Telerik Corp.

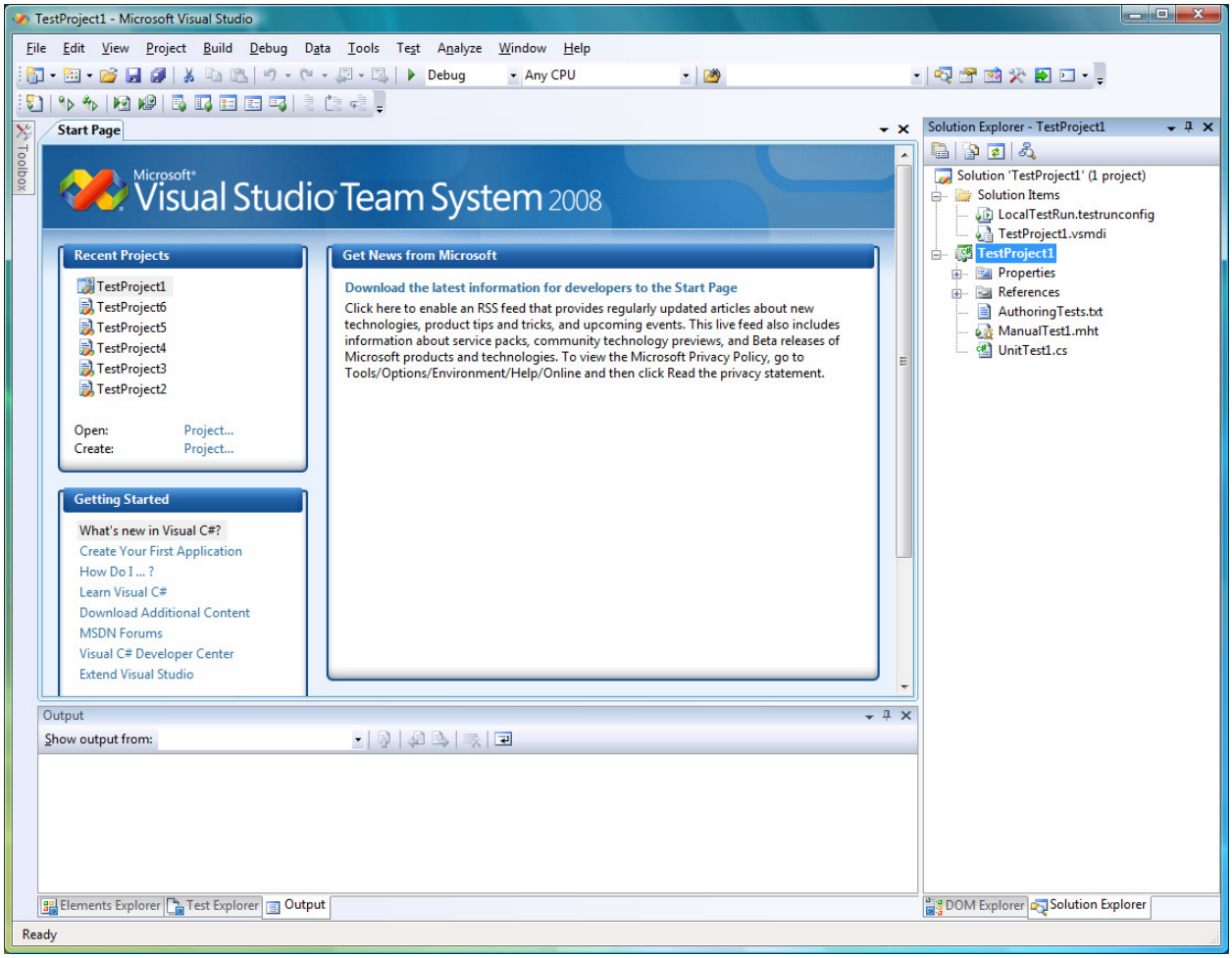
WebUI Test Studio 2010.1 Developer Edition Quick-Start Guide

Contents

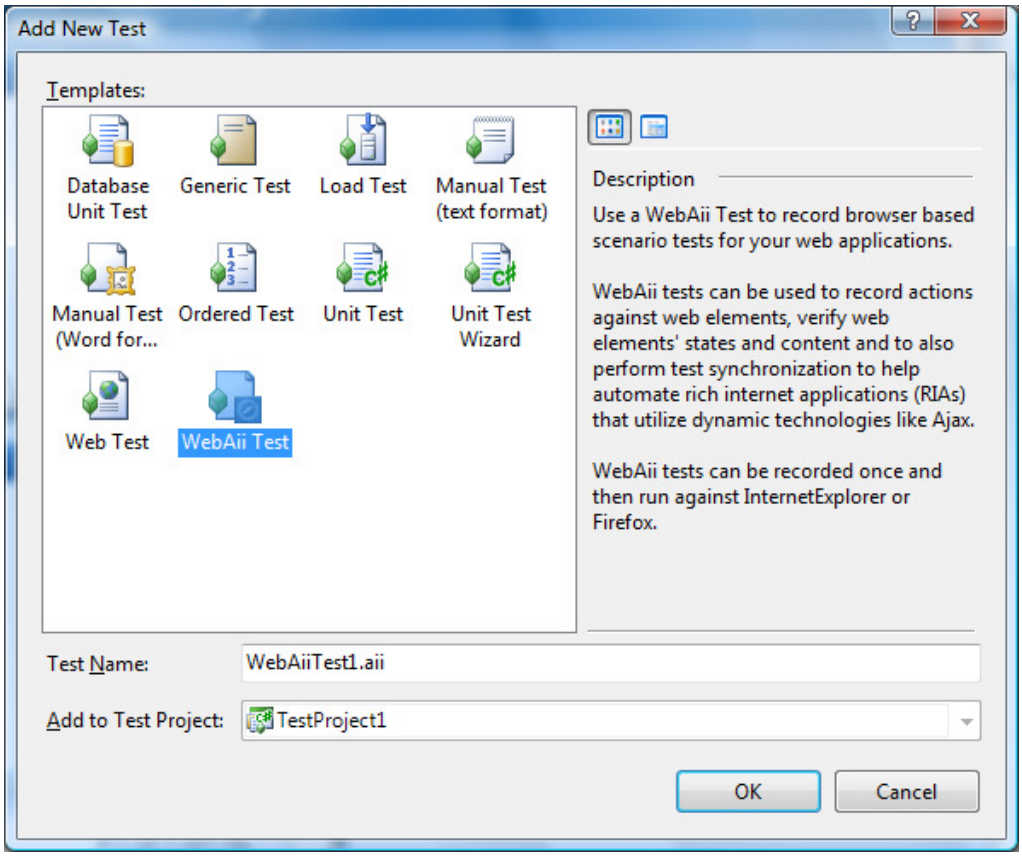
Creating and Running Your First Test

Let's jump right in and show you how easy it is to record and run your tests.

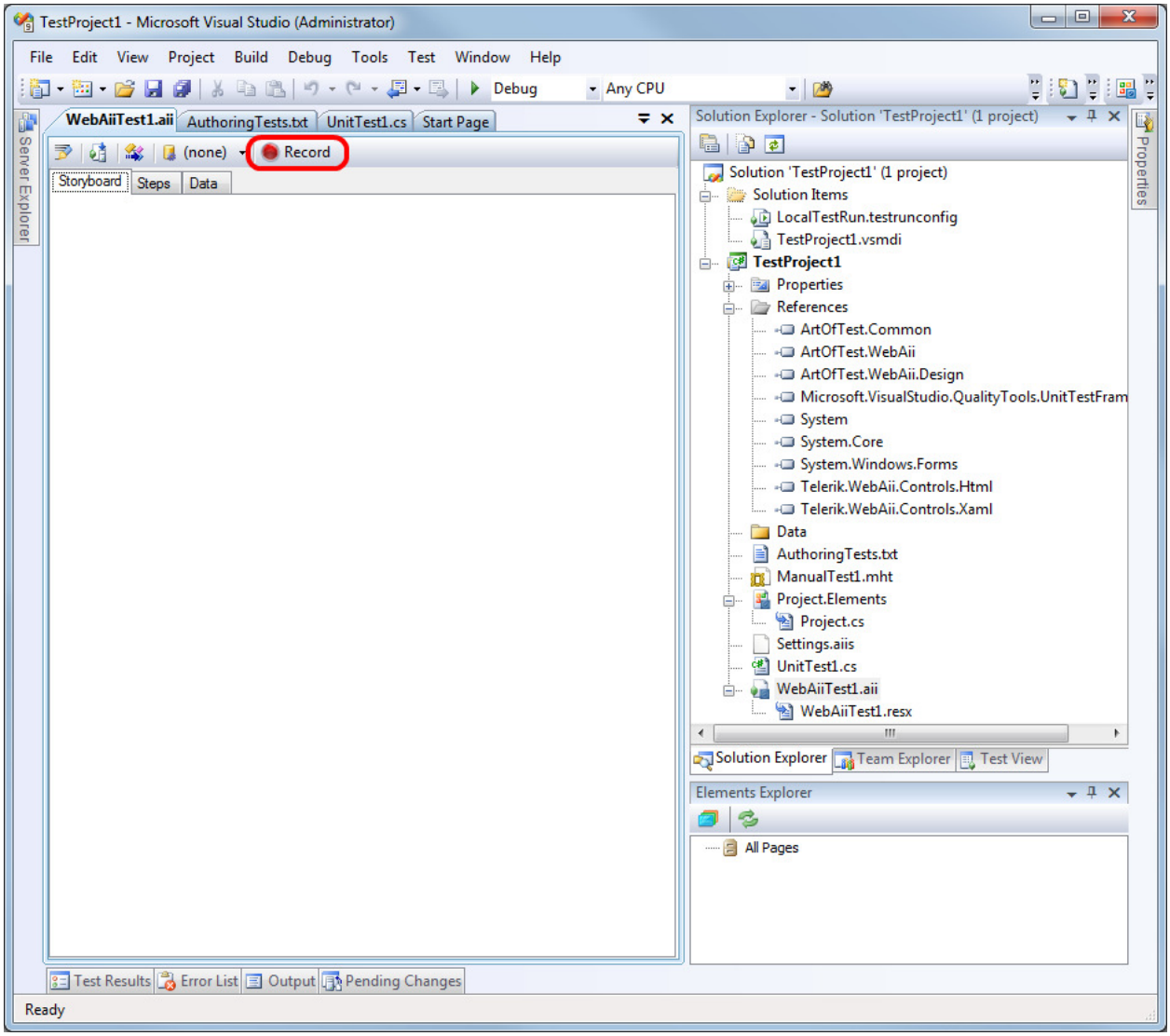
1) Open Visual Studio and create a new test project.



2) Right click on the project node in the solution explorer and select “Add -> New Test…”.

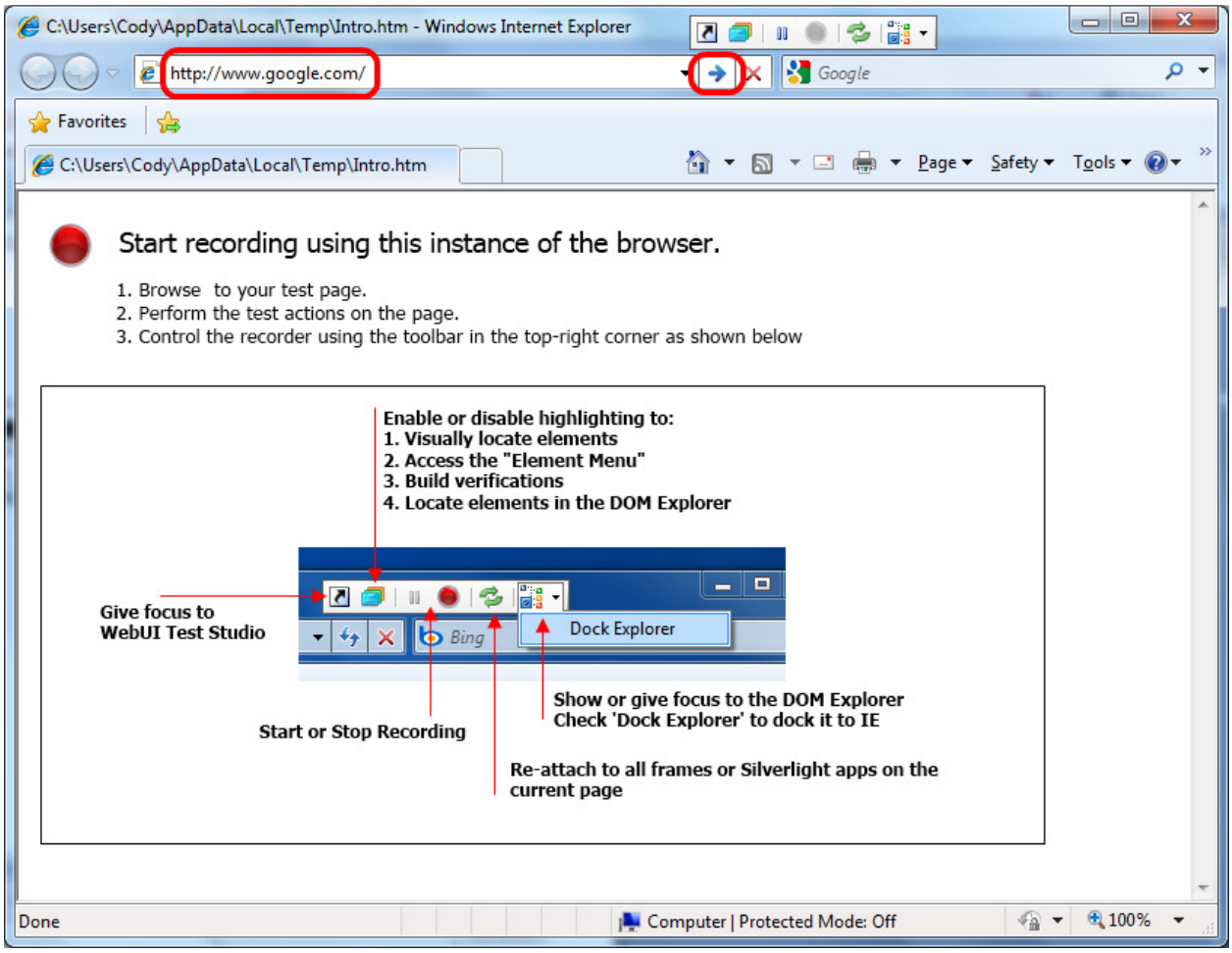


3) Click on the “WebAii Test” type and select OK.
4) Your new empty test should open automatically.



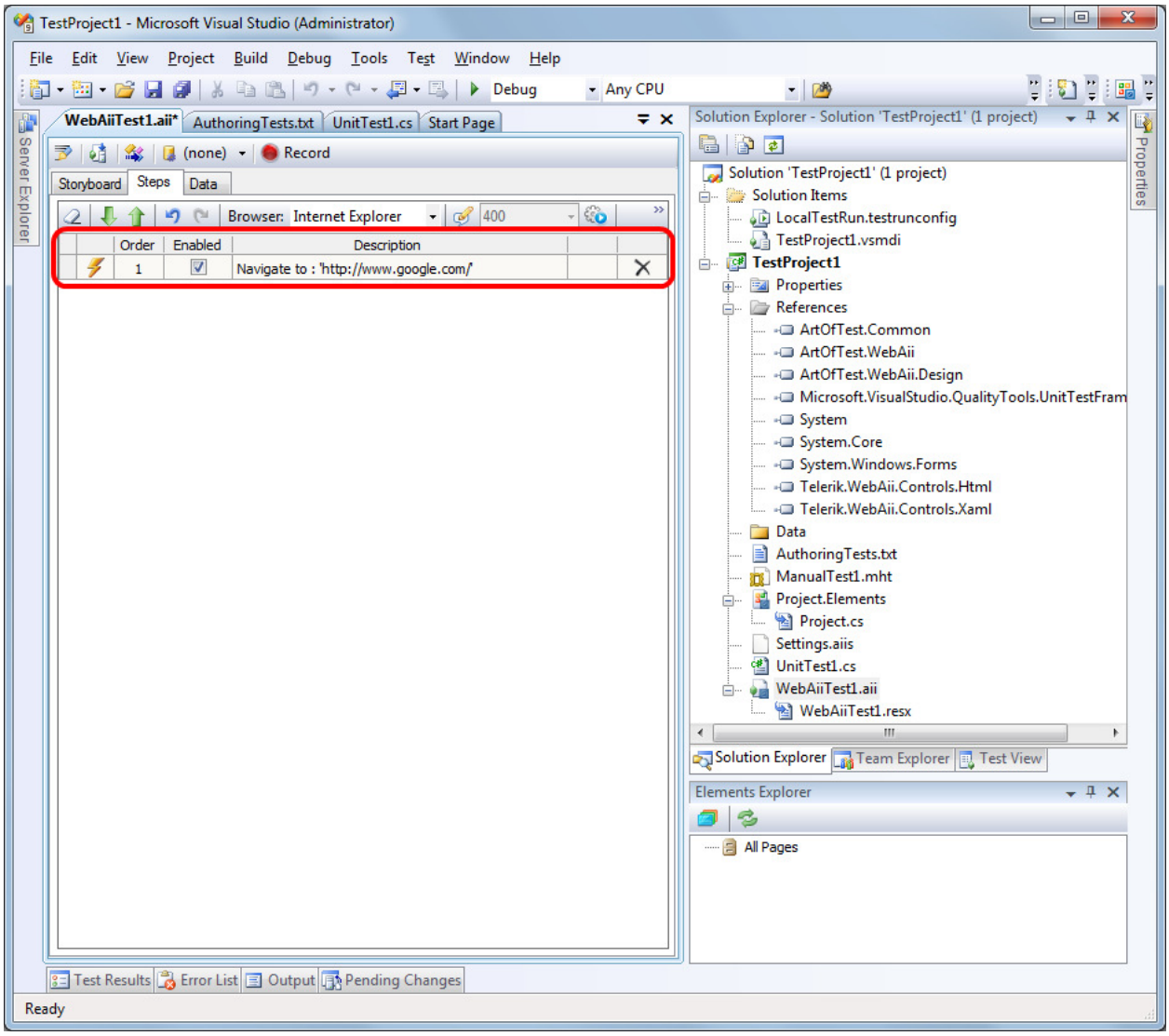
5) To begin recording the steps of your test, click on the “Record” button (circled in red above).

6) Internet Explorer will automatically open in record mode.



7) In the “Url” box, enter www.google.com.

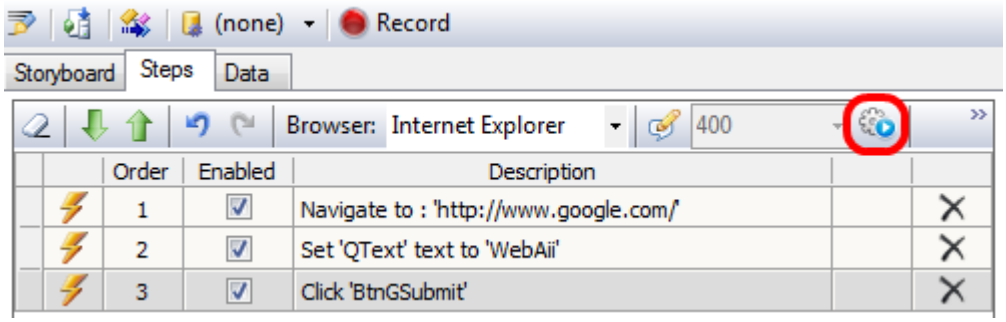
8) Click the “Go to URL” button.



9) You will notice a recording step has been added to the Steps tab.

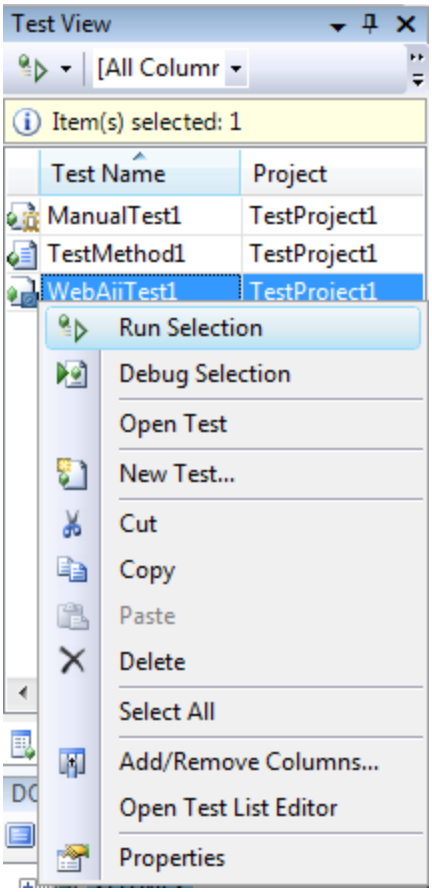
10) Type in “WebAii” in the Google search box and click the search button.

11) You will notice that two more steps have been added.

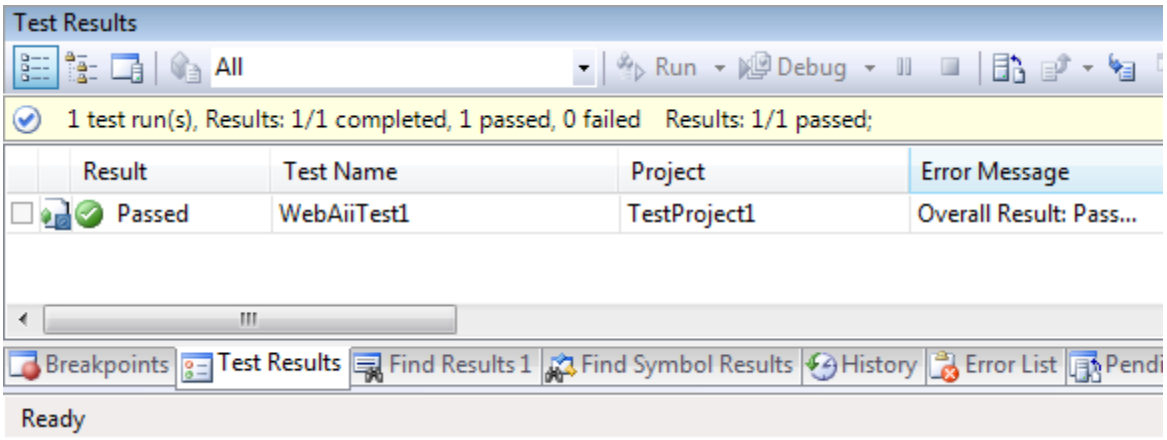


That’s how easy it is to create your first test! Save and build your project. To execute your test you have two options:

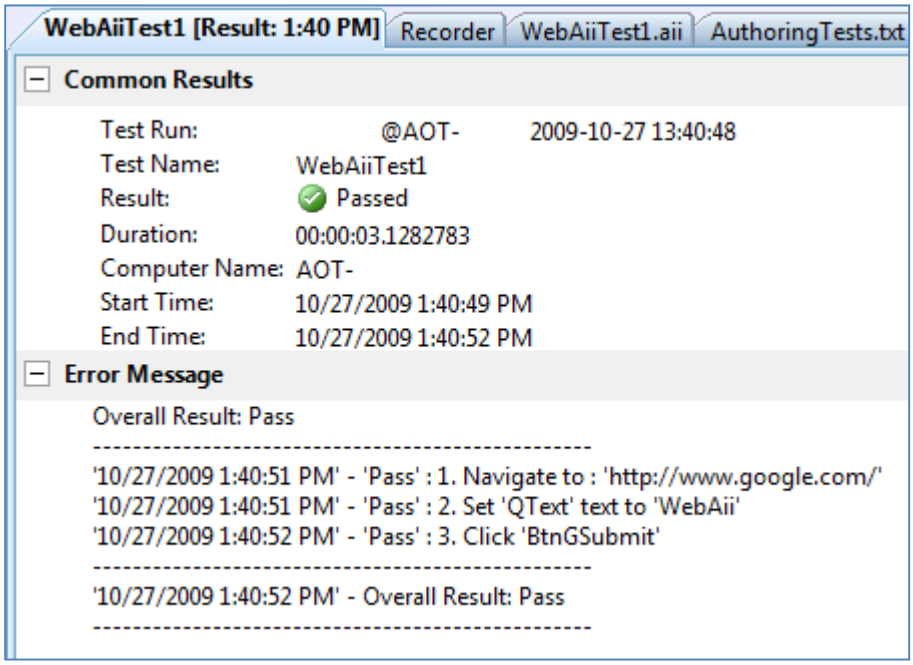
1. You can click the Quick Execute button (circled in red above).
2. To execute your test using Visual Studios testing framework
    i. Open the “Test View” tool window. This can be found by clicking on “Test” on the top menu then -> “Windows” -> “Test View”.
    ii. Right click on your test and click “Run Selection”.



An Internet Explorer window will open and the steps of your test will automatically execute in that browser window. When the test completes the browser window will close.



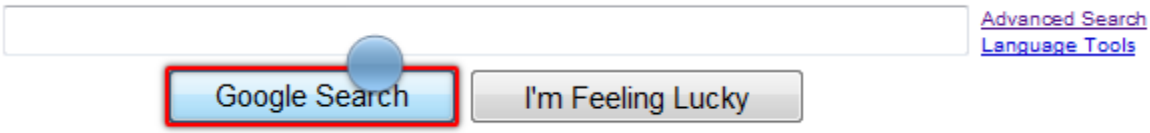
After the test runs, double click on results summary in the “Test Results” tool window to view the execution log.



Adding Quick Verification Steps

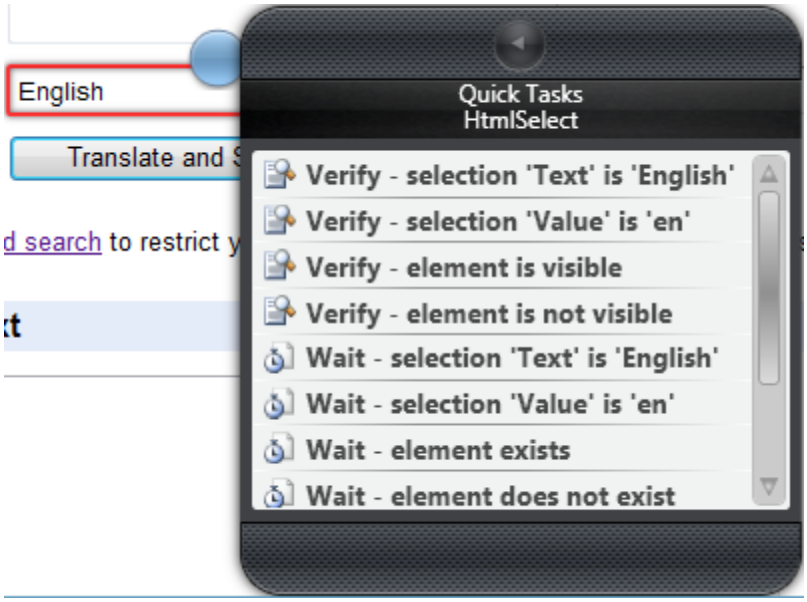
Adding verification steps is quick and simple using the Quick Tasks menu.

1. In the IE recording window hover the cursor over the element that you would like to verify a property of.
2. Wait 1 second for the little blue round nub to appear as shown below:



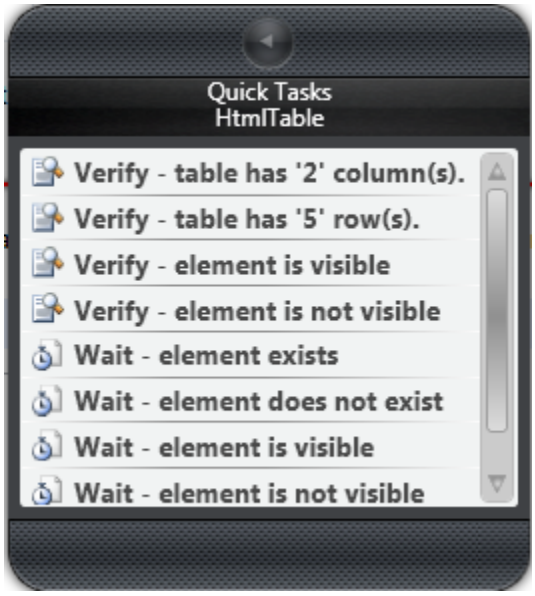
3. Click on the nub. This will open the Element Menu.

4. Click on the Quick Tasks button (circled in red above). This will open a list of available quick tasks. What is listed depends on what type of element you have selected. Here's an example with a select drop down selected:



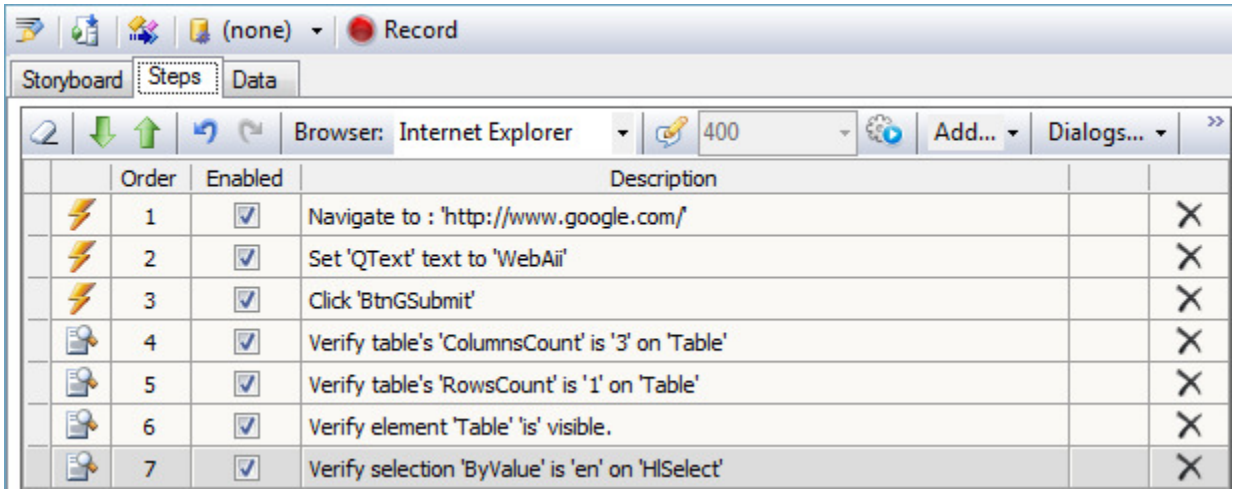
Notice that is has two “Verify selection is” items.

5. Here’s an example with a Table element selected:

