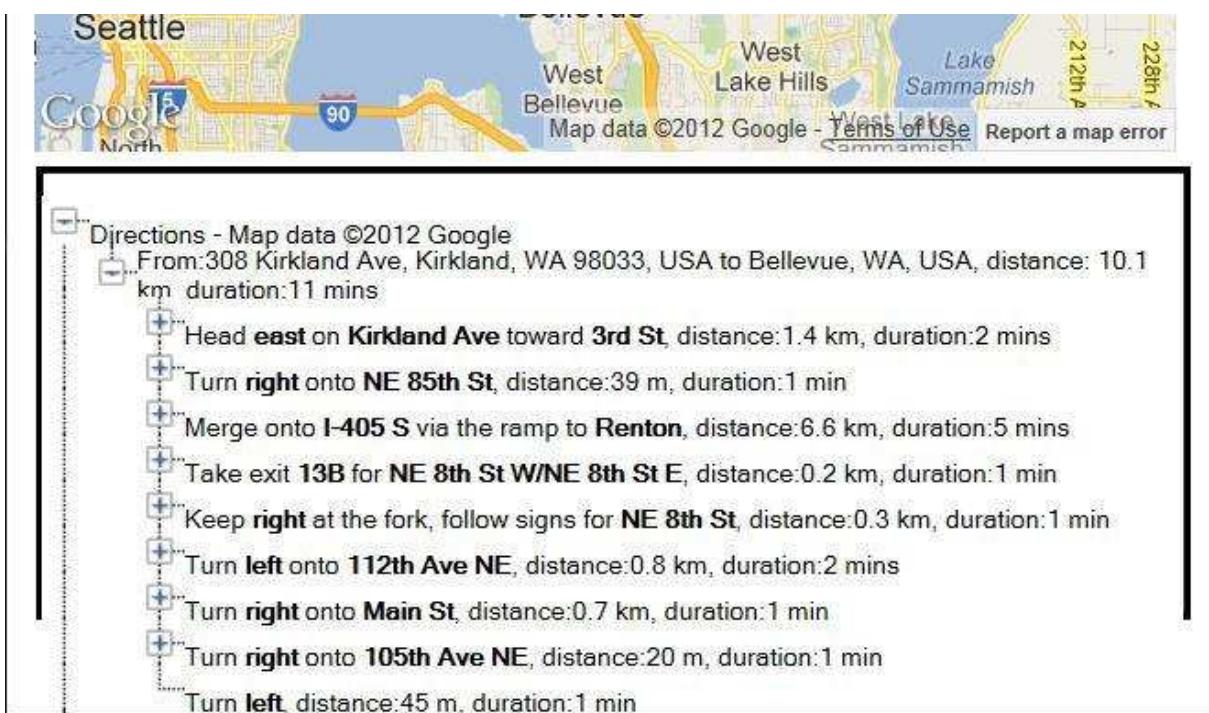
The web page appears. Enter origin and destination. Click “Get Directions”.



A message box appears:



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



Directions - Map data ©2012 Google
From: 308 Kirkland Ave, Kirkland, WA 98033, USA to Bellevue, WA, USA, distance: 10.1 km duration:11 mins
Head east on Kirkland Ave toward 3rd St, distance:1.4 km, duration:2 mins
Turn right onto NE 85th St, distance:39 m, duration:1 min
Merge onto I-405 S via the ramp to Renton, distance:6.6 km, duration:5 mins
Take exit 13B for NE 8th St W/NE 8th St E, distance:0.2 km, duration:1 min
Keep right at the fork, follow signs for NE 8th St, distance:0.3 km, duration:1 min
Turn left onto 112th Ave NE, distance:0.8 km, duration:2 mins
Turn right onto Main St, distance:0.7 km, duration:1 min
Turn right onto 105th Ave NE, distance:20 m, duration:1 min
Turn left, distance:45 m, duration:1 min

Places Search

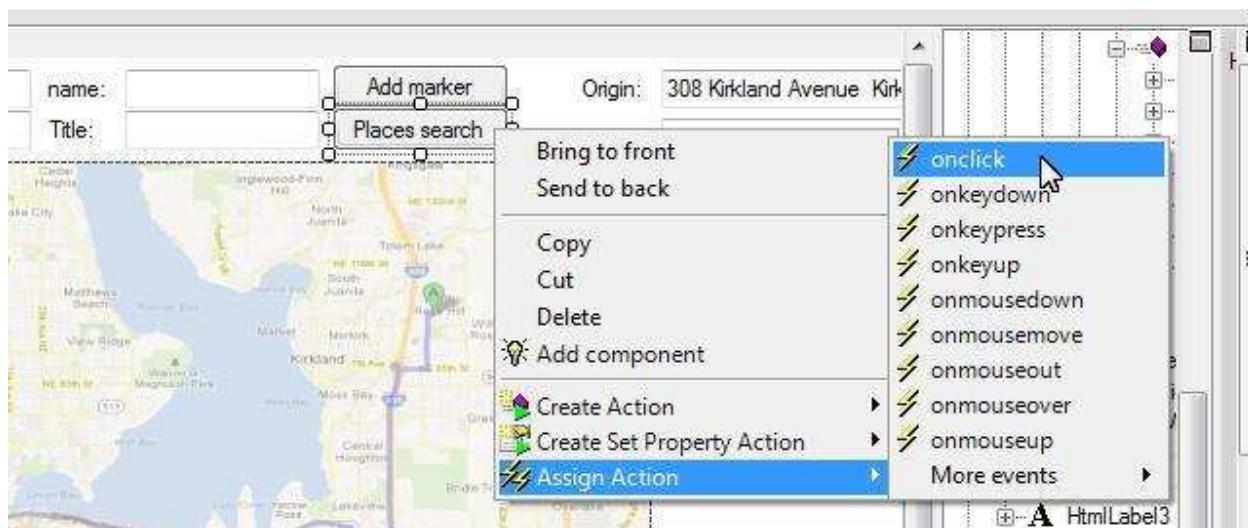
HtmlGoogleMap may be used to search places.

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

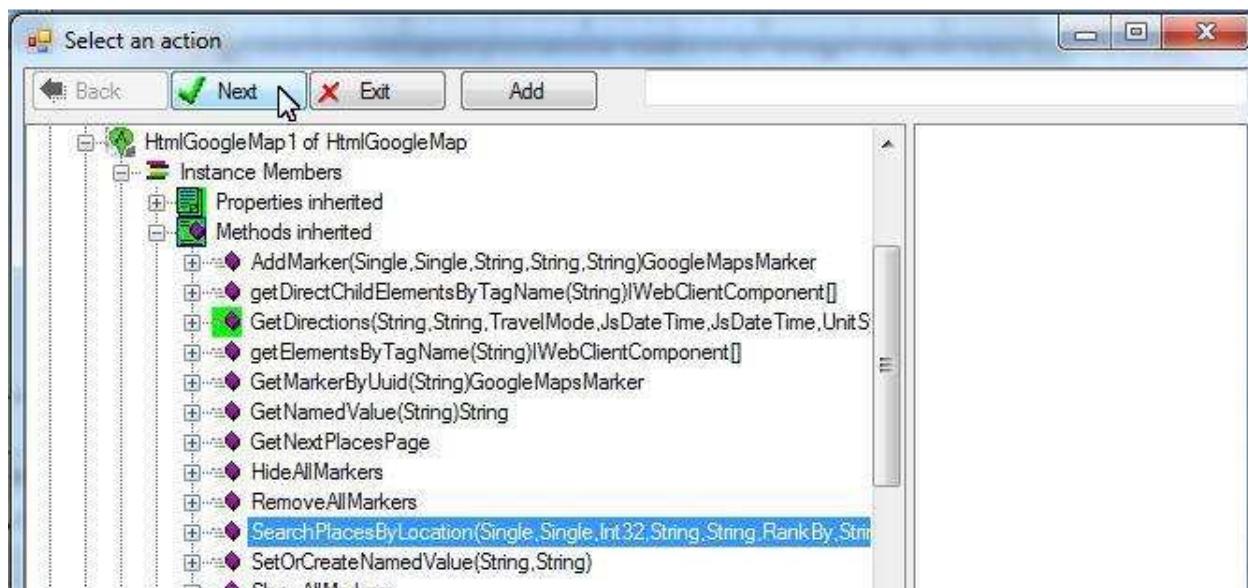
- Event `PlacesQueryReturned` occurs when Google Maps services respond to search request and get-next-page request.
- Property `GoogleMapsPlacesQuerytStatus` indicates status on Google Maps responding to search request and get-next-page request.
- Google Maps services return places one page at a time. 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

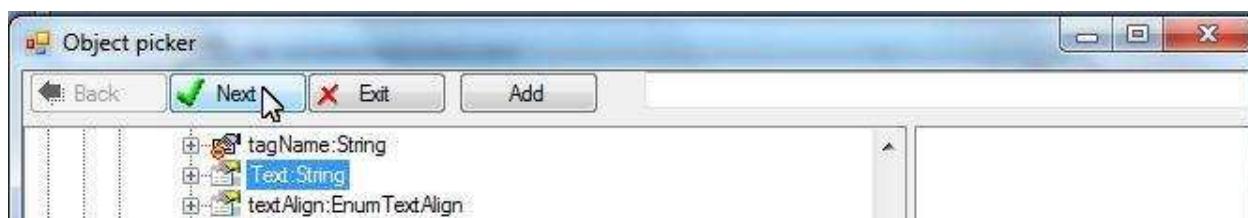
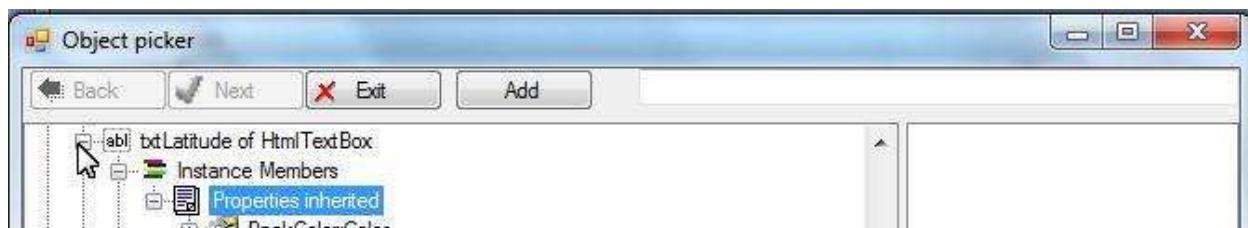
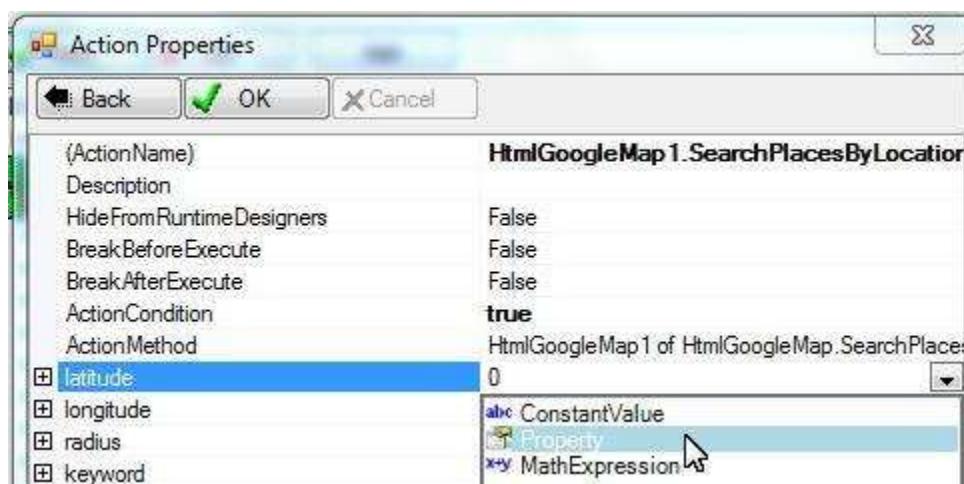
Suppose we use a button to initiate a places-search:

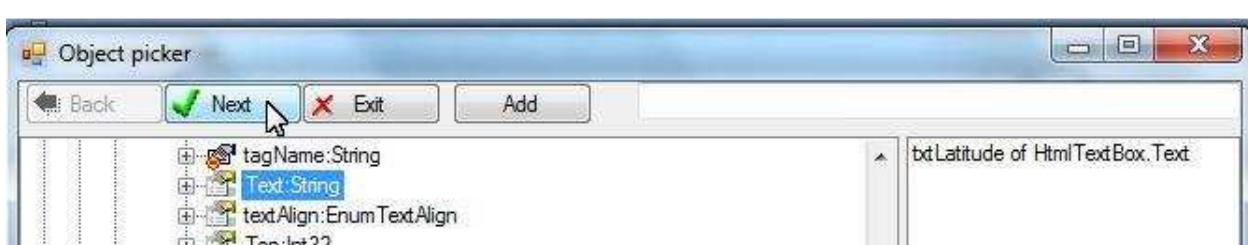
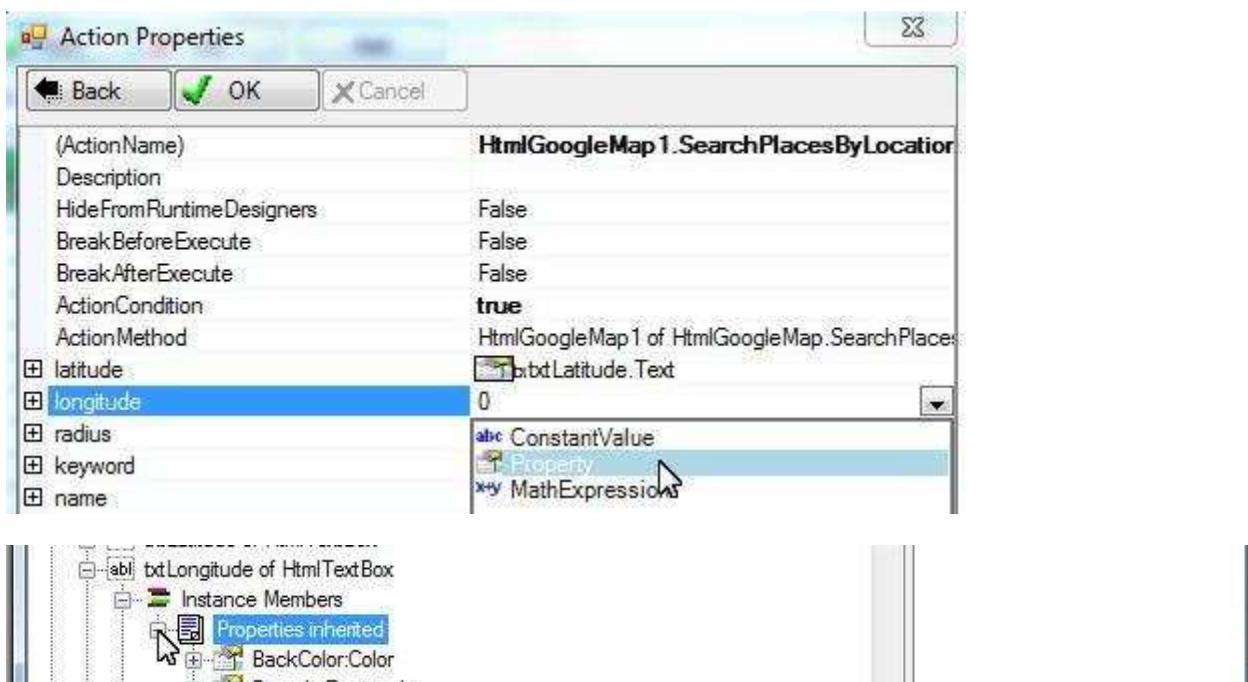


Select `SearchPlacesByLocation`:



Set search center and radius:

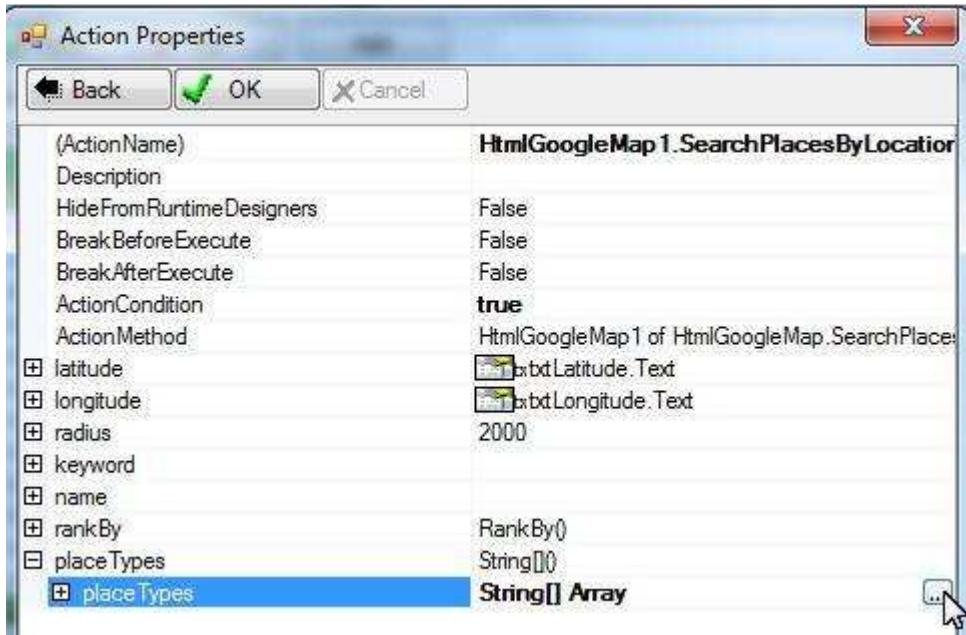




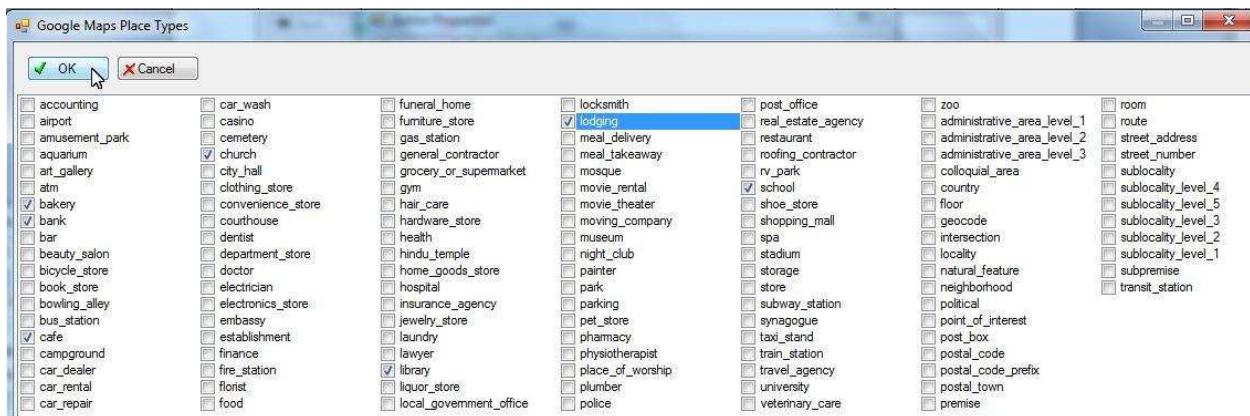
Radius is in meters.



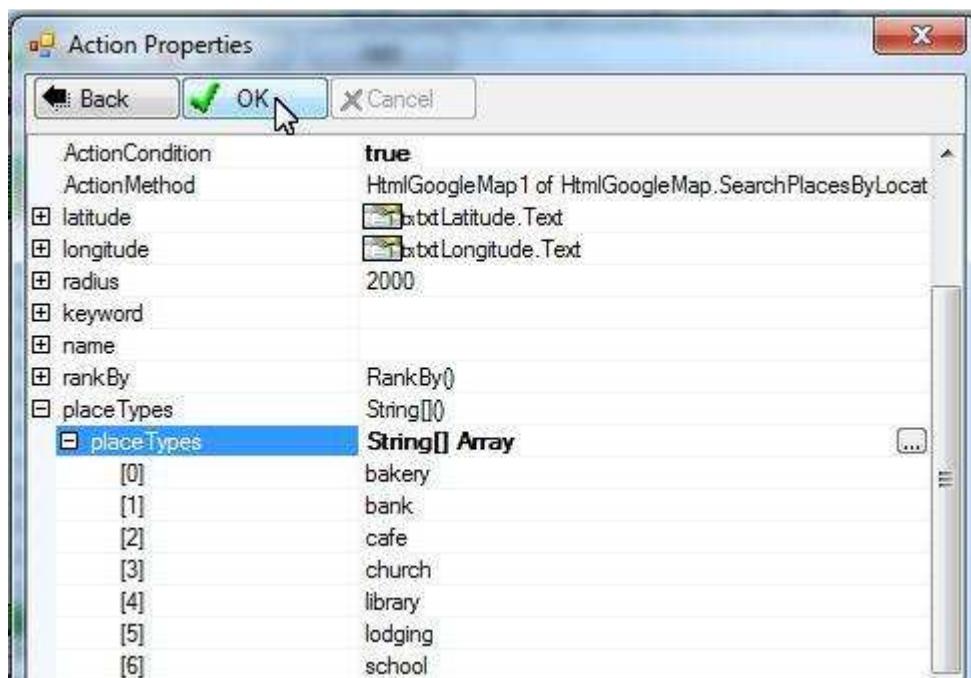
Select places to be searched:



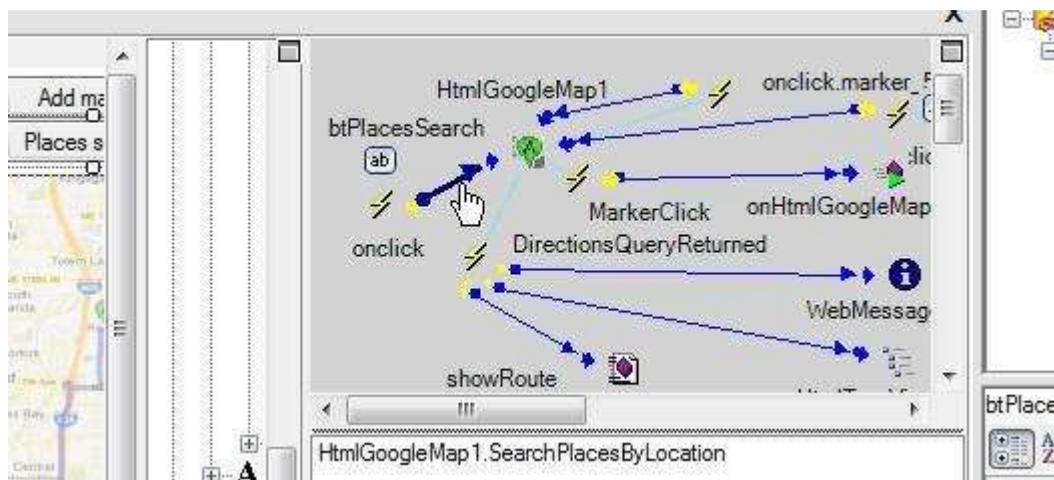
Check all places to be searched:



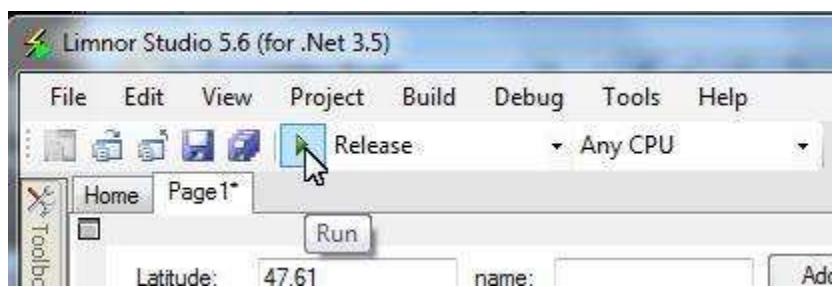
Click OK to create the action:



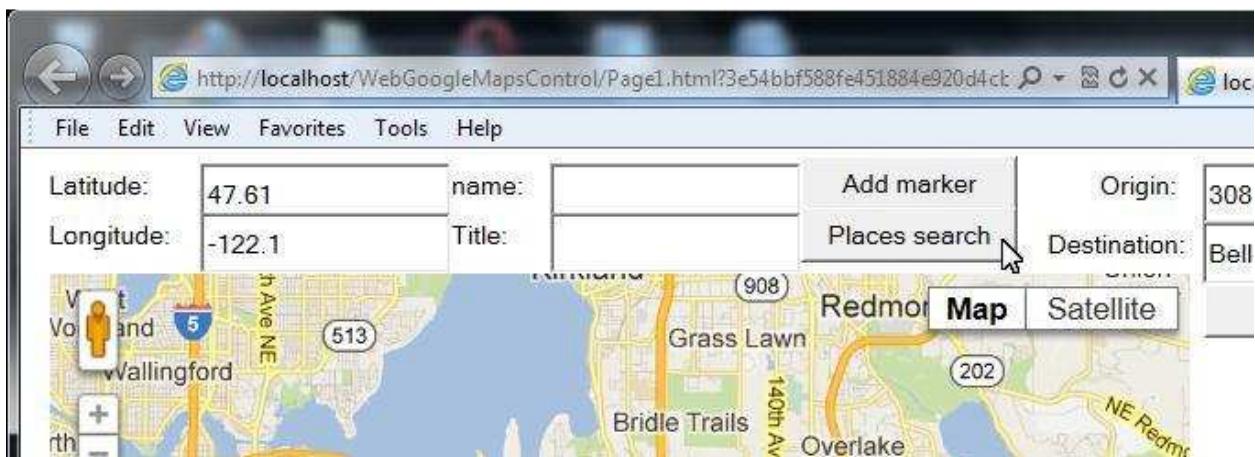
The action is created and assigned to the button:



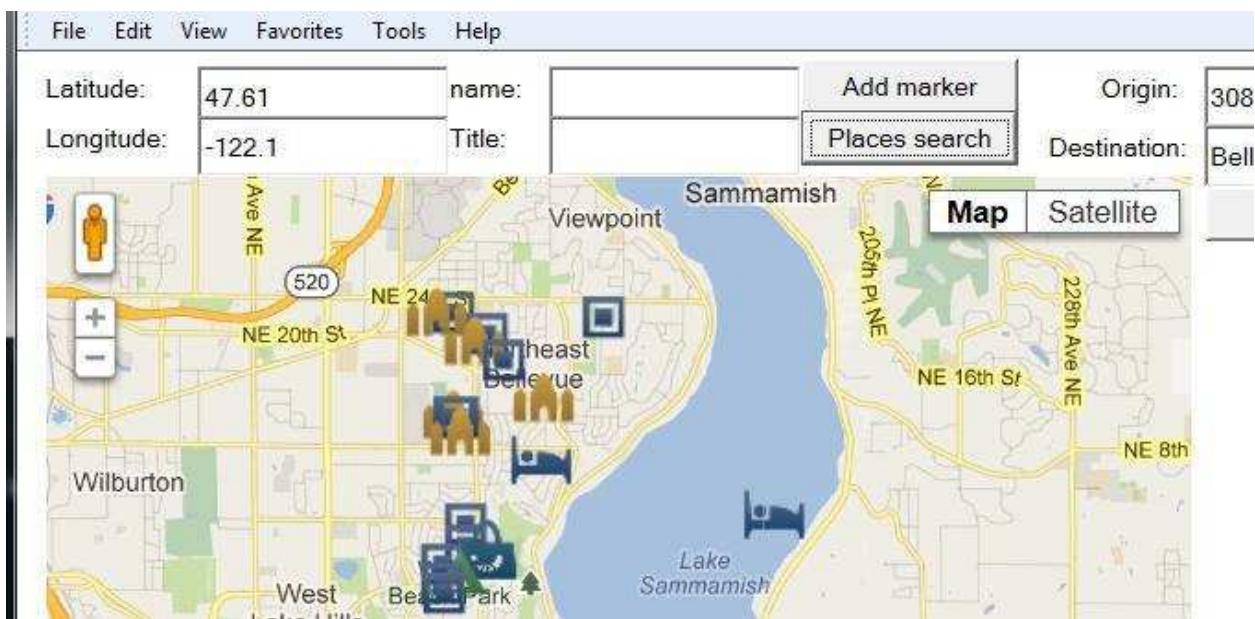
We can test the web page now:



The web page appears. Click "Places search":



Found places appear on the map:



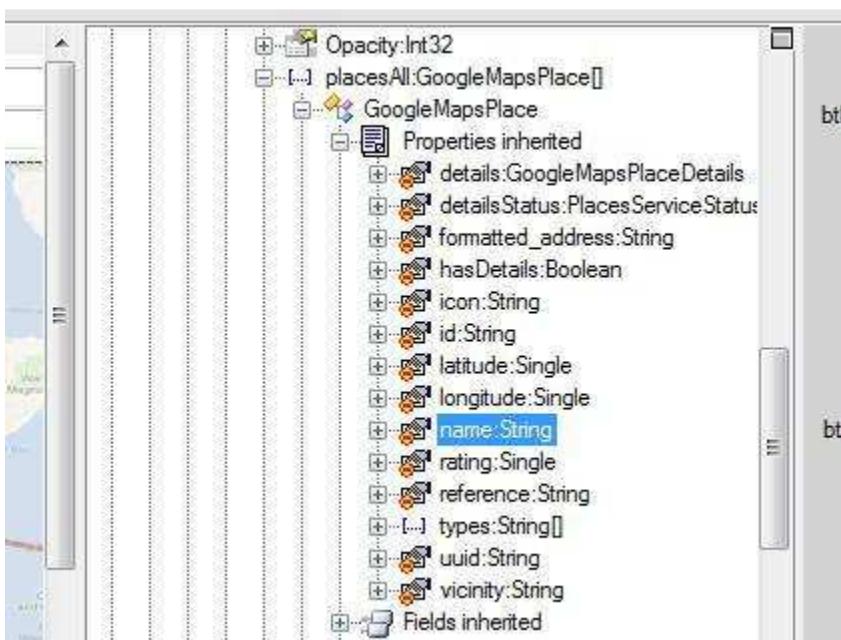
Note that these places are from the first page. We need to use a `GetNextPlacesPage` action to get the next page. We will do it later.

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:

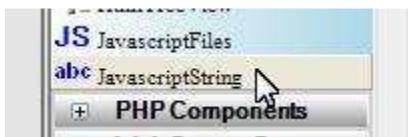


Suppose we want to display latitude, longitude, name, rating, and vicinity.

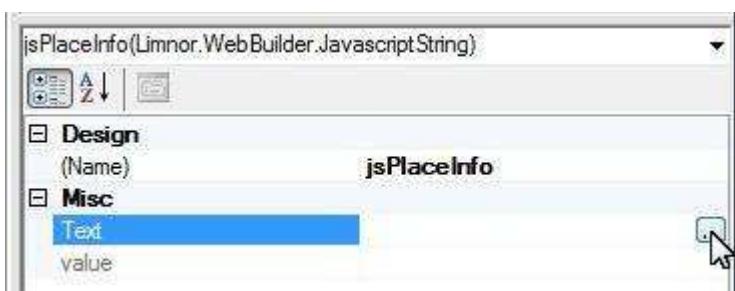
An information window may display HTML contents. In previous examples, we have used expressions to form text for information windows. Here we introduce another way to form HTML text for information windows.

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:



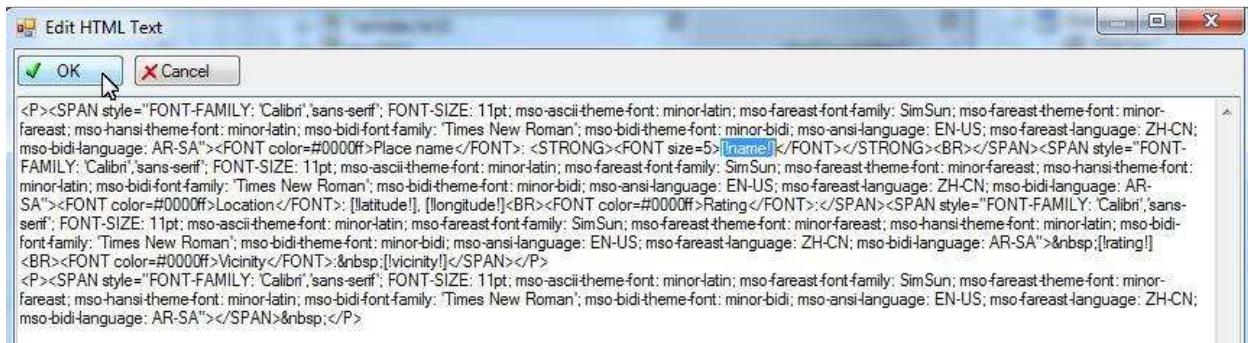
Set its Text property to create HTML contents:





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 [

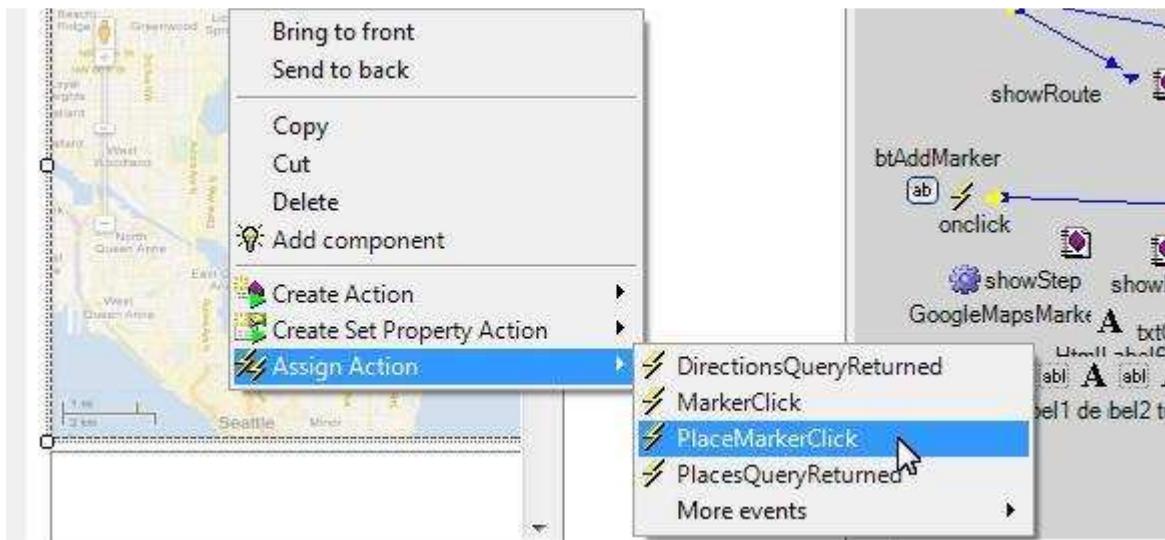


Each field will become a property:

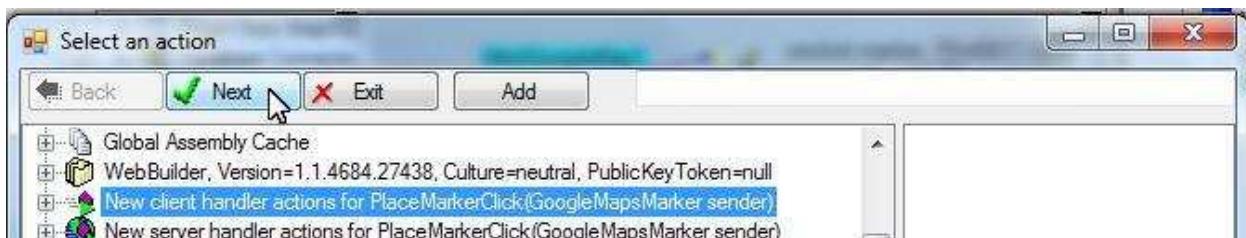


If a field you want to create does not appear as a property then there are HTML tags getting into its field markings. You may use “Edit HTML” to move those HTML tags out of [! and !].

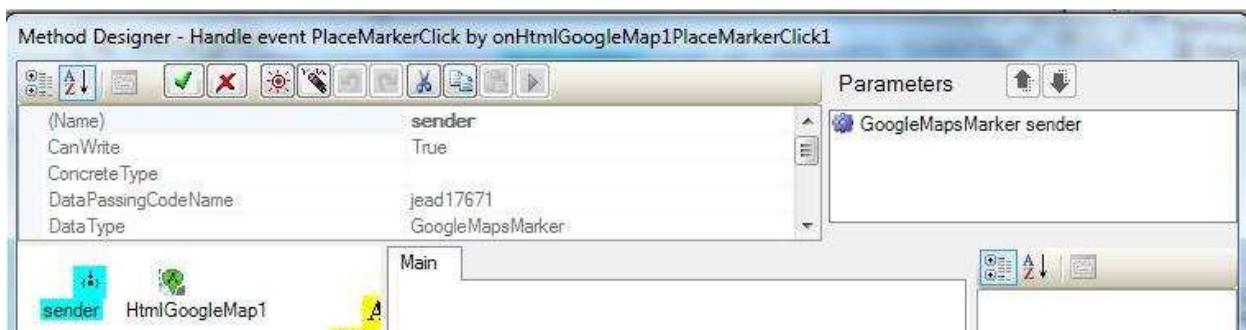
Now let's handle event `PlaceMarkerClick` to show an information window:



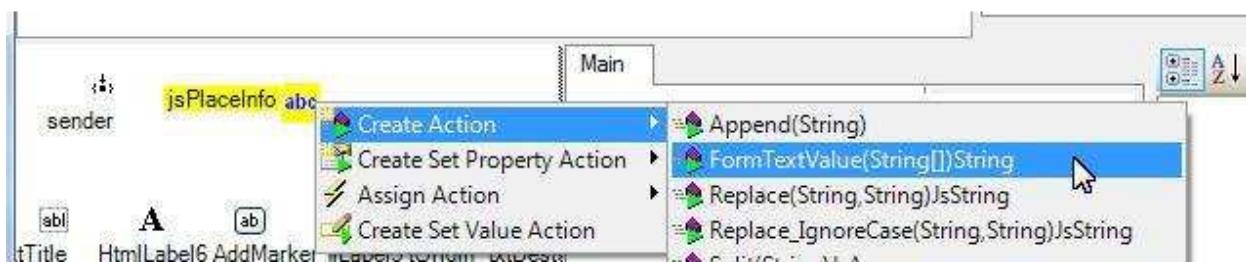
Select “New client handler actions for PlaceMarkerClick”:



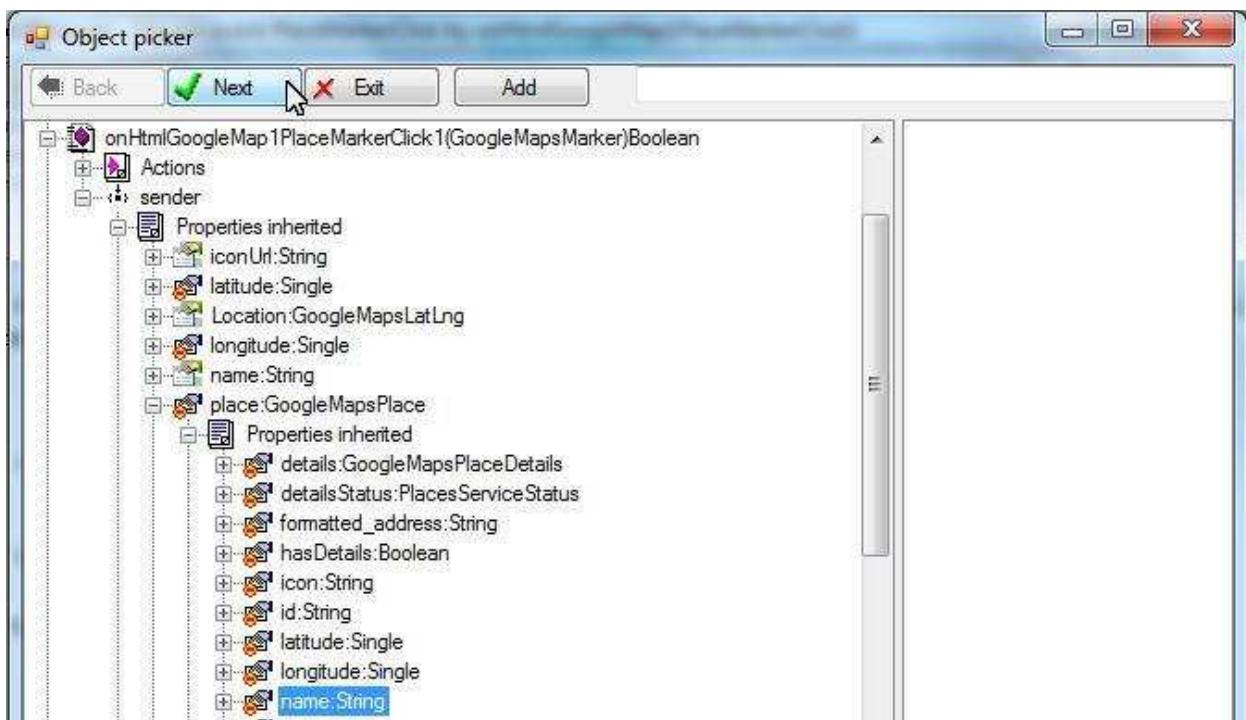
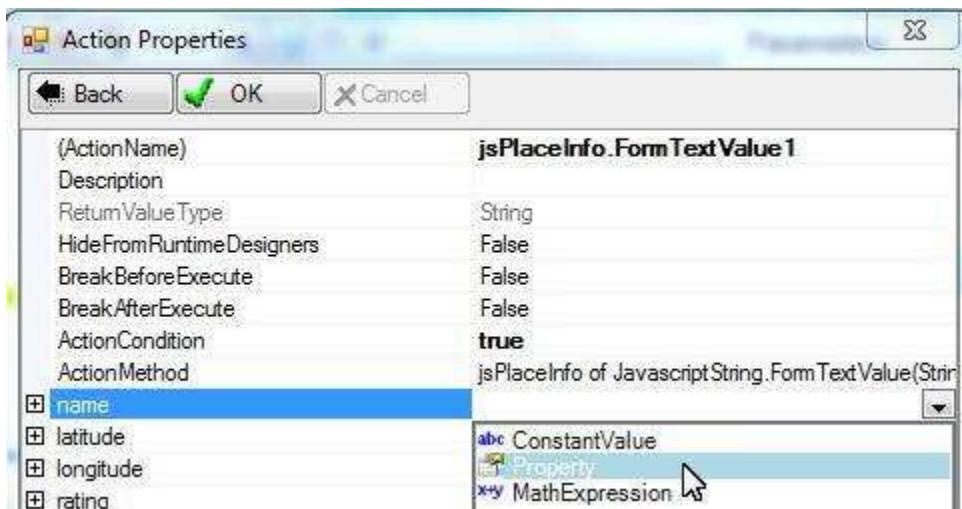
A method editor appears. Note that “sender” is the marker being clicked.

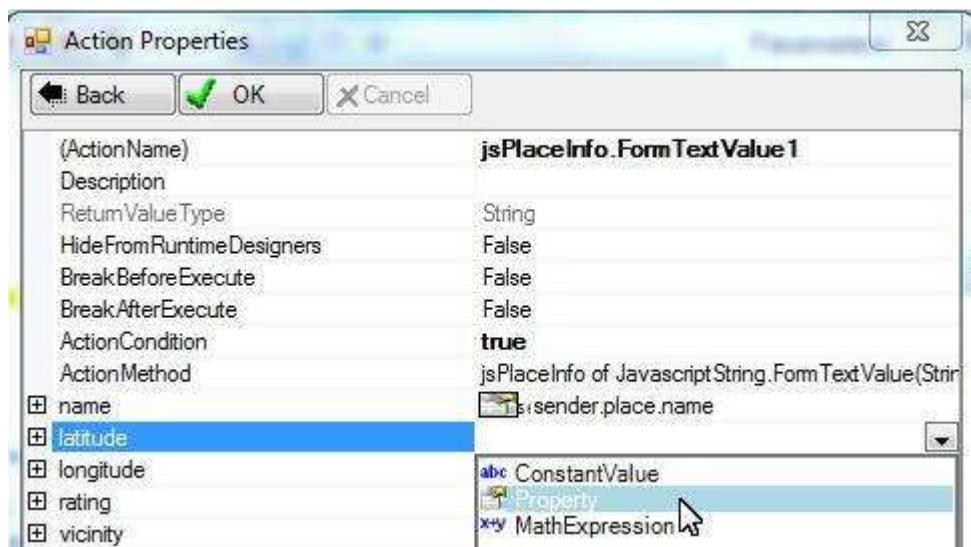


Let's use `JavascriptString` to form display information:



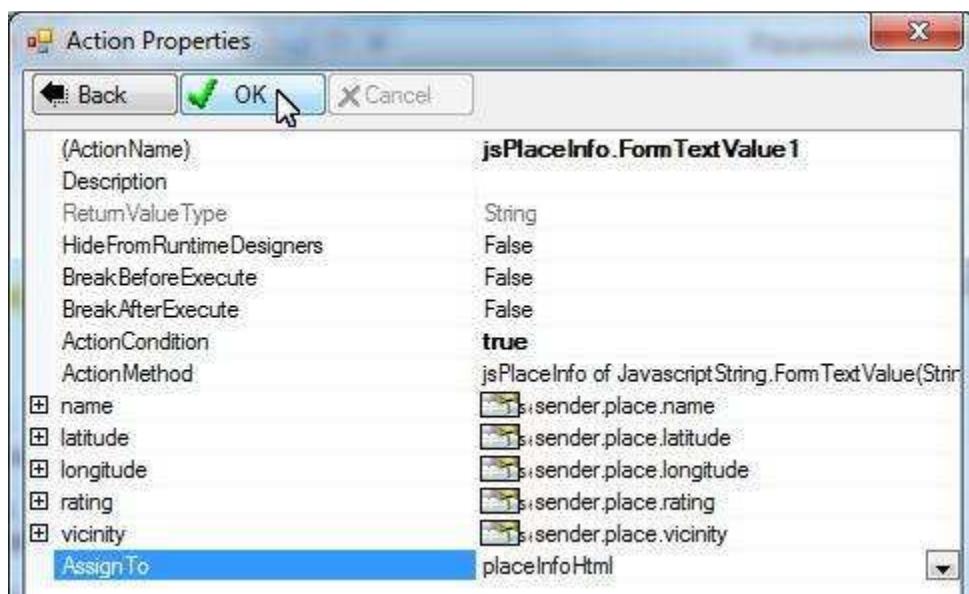
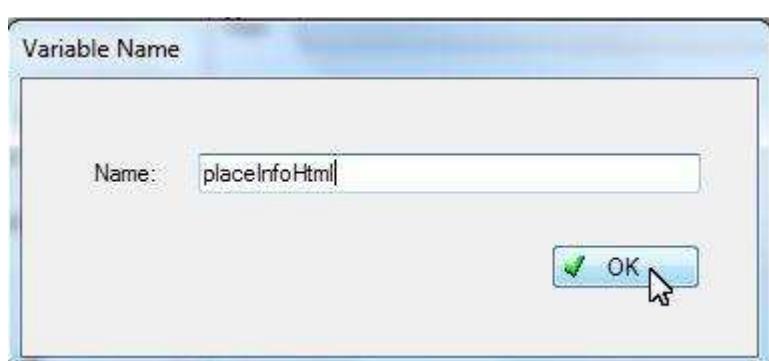
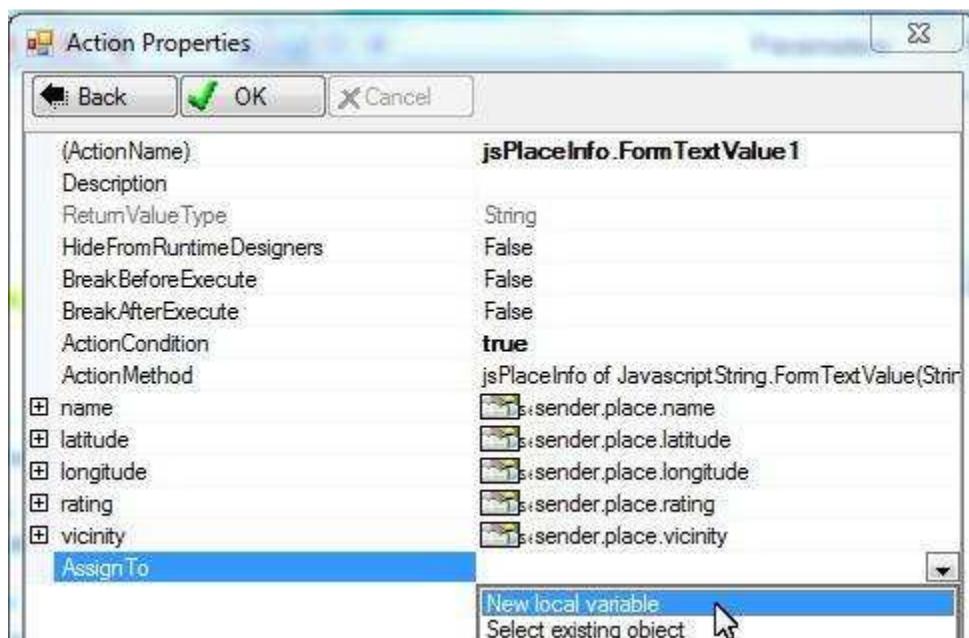
Pass properties of the place to the action:



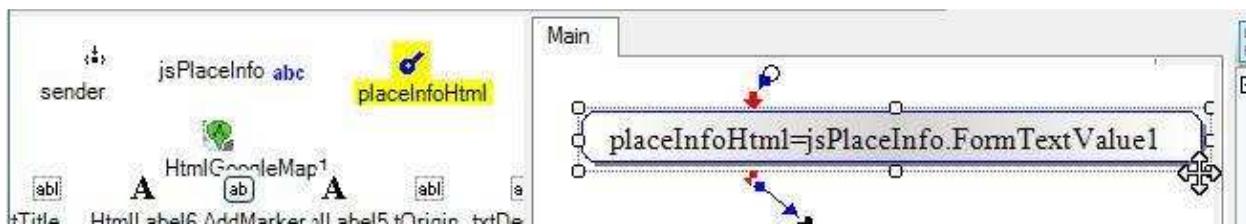


Click a recent selection to help quickly locate a property we want to select:

Use a new variable for the information it forms:



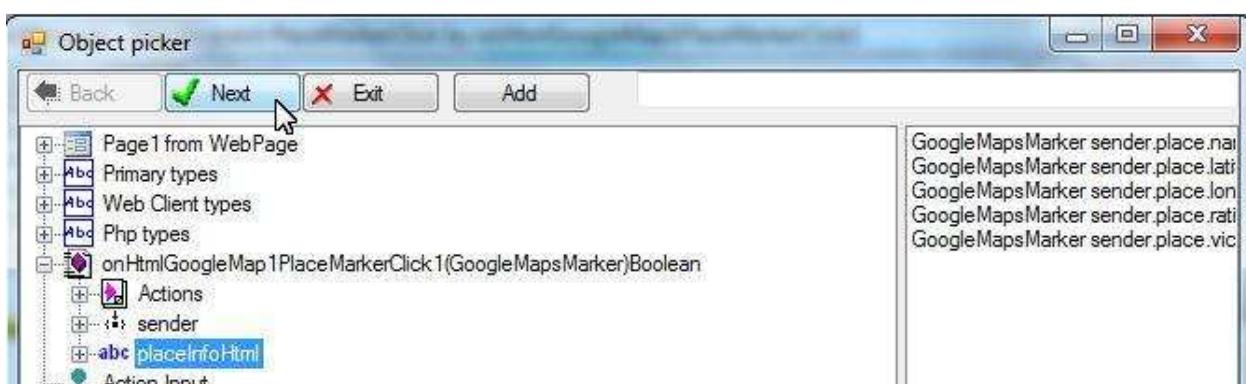
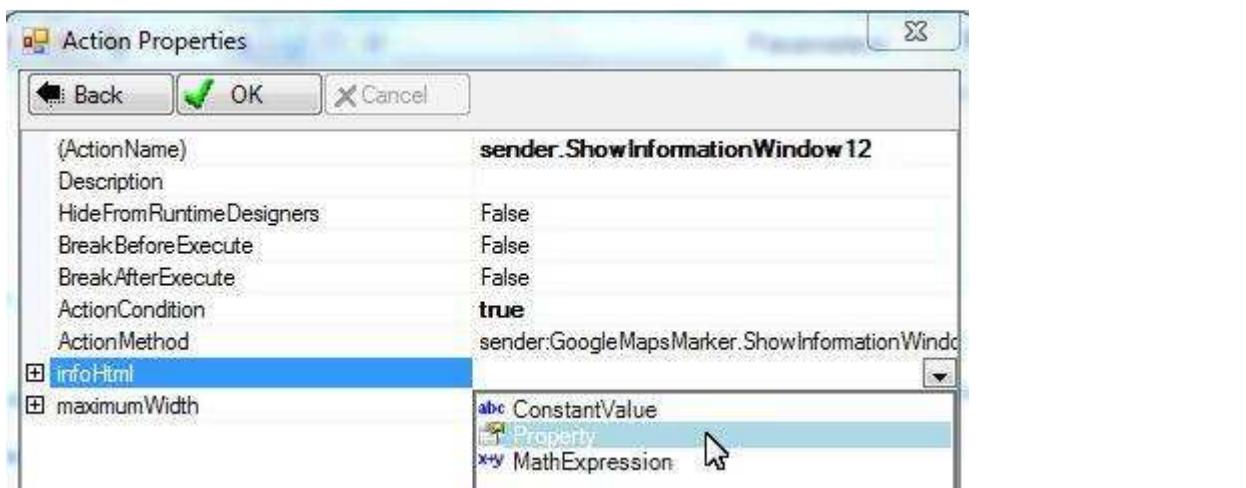
The action and variable appear:



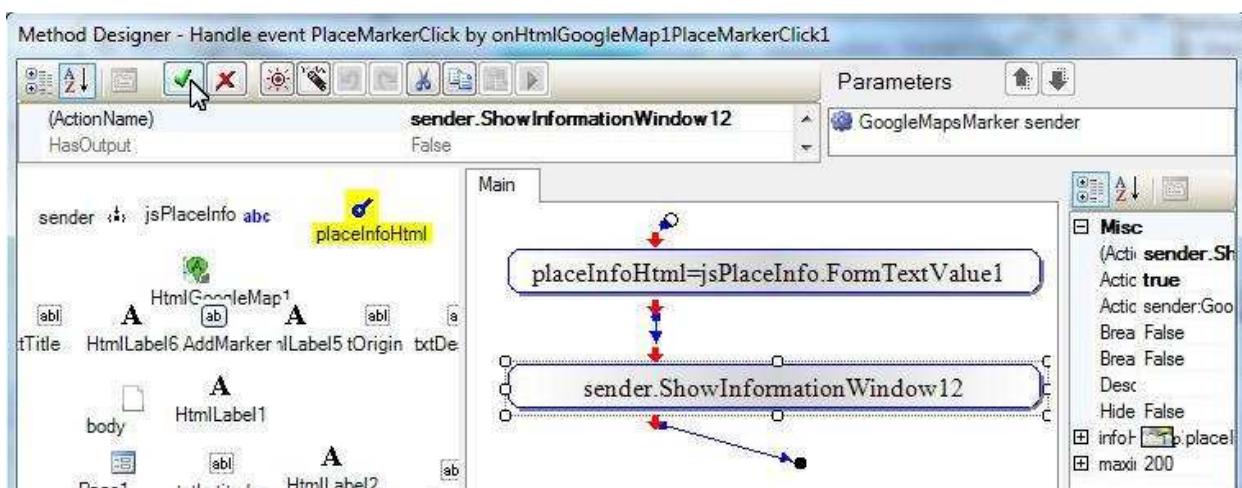
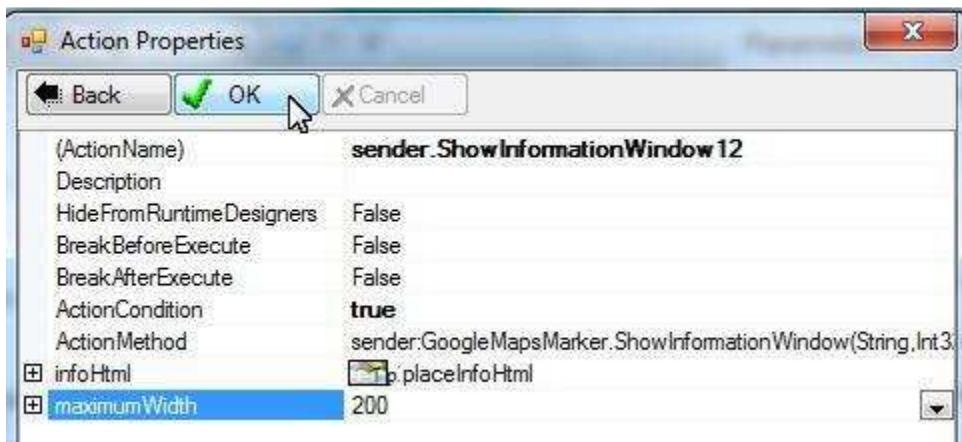
Create an action to show information window:



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:



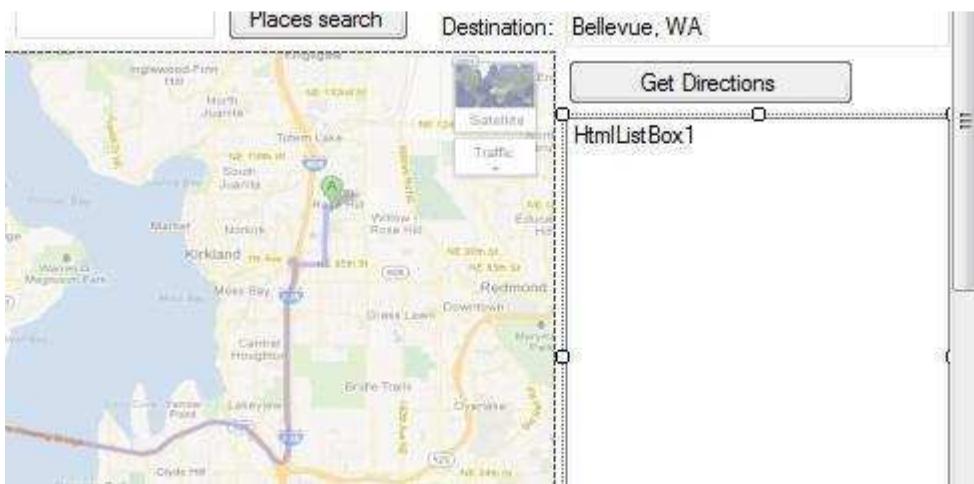
Information window appears:



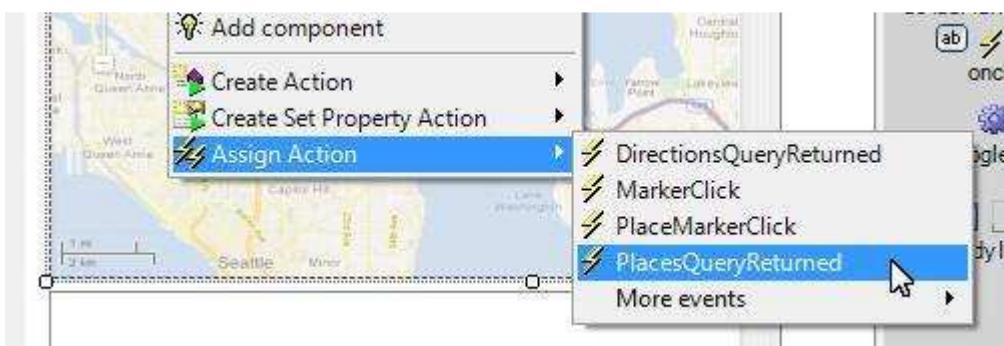
List places

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.

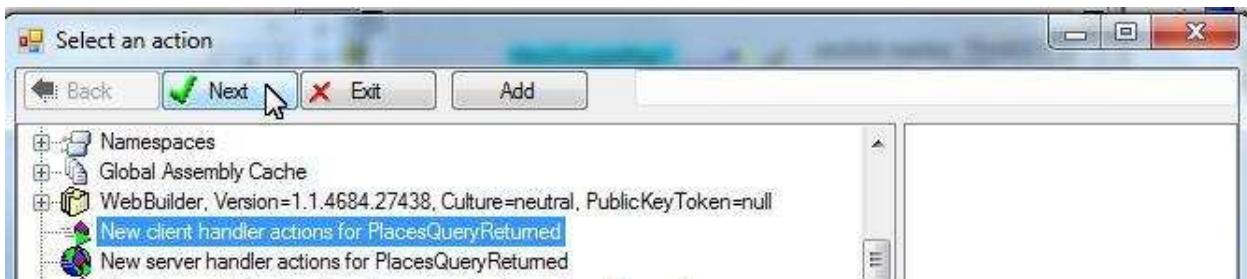
Add a list box to the web page:



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

The screenshot shows the Method Designer interface for handling the event `PlacesQueryReturned`. In the top panel, the method is named `onHtmlGoogleMap1PlacesQueryReturned1`. A context menu is open over a node in the main workspace, with the option `Create condition` highlighted.

The main workspace contains a flowchart node labeled `Condition1`. This node has three outgoing branches: one labeled `Yes`, one labeled `No`, and one unlabeled branch pointing upwards. The `Yes` branch leads to a `Math Expression Editor` window.

The `Math Expression Editor` window displays the expression `Logic 'equal' operator`. The toolbar above it includes buttons for Decimal, Integer, Logic, Text, α-ω, System, and various operators like Val, null, F, T, not, and, or, >, ≥, ≠, =, <, ≤, and &. The expression `=` is currently selected.

To the right of the editor, a `Misc` panel shows a tree structure under `Condition`. The `False` branch is expanded, showing options for ConstantValue, Property, and MathExpression. The `MathExpression` option is selected.

Below the editor, the expression `TRUE` is shown next to the `=` operator, indicating the current state of the logic being edited.

Object picker

```

Page1 from WebPage
  Constructors
  Overrides and Interfaces
  Instance Members
  Static Members
  ab btAddMarker of HtmlButton
  ab btGetDirections of HtmlButton
  ab btPlacesSearch of HtmlButton
  HtmlGoogleMap1 of HtmlGoogleMap
    Instance Members
      Properties inherited
        BorderStyle:BorderStyle
  
```

Object picker

GoogleMapsPlacesHasNextPage: Boolean
GoogleMapsPlacesQueryStatus: PlacesServiceStatus
Height: Int32

Math Expression Editor

Val null

DataType: Boolean

HtmlGoogleMap1.GoogleMapsPlacesQueryStatus = TRUE

Math Expression Editor

Finish the editing.

Val null

DataType	(PlacesServiceStatus)
RunAt	Inherit
Value	OK
ValueType	(PlacesServiceStatus)

HtmlGoogleMap1.GoogleMapsPlacesQueryStatus = OK

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

The screenshot illustrates a workflow or script editor interface. At the top, a decision diamond is labeled "places search OK". A "Yes" branch leads down, while a "No" branch leads to a context menu. The context menu options are:

- Add local variable
- Add an action** (selected)
- Create condition
- Execute actions repeatedly
- Execute actions a number of times

A secondary window titled "Select an action" is open, showing a tree structure under "WebMessageBox". The "Static methods" node is expanded, revealing "alert(String)" and "confirm(String)Boolean".

A third window, "Action Properties", is also open, showing properties for "WebMessageBox.alert3". The "ActionMethod" property is set to "WebMessageBox.alert(String)". The "message" field is expanded, showing a dropdown menu with "ConstantValue", "Property", and "MathExpression".

A fourth window, "Math Expression Editor", is open at the bottom. It has tabs for Decimal, Integer, Logic, Text, and System. The Text tab is selected. The input field contains the expression "A +". Below the input field is the text "Concatenate two strings together to form a new string".

Math Expression Editor

Val null

Data Type: (String)

Data Type

"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
- Height:Int32

HtmlGoogleMap1 of HtmlGoogleMap

Math Expression Editor

Finish the editing.

Val null

RunAt: Client
TaskId: 0

"Places query failed." HtmlGoogleMap1.GoogleMapsPlacesQuerytStatus

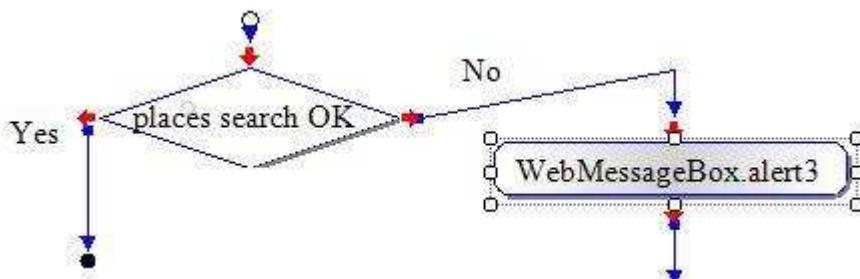
Action Properties

Back OK Cancel

(ActionName) WebMessageBox.alert3

Description
HideFromRuntime: False
BreakBeforeExec: False
BreakAfterExec: False
ActionCondition: true
ActionMethod: WebMessageBox.alert(String)

message: Places query failed:HtmlGoogleMap1.GoogleMapsPlacesQuerytStat

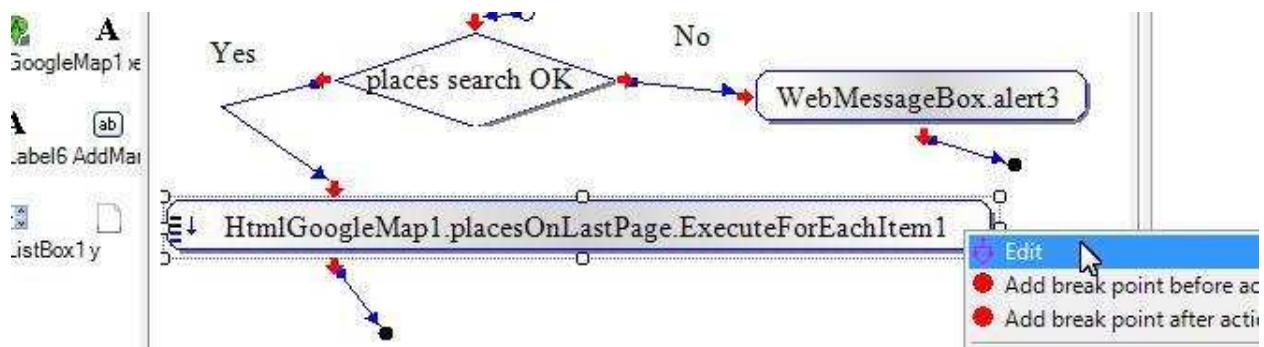


Create an “Execute actions for all items” action for placesOnLastPage:

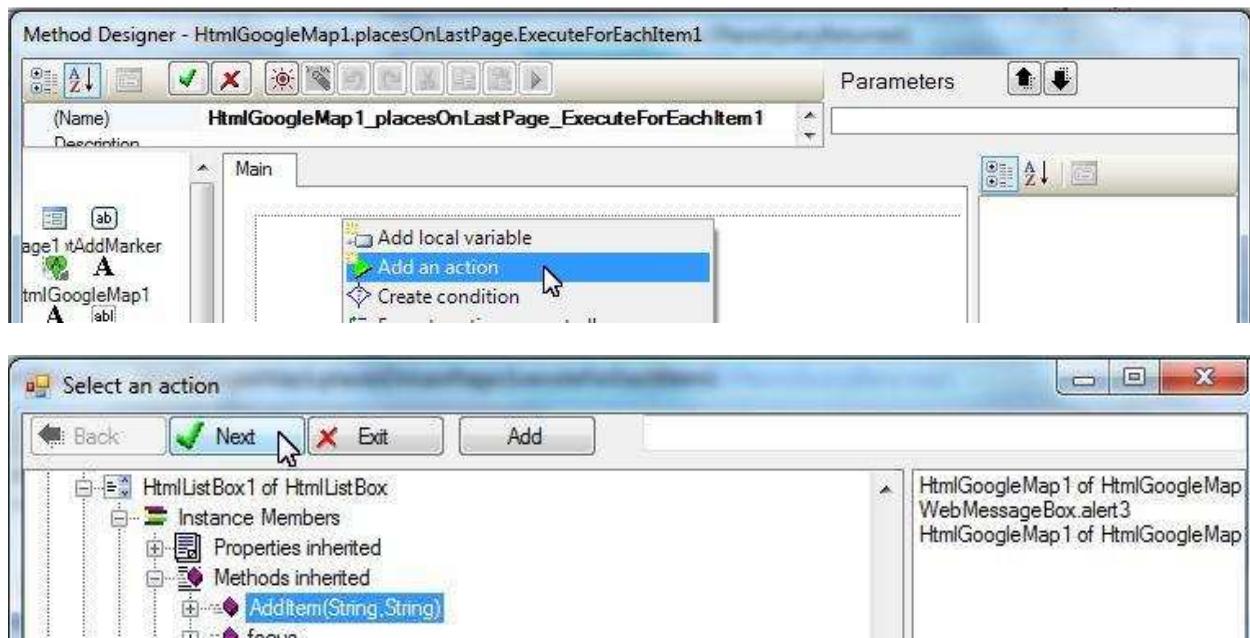
The screenshot shows the following components:

- Object Explorer:** Shows 'HtmlGoogleMap1 of HtmlGoogleMap' with its properties. The 'PlacesCountLastPage' and 'PlacesOnLastPage' properties are highlighted.
- Action Context Menu:** A context menu is open over the 'PlacesOnLastPage' property, with the 'Execute actions for all items' option selected.
- Object Picker:** An 'Object picker' window is open, showing the 'PlacesCountLastPage' and 'PlacesOnLastPage' properties.

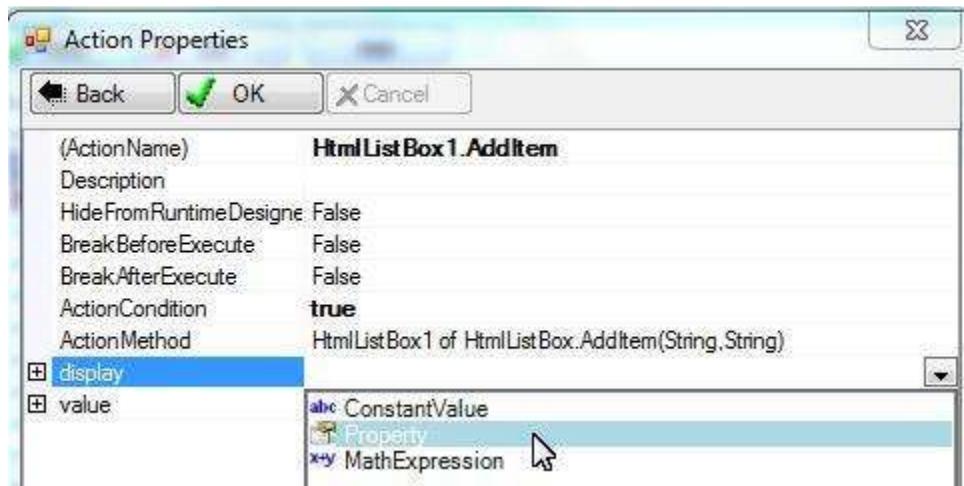
Edit the action to process each place:

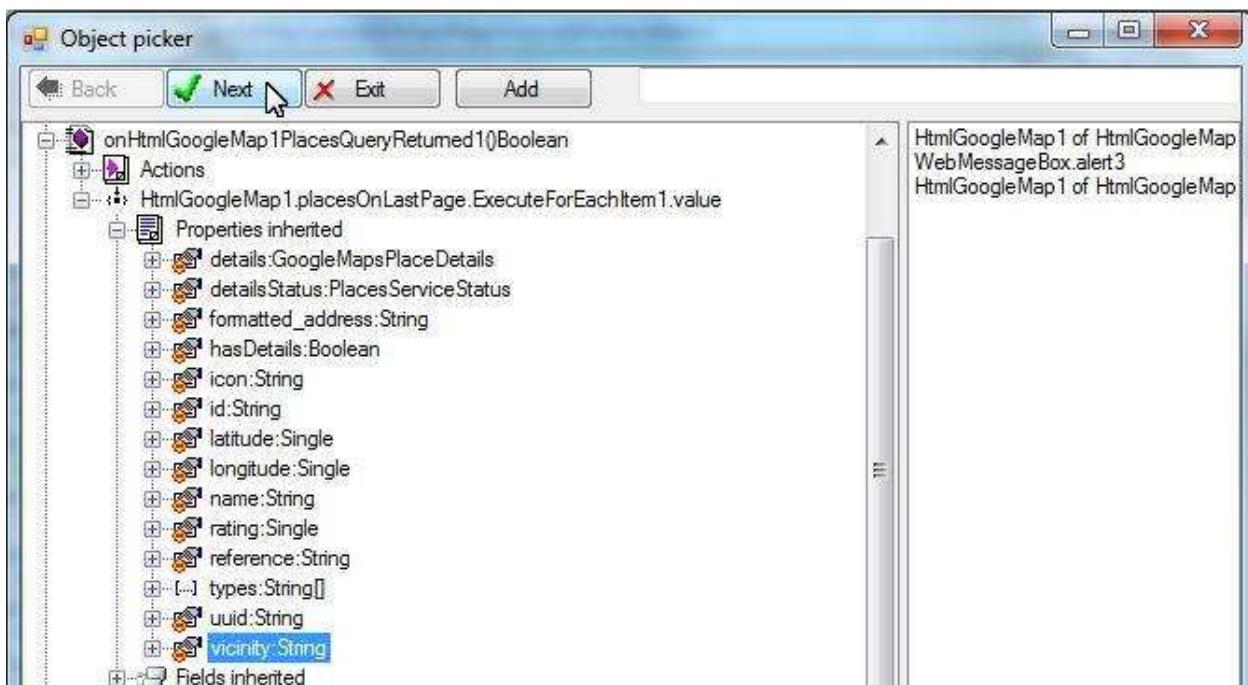


A method editor appears. Create an action to add a list item:

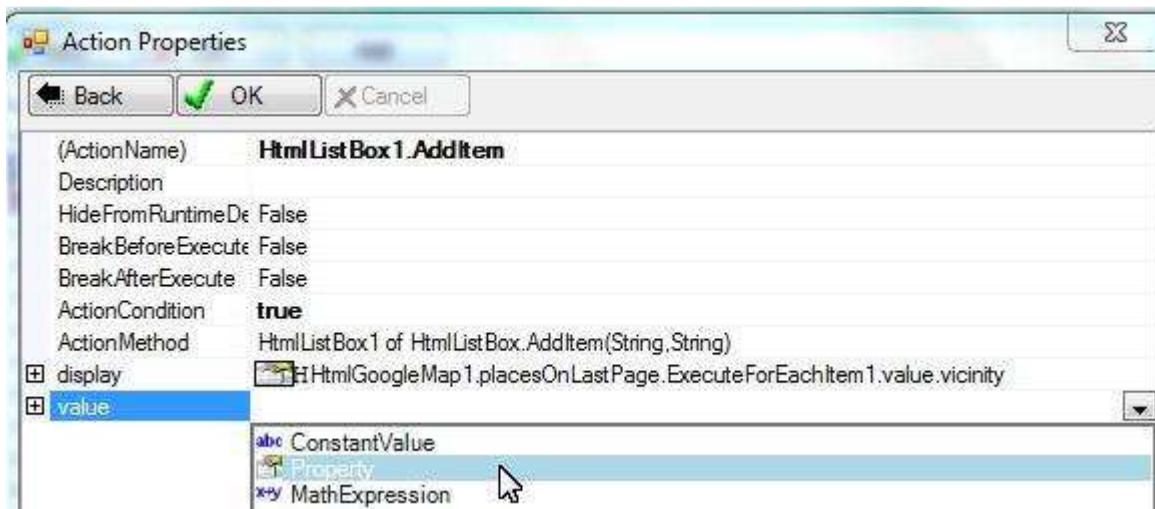


Display vicinity on the list box:





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

```

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
        vicinity:String
  
```

Action Properties

(ActionName) **HtmlListBox1.AddItem**

Description
HideFromRuntimeDe False
BreakBeforeExecute False
BreakAfterExecute False
ActionCondition **true**
ActionMethod **HtmlListBox1 of HtmlListBox.AddItem(String, String)**

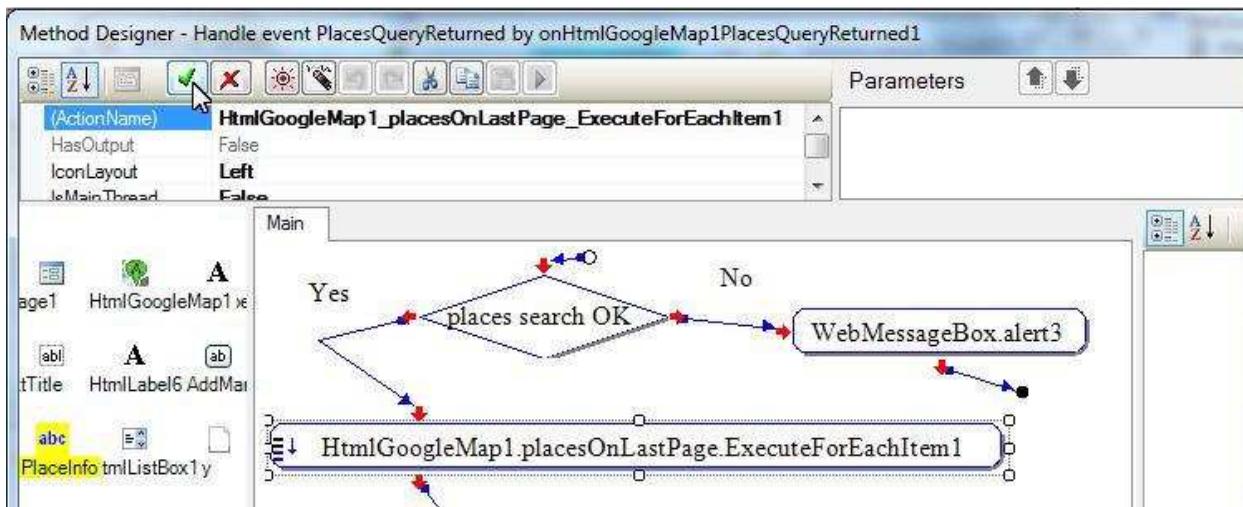
display	HHtmlGoogleMap1.placesOnLastPage.ExecuteForEachItem1.value.vicinity
value	HHtmlGoogleMap1.placesOnLastPage.ExecuteForEachItem1.value.uuid

Method Designer - HtmlGoogleMap1.placesOnLastPage.ExecuteForEachItem1

```

graph TD
    Main --> AddItem
    subgraph Main
        direction TB
        A[age1xtAddMarker]
        B[HtmlGoogleMap1]
        A --- B
    end
    AddItem[HtmlListBox1.AddItem]

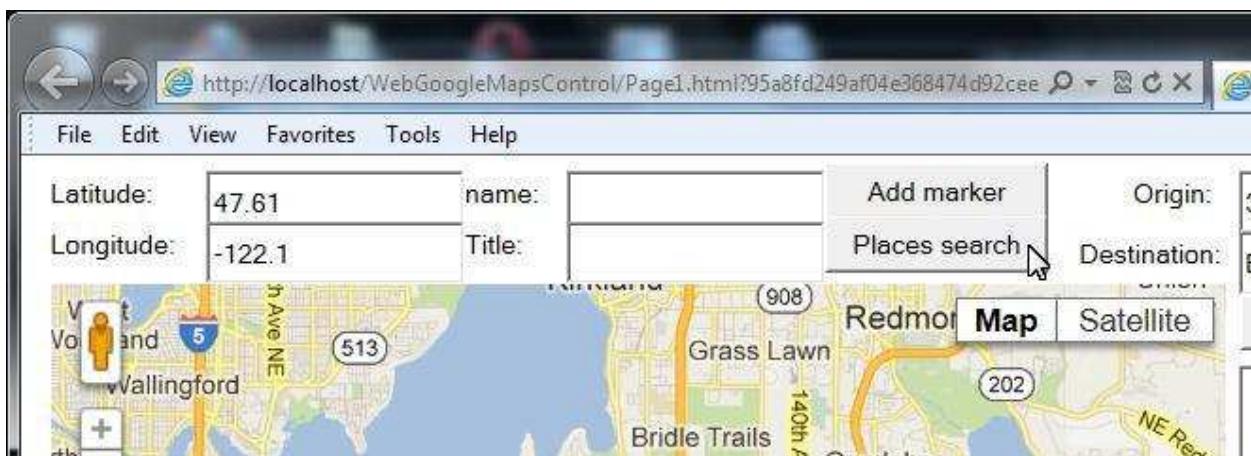
```



The event handler method is created:



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

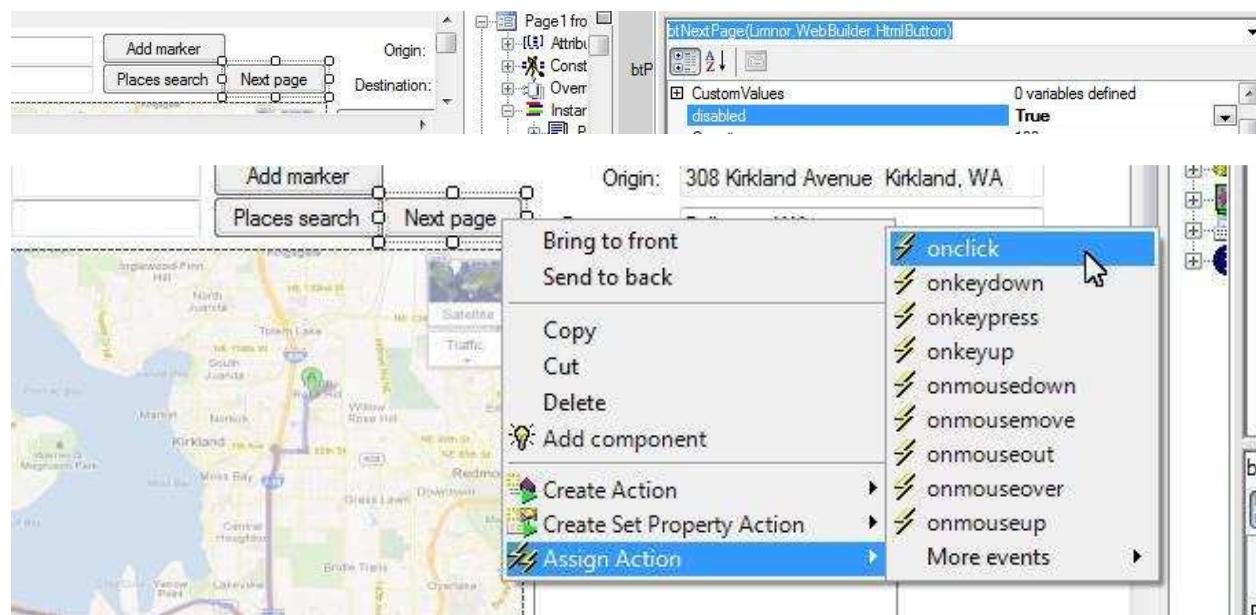


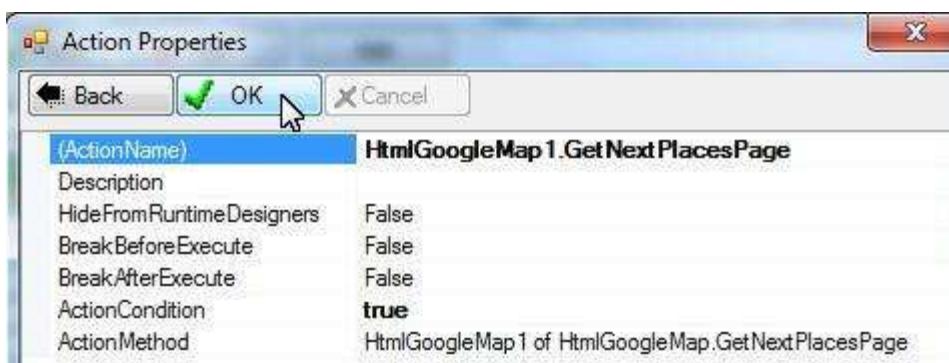
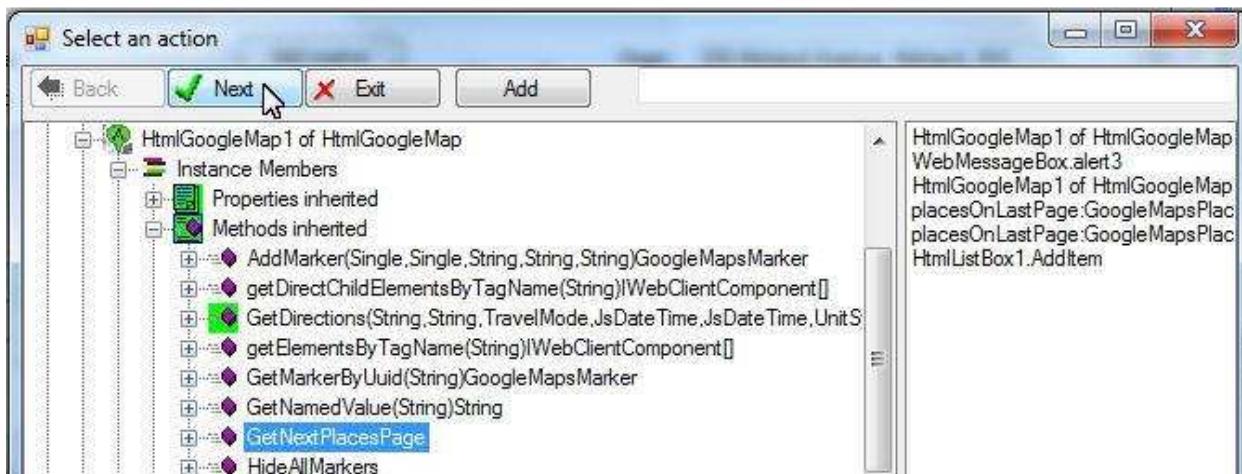
Places appear in the list box:



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`.

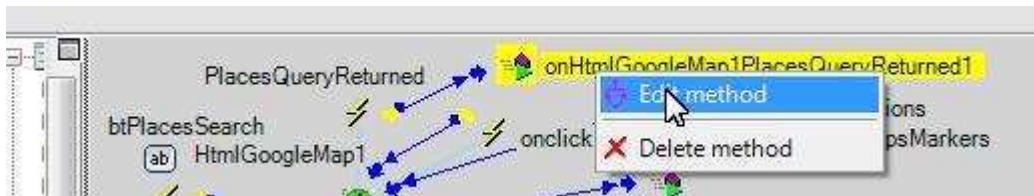


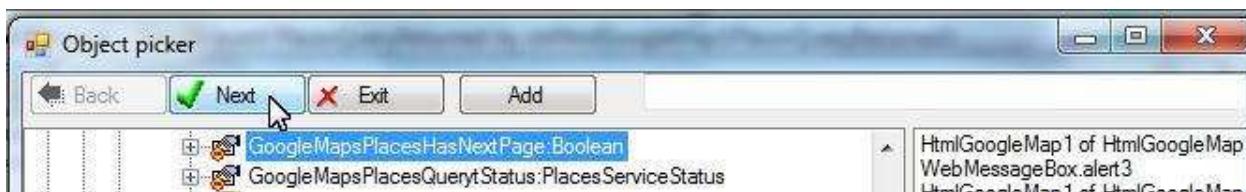
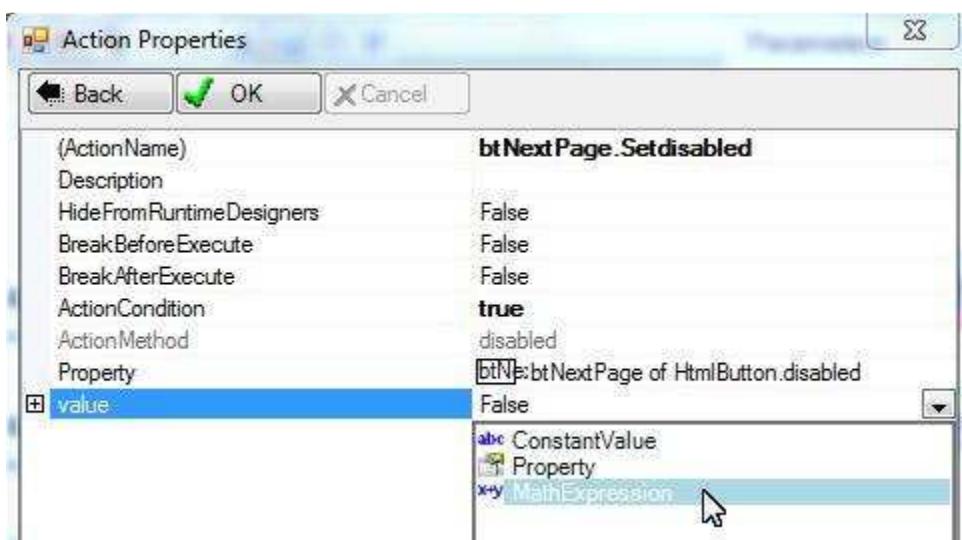
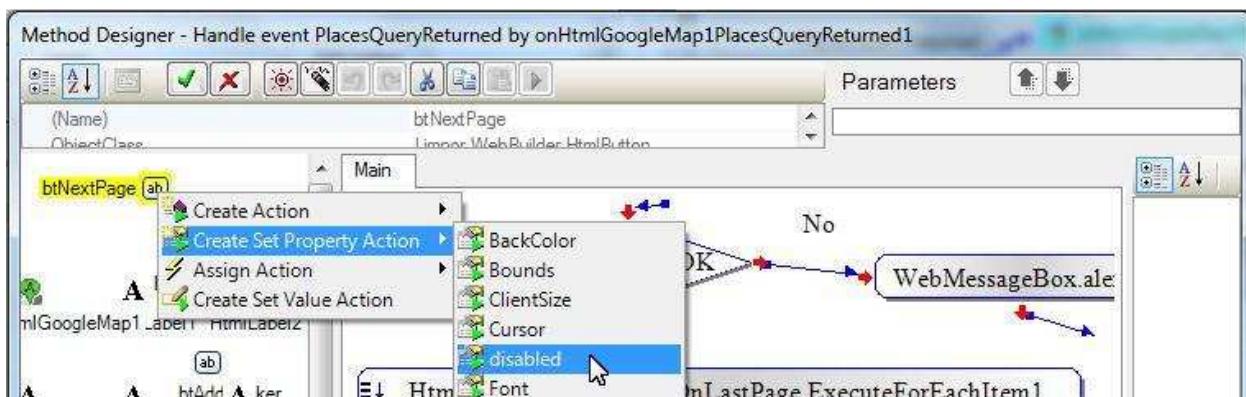


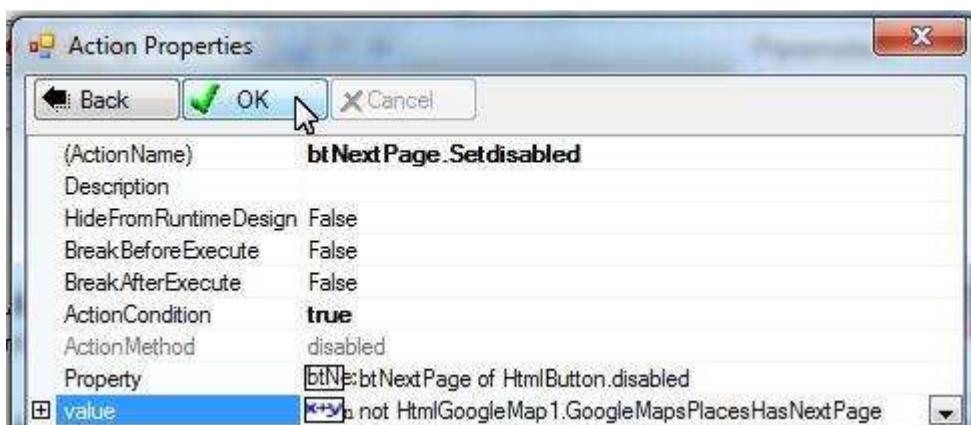
The action is created and assigned to the button:



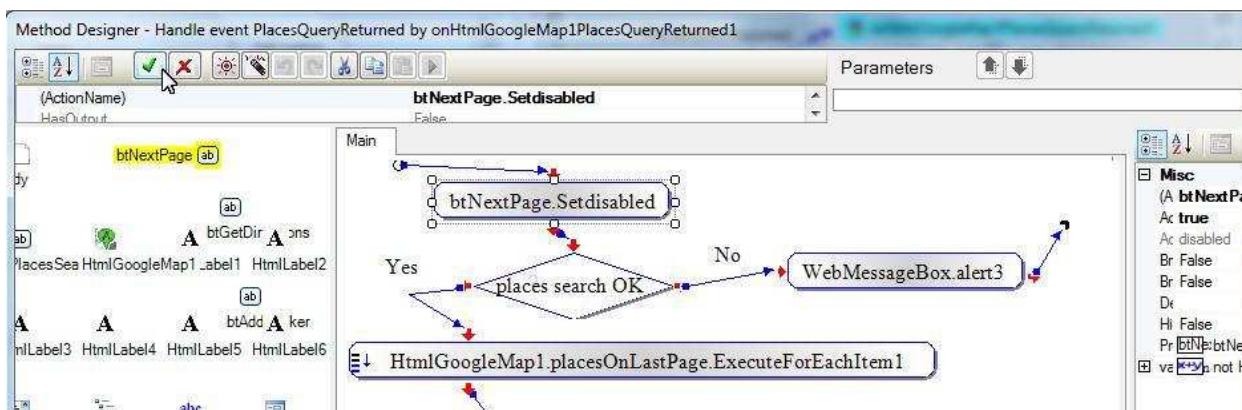
Let's enable or disable the button at event `PlacesQueryReturned`. We may do it on the existing event handler:







Link the new action as the first action:



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

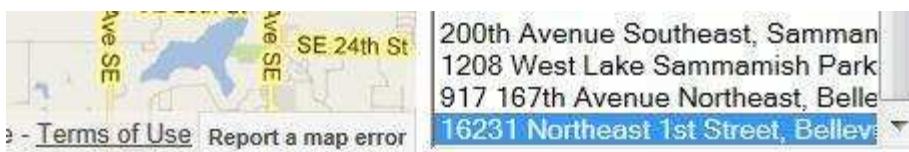
Google Maps Control | 2012



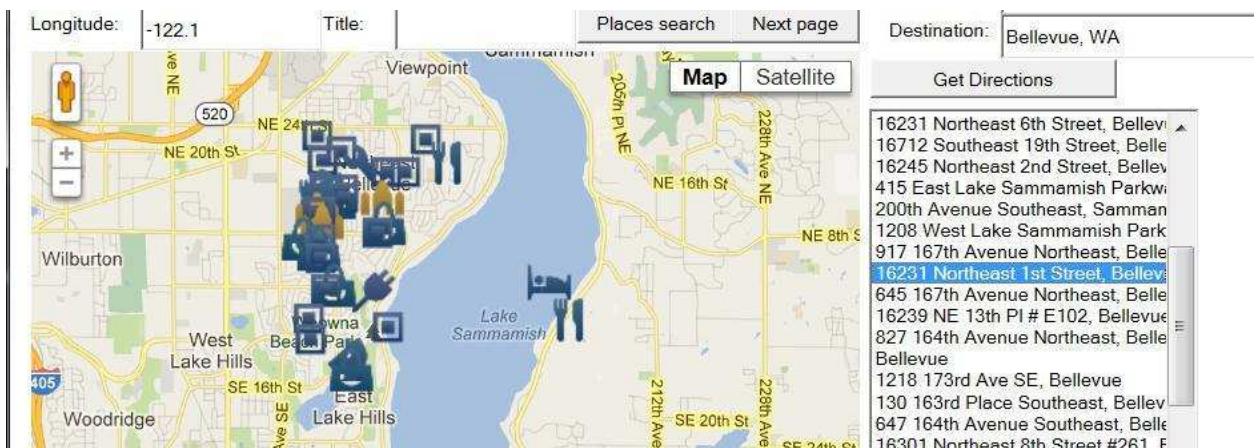
Click "Places search". We can see that "Next page" is enabled:



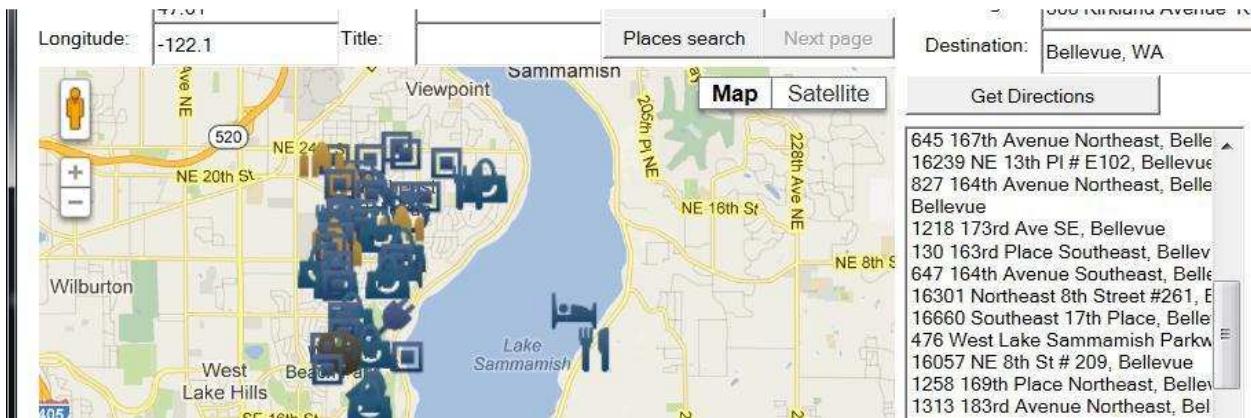
Also note how many items in the list box:



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.

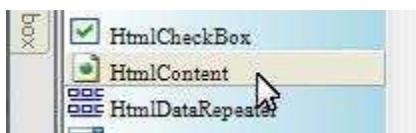
Programming sample

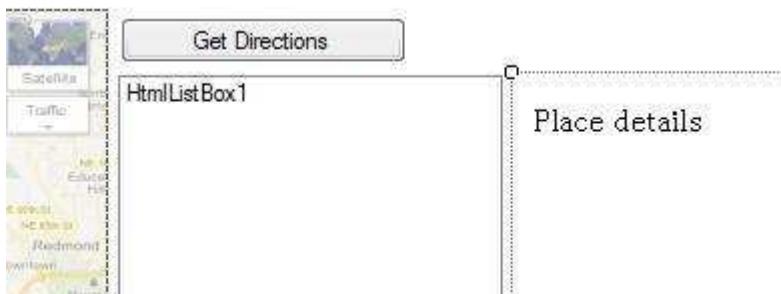
Suppose we want to do such programming:

- ❖ 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.

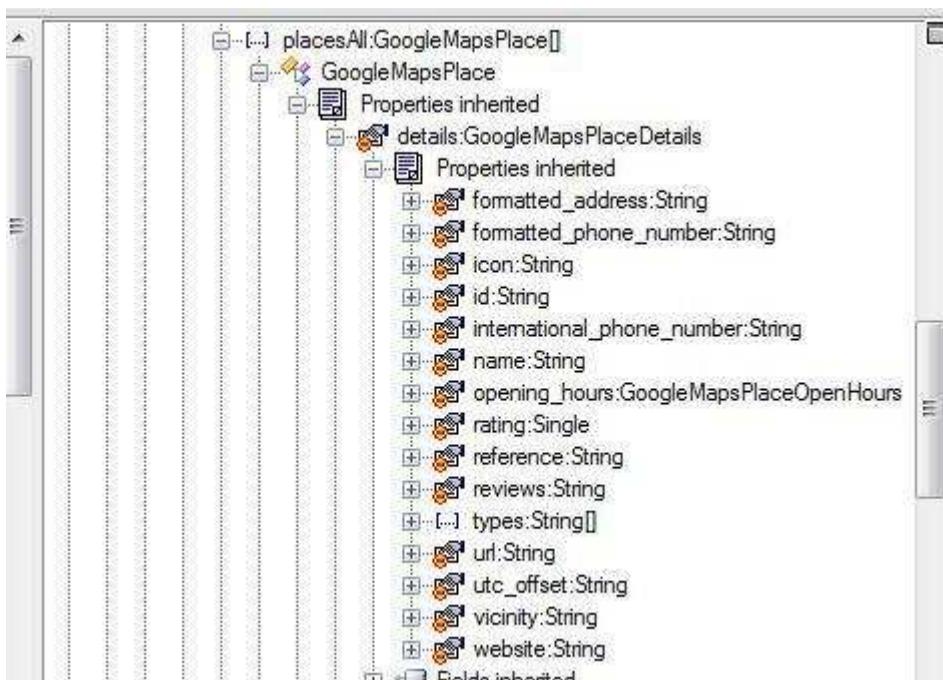
Display details

Add an `HtmlContent` control for displaying place details:



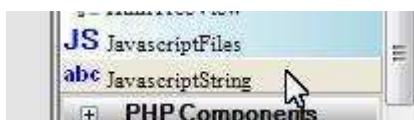


We may examine place details object from Object explorer:



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:

The screenshot shows the Limnor Web Builder interface. At the top, there's a toolbar with icons for file operations like New, Open, Save, and Print. Below the toolbar is a navigation bar with tabs for Design, Misc, and Properties. Under the Design tab, a section labeled '(Name)' contains the text 'jsPlaceDetails'. Under the Misc tab, there's a 'Text' section with a 'value' field containing '[!name!]'. A small '...' button is located next to the value field.

HTML Contents:

```

Place name: [!name!]

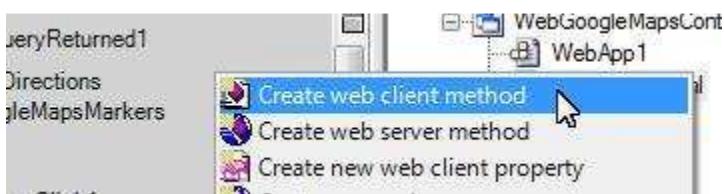
Address: [!formatted_address!]

Phone number: [!formatted_phone_number!]

Web site: [!website!]

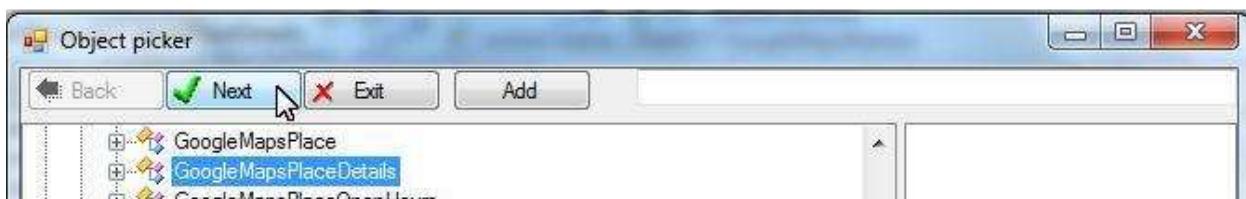
```

Create a method to display place details:

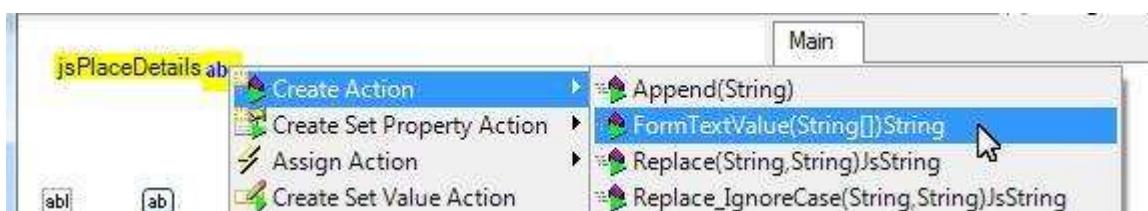


Add a parameter for place details:

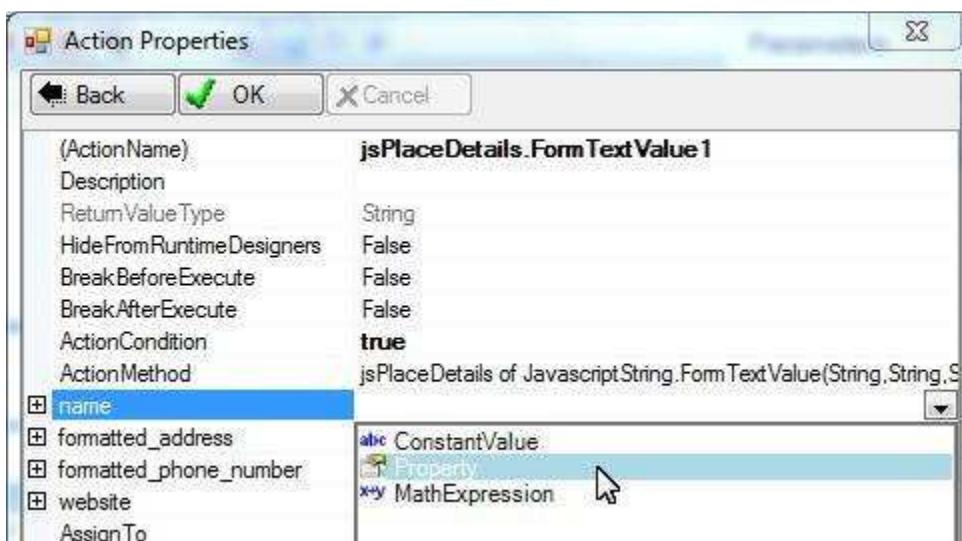
The screenshot shows the Limnor Web Builder Method Designer window for a method named 'showPlaceDetails'. The 'Parameters' section has a 'Add parameter' button highlighted with a blue selection bar. Below the Method Designer is the 'Object picker' window, which displays a tree structure of available components. The 'Namespaces' node is expanded, and the 'Limnor.WebBuilder' namespace is selected. Under 'Limnor.WebBuilder', the 'DialogHtmlContents' and 'DialogJsVariables' components are visible.

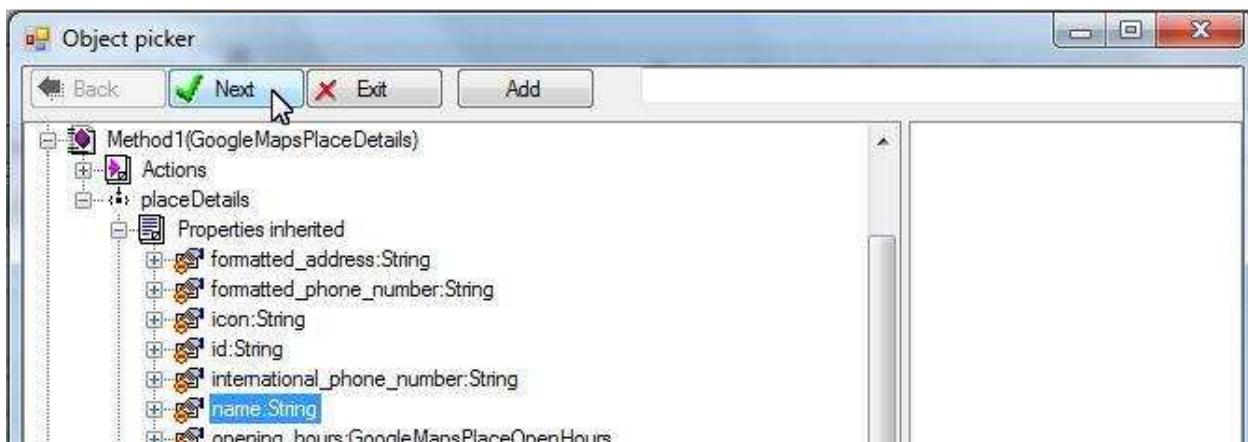


Create an action to form HTML for place details:

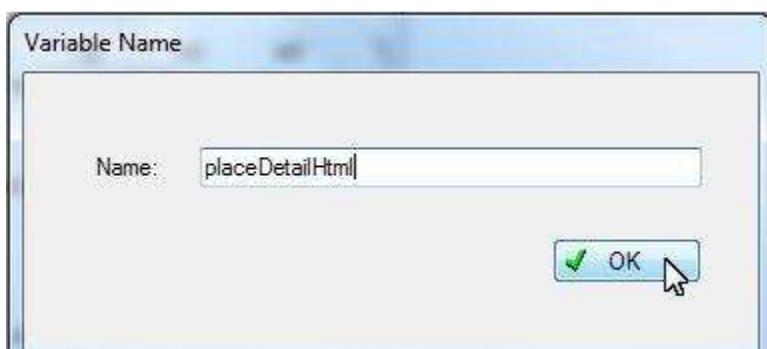
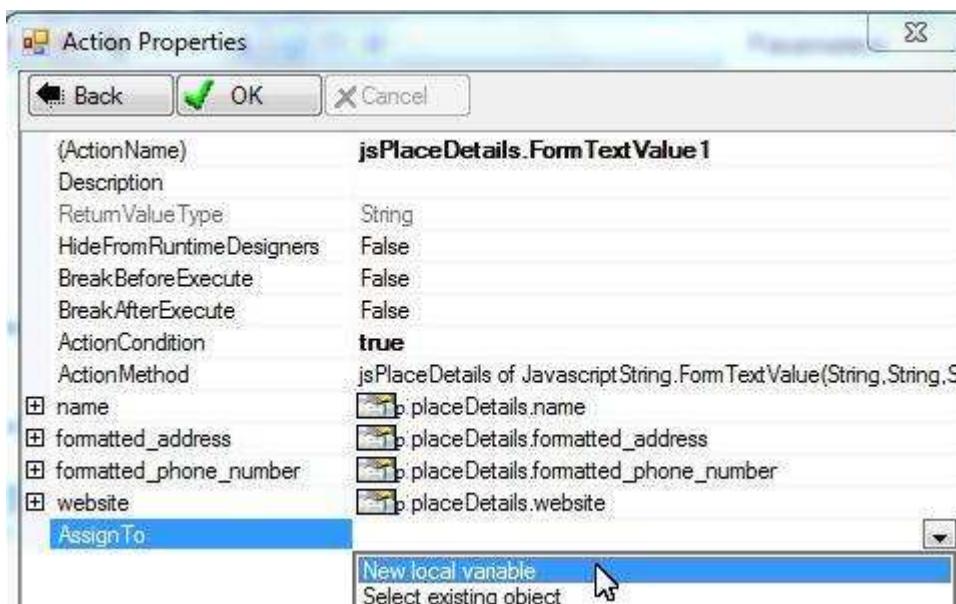


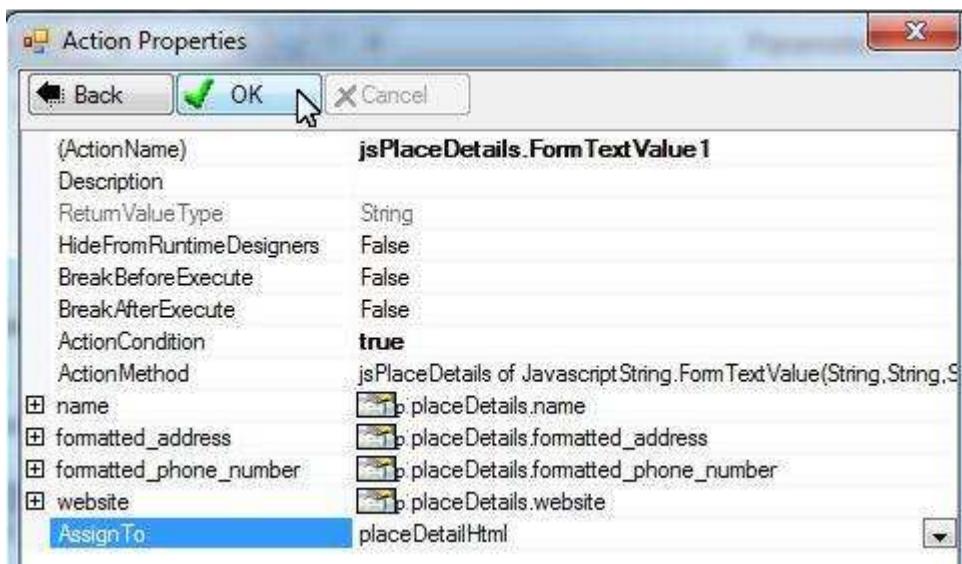
Pass place details to the action:



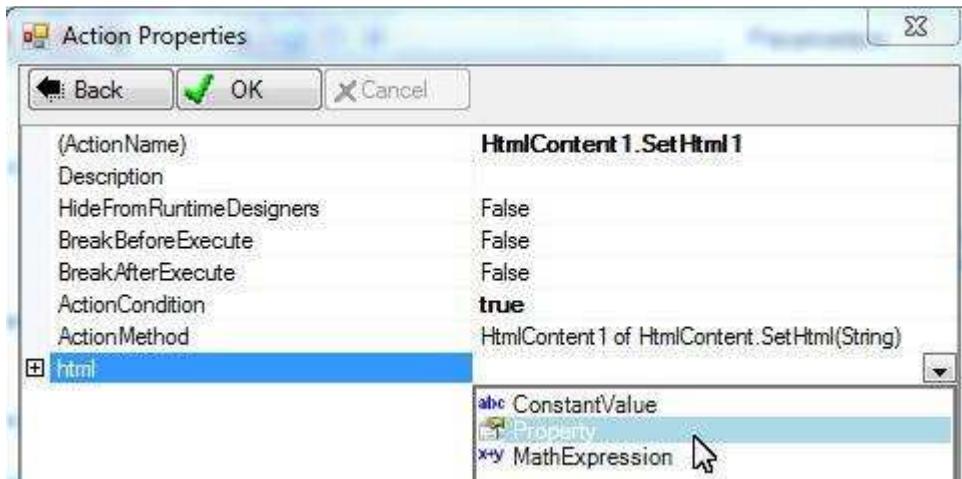
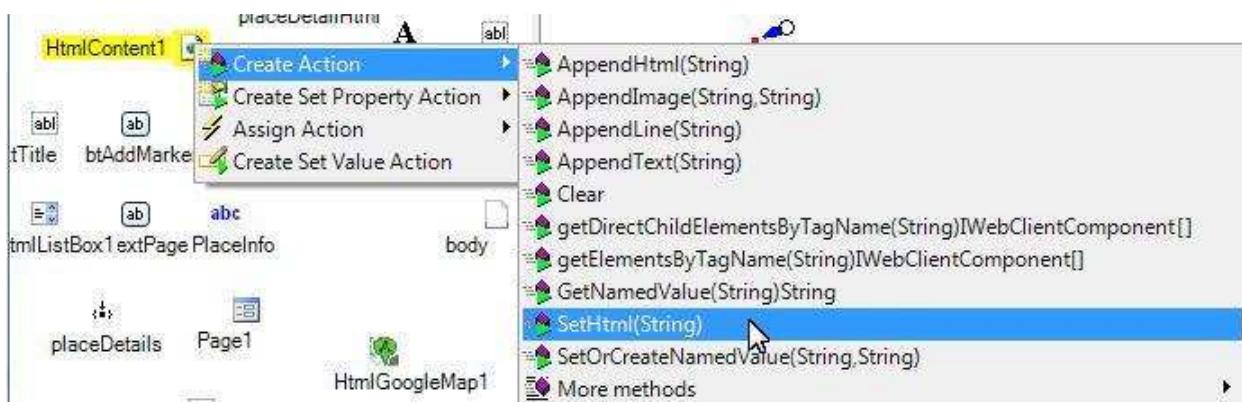


Use a variable for the result:





Use the html on the HTML box:

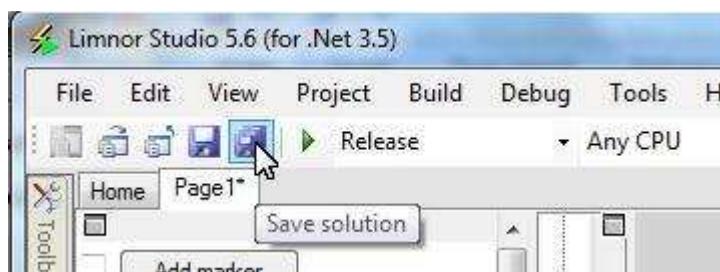


The screenshot shows three windows from the Limnor Studio interface:

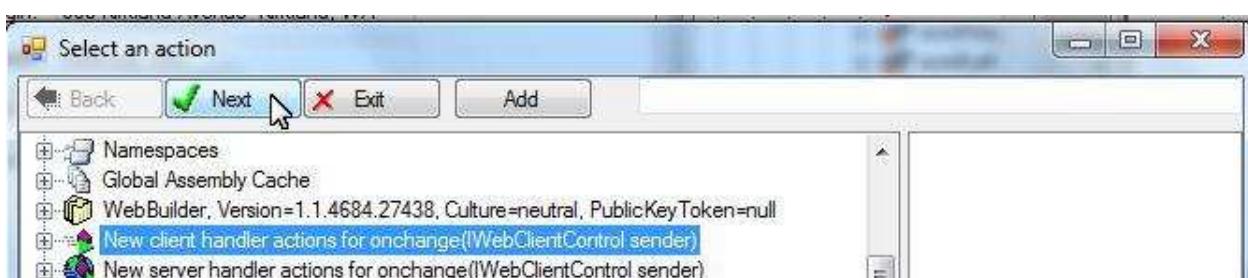
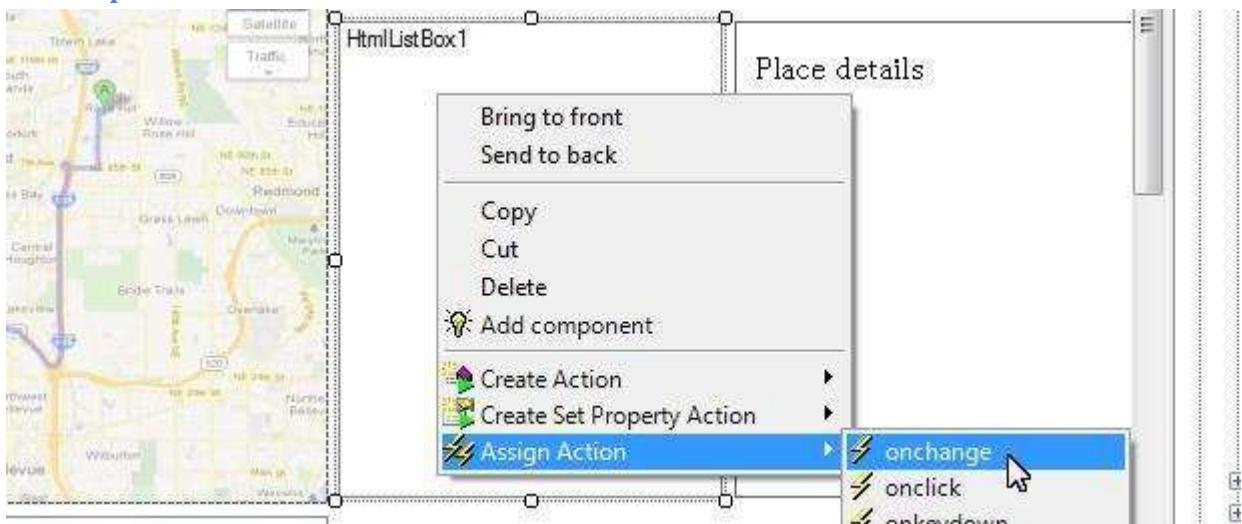
- Object picker:** Shows a tree view of actions under "Method1(GoogleMapsPlaceDetails)". The "placeDetailHtml" action is selected.
- Action Properties:** A dialog for "HtmlContent1.SetHtml1". It shows the following properties:

(ActionName)	HtmlContent1.SetHtml1
Description	
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	HtmlContent1 of HtmlContent.SetHtml(String)
html	placeDetailHtml
- Method Designer - showPlaceDetails:** A flowchart showing the logic for setting the place detail HTML. It starts with "jsPlaceDetails abc" (with a condition "placeDetailHtml A"), followed by "HtmlContent1" (with a condition "A"). This leads to "placeDetailHtml=jsPlaceDetails.FormTextValue1" (with a condition "A"). Finally, it leads to "HtmlContent1.SetHtml1" (with a condition "A").

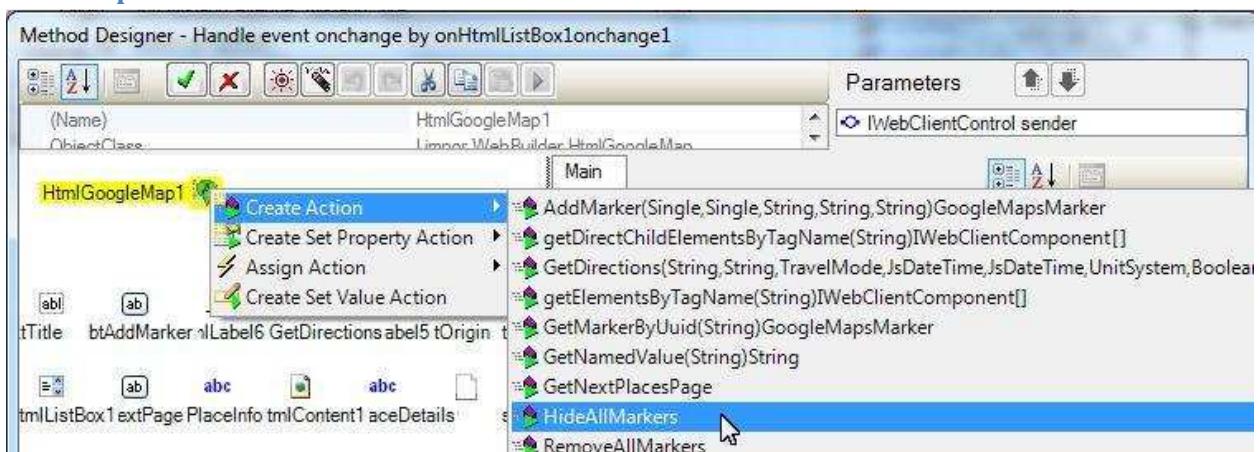
Save the project so that the method is available for programming:



Handle place item selection



Hide all places

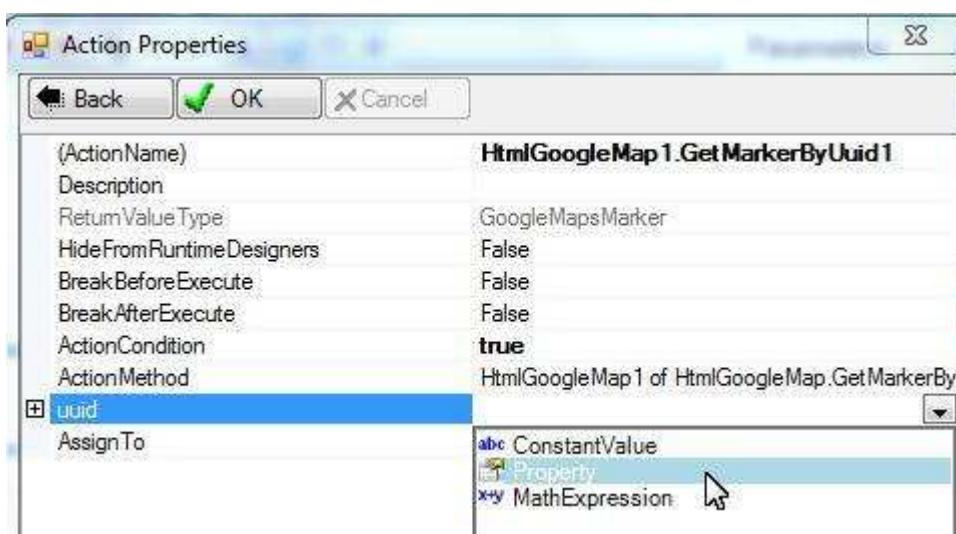




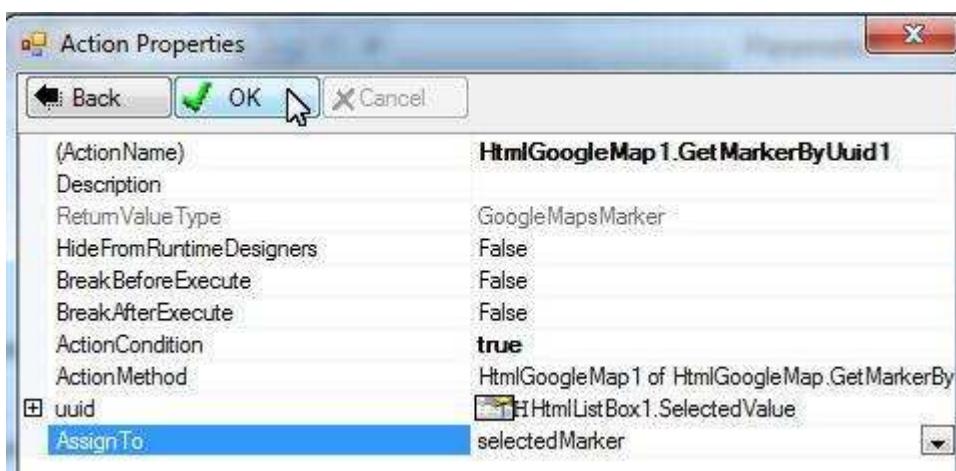
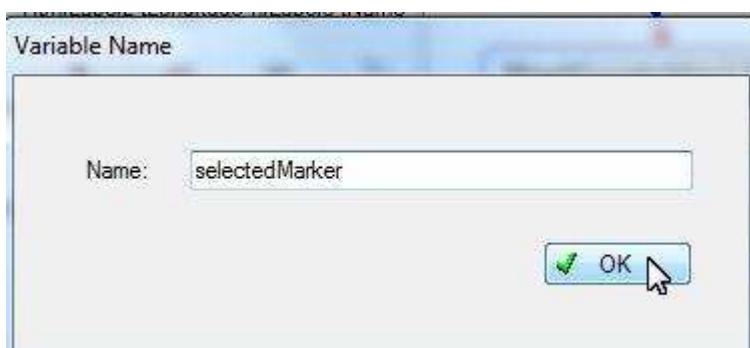
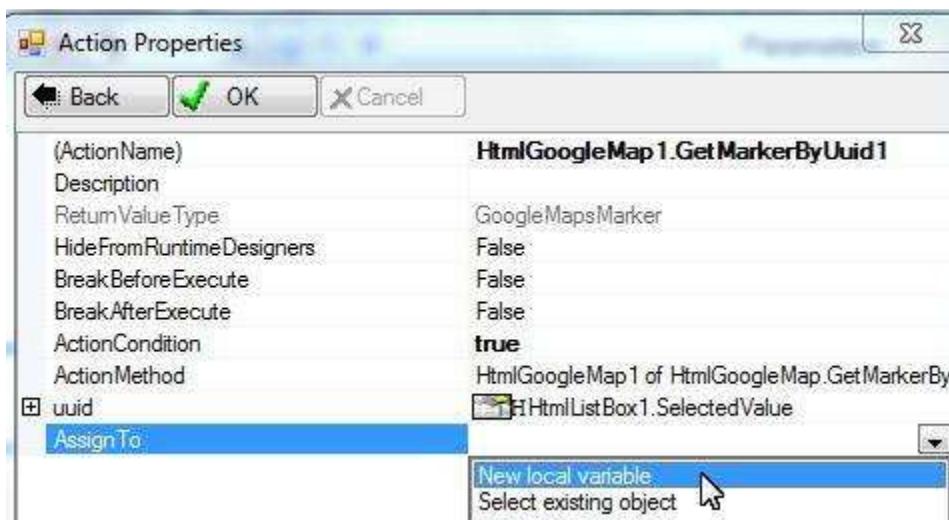
Find place marker

Diagram illustrating the creation of a 'GetMarkerByUuid' action:

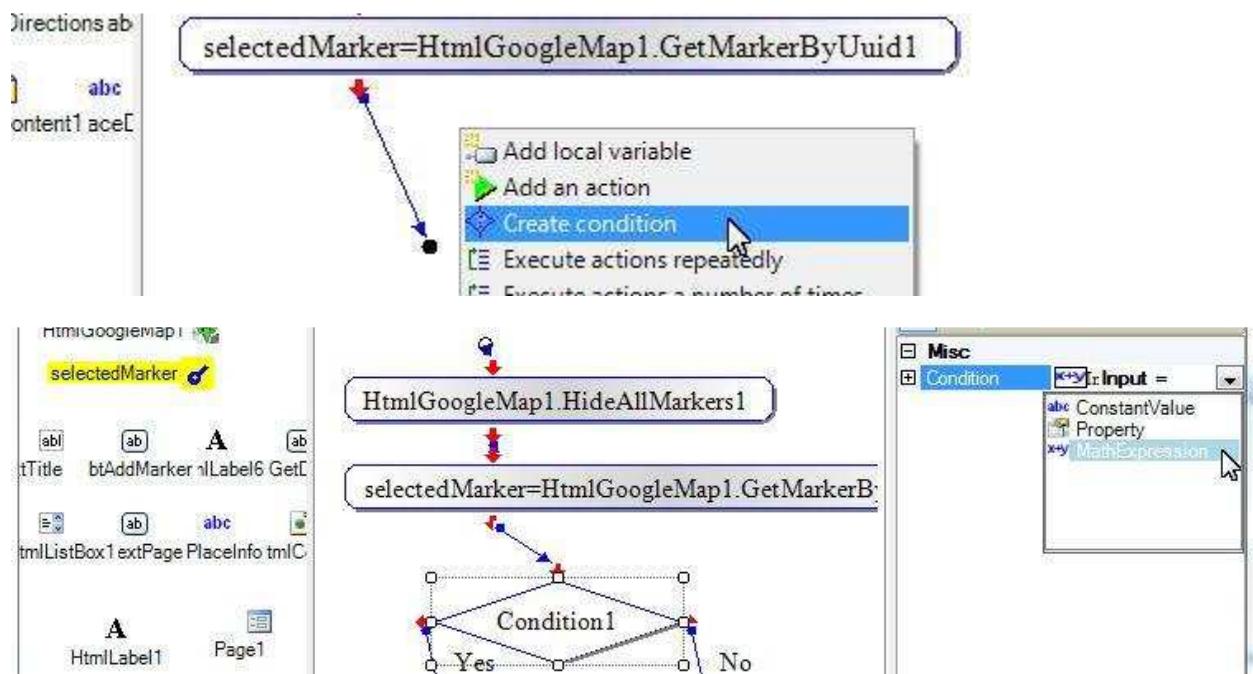
- The 'Main' tab is selected.
- The 'HtmlGoogleMap1' component is selected.
- A context menu is open, showing options like 'Create Action', 'Create Set Property Action', 'Assign Action', and 'Create Set Value Action'. The 'Create Action' option is highlighted.
- The 'Create Action' submenu lists several methods:
 - AddMarker(Single, Single, String, String, String) GoogleMapsMarker
 - getDirectChildElementsByTagName(String) IWebClientComponent
 - GetDirections(String, String, TravelMode, JsDateTime, JsDateTime, Uuid) GoogleMapsPlaceDetails
 - getElementsByTagName(String) IWebClientComponent[]
 - GetMarkerByUuid(String) GoogleMapsMarker** (highlighted)
 - GetNamedValue(String) String
 - GetNextPlacesPage



Use a variable for found marker:



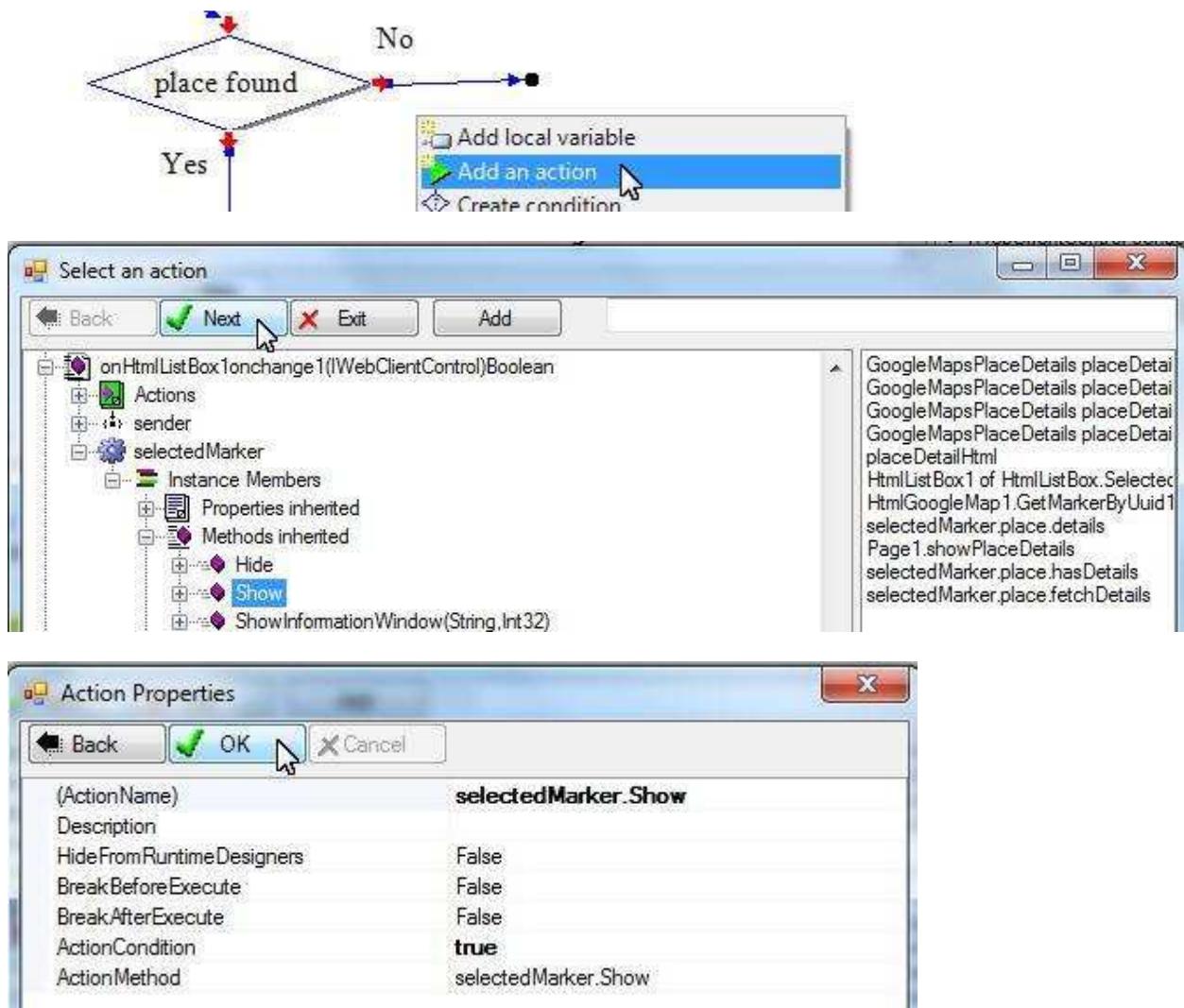
Add a Condition action to check whether a place marker is found:



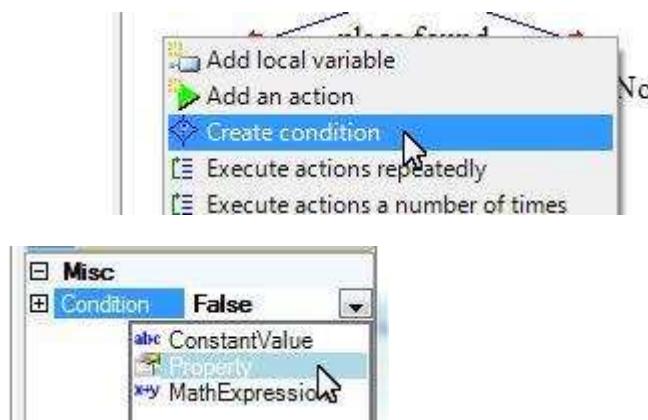
If the result is not null then we found a marker:

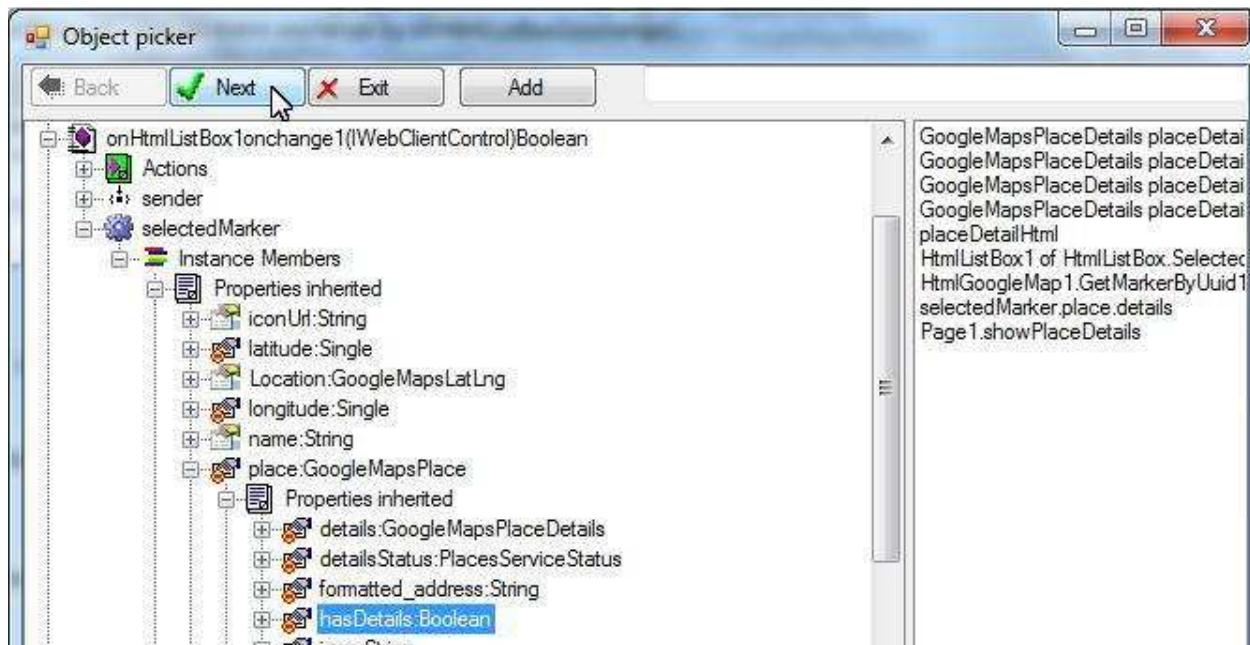
The screenshot shows two instances of the 'Math Expression Editor' window. The top window has the title 'Logic' selected. The input field contains 'Input ≠ null'. The status bar says 'Logic 'not equal' operator'. The bottom window also has the 'Logic' tab selected. The input field contains 'Input ≠ null'. A tooltip 'Finish the editing.' is visible above the toolbar. Both windows show a toolbar with various operators like Val, null, var, F, T, not, and, or, >, ≥, ≠, etc., and a 'DataType' dropdown set to '(Boolean)'.

Show the marker:



Check if place details are available:





Execute showPlaceDetails if place details are available:

The screenshot shows the 'Action Properties' dialog for the 'Page1' object. In the 'Create Action' dropdown, 'showPlaceDetails(GoogleMapsPlaceDetails)' is selected. The 'ActionName' field is set to 'Page1.showPlaceDetails'. The 'ActionCondition' field is set to 'true'. The 'ActionMethod' field is set to 'showPlaceDetails' with the value 'GoogleMapsPlaceDetails()'. The 'placeDetails' field is expanded, showing options for 'ConstantValue', 'Property', and 'MathExpression'.

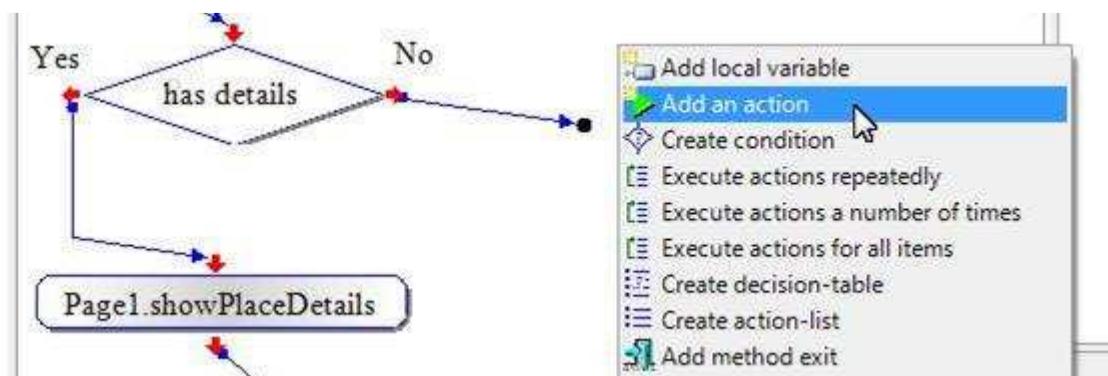
(ActionName)	Page1.showPlaceDetails
Description	
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	showPlaceDetails GoogleMapsPlaceDetails()
placeDetails	<ul style="list-style-type: none"> <input type="radio"/> ConstantValue <input checked="" type="radio"/> Property <input type="radio"/> MathExpression

The screenshot shows the Google Maps Control interface with three main windows:

- Object Browser:** Shows the event `onHtmlListBox1onchange1(IWebClientControl)Boolean` expanded. It lists `Actions`, `sender`, and `selectedMarker`. Under `selectedMarker`, there are `Instance Members` and `Properties inherited`. A tooltip on the right lists methods like `placeDetailHtml`, `HtmlListBox1`, and `GetMarkerByUuid1`.
- Object Picker:** A modal window titled "Object picker" with buttons for Back, Next, Exit, and Add. It shows the path `place:GoogleMapsPlace` expanded to `Properties inherited`, `details:GoogleMapsPlaceDetails`, and `detailsStatus:PlacesServiceStatus`. A tooltip on the right lists `placeDetailHtml` and other details.
- Action Properties:** A dialog titled "Action Properties" for the action `Page1.showPlaceDetails`. It shows the following properties:

(ActionName)	<code>Page1.showPlaceDetails</code>
Description	
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	<code>true</code>
ActionMethod	<code>showPlaceDetails</code>
placeDetails	<code>s:selectedMarker.place.details</code>

Execute `fetchDetails` if place details are not available:



Select an action

Back Next Exit Add

onHtmlListBox1 onchange1(IWebClientControl)Boolean

- Actions
- sender
- selectedMarker**
- Instance Members
 - Properties inherited
 - iconUrl:String
 - latitude:Single
 - Location:GoogleMapsLatLng
 - longitude:Single
 - name:String
 - place:GoogleMapsPlace
 - Fields inherited
 - Events inherited
 - Methods inherited
 - fetchDetails**
- title:String

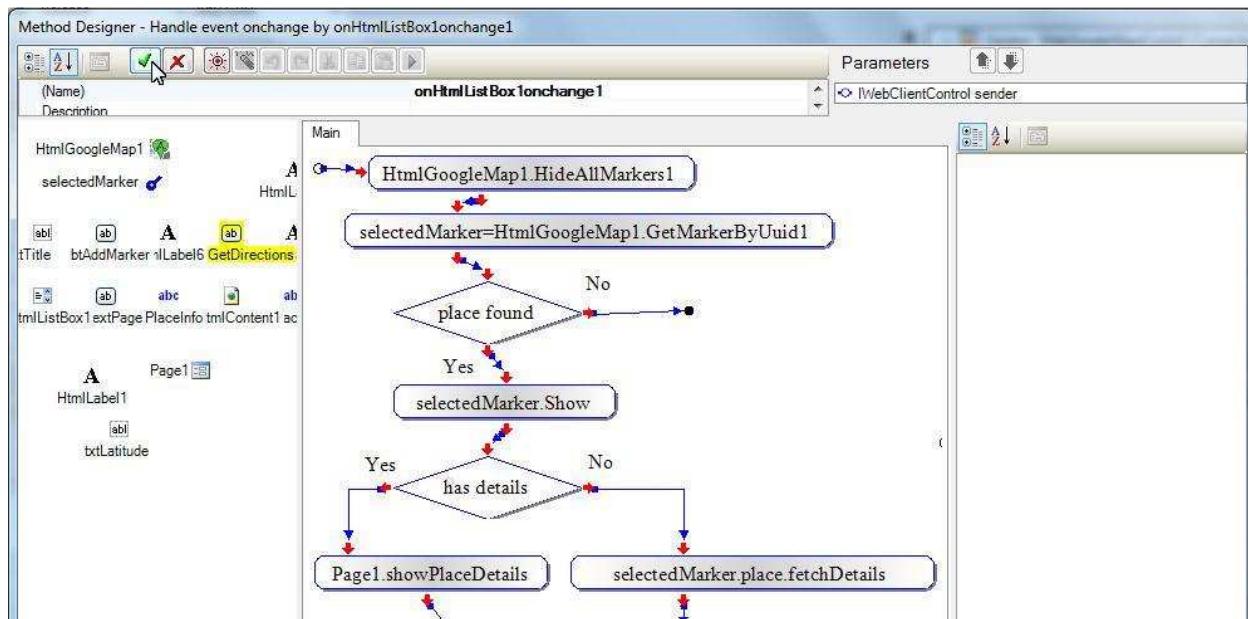
Google Maps Place Details placeDetail
 Google Maps Place Details placeDetail
 GoogleMapsPlaceDetails placeDetail
 GoogleMapsPlaceDetails placeDetail
 placeDetailHtml
 HtmlListBox1 of HtmlList Box. Selected
 HtmlGoogleMap1.GetMarkerByUuid1
 selectedMarker.place.details
 Page1.showPlaceDetails
 selectedMarker.place.hasDetails

Action Properties

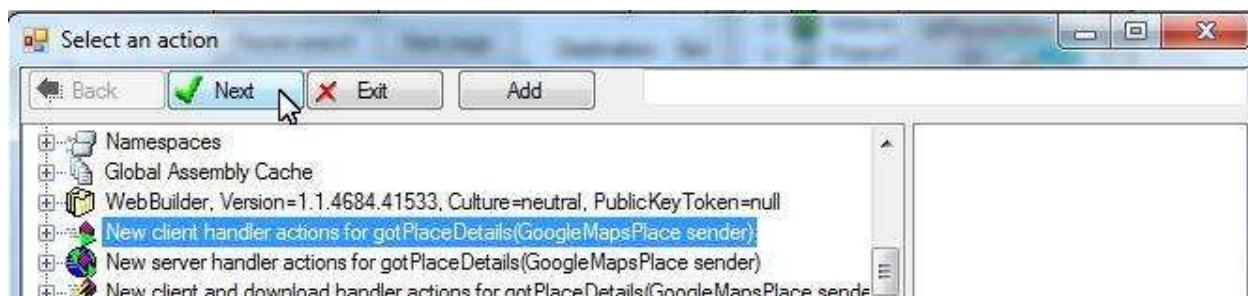
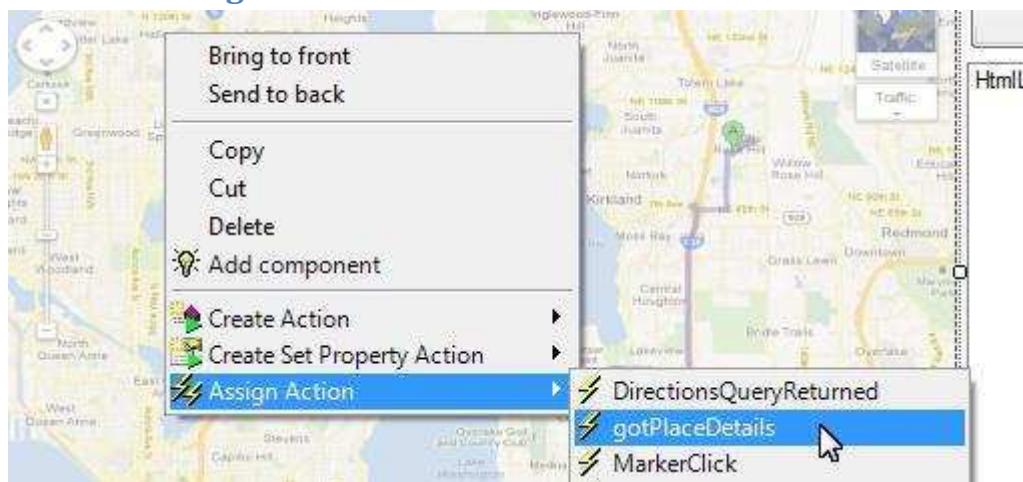
Back OK Cancel

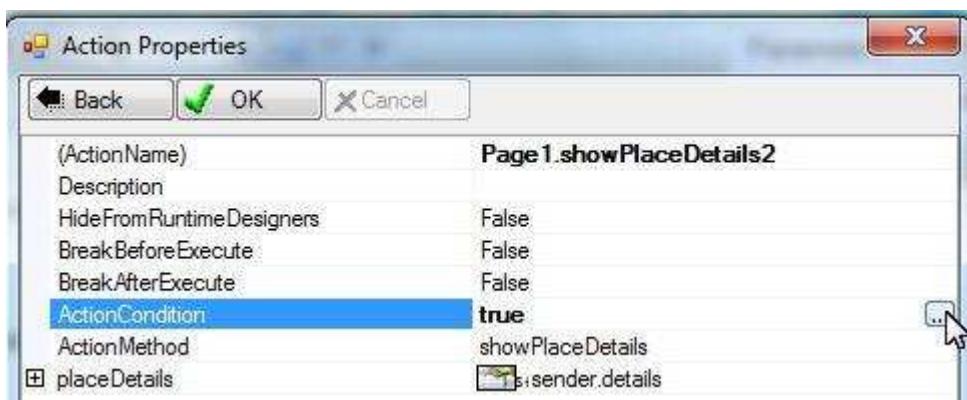
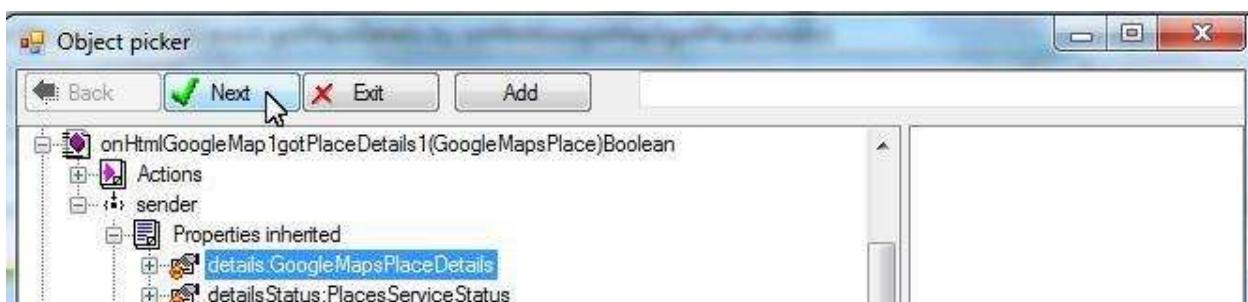
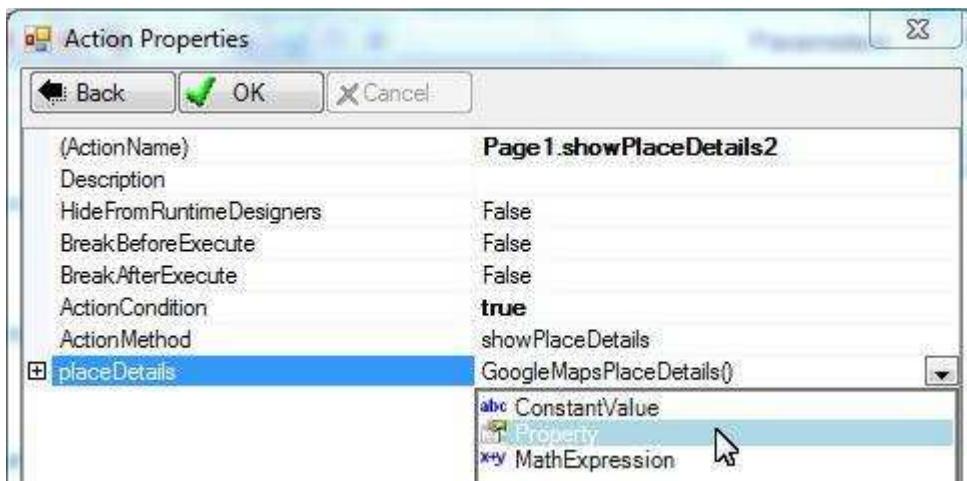
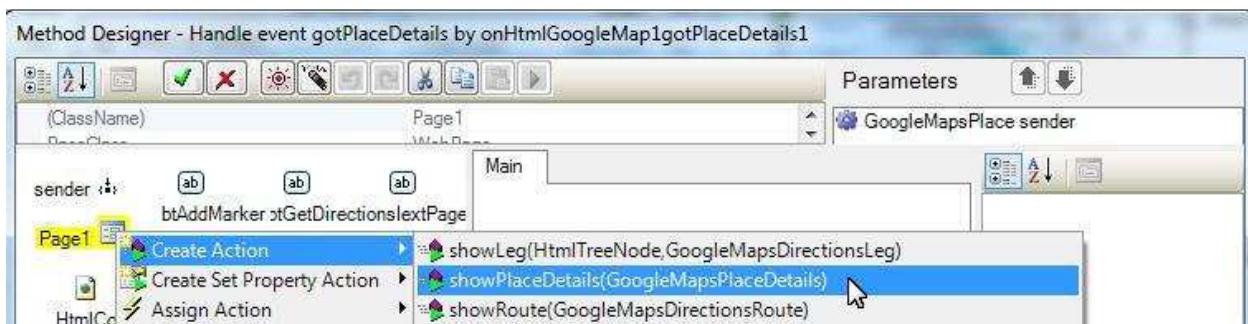
(ActionName)	selectedMarker.place.fetchDetails
Description	
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	true
ActionMethod	place:GoogleMapsPlace.fetchDetails

We are done handling the place selection:



Handle event gotPlaceDetails





Math Expression Editor

0 A property

Object picker

onHtmlGoogleMap1gotPlaceDetails1(GoogleMapsPlace)Boolean

- Actions
- sender
 - Properties inherited
 - details:GoogleMapsPlaceDetails
 - detailsStatus:PlacesServiceStatus
 - formatted_address:String
 - hasDetails:Boolean
 - icon:String

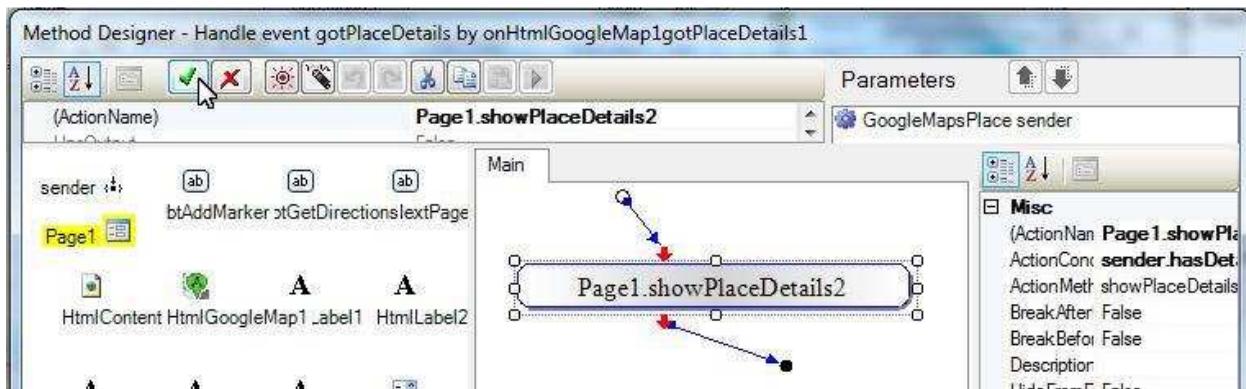
Math Expression Editor

Finish the editing.

sender.hasDetails

Action Properties

(ActionName)	Page1.showPlaceDetails2
Description	
HideFromRuntimeDesigners	False
BreakBeforeExecute	False
BreakAfterExecute	False
ActionCondition	sender.hasDetails
ActionMethod	showPlaceDetails
placeDetails	1:s:sender.details



Test

Launch the web page. Click "Places search":

Latitude: 47.61 Longitude: -122.1 name:

Add marker Places search Next page

Origin: 308 Kirkland Avenue, Kirkland
Destination: Bellevue, WA

Get Directions

17900 NE 16th Street, Bellevue
16020 Southeast 16th Street, Bellevue
16330 Northeast 4th Street, Bellevue
17222 Northeast 8th Street, Bellevue
827 164th Avenue Northeast, Bellevue
16661 Northup Way, Bellevue
16256 Northeast 8th Street, Bellevue
16231 Northeast 6th Street, Bellevue
16242 Northup Way, Bellevue
16231 Northeast 6th Street, Bellevue
1209 176th Avenue Northeast, Bellevue
16231 Northeast 6th Street, Bellevue
16712 Southeast 19th Street, Bellevue
16245 Northeast 2nd Street, Bellevue

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

Place name: Chevron

Address: 16256 Northeast 8th Street, Bellevue

Phone number: (425) 643-2278

Web site: <http://www.chevron.com>

Select another item:



Feedback

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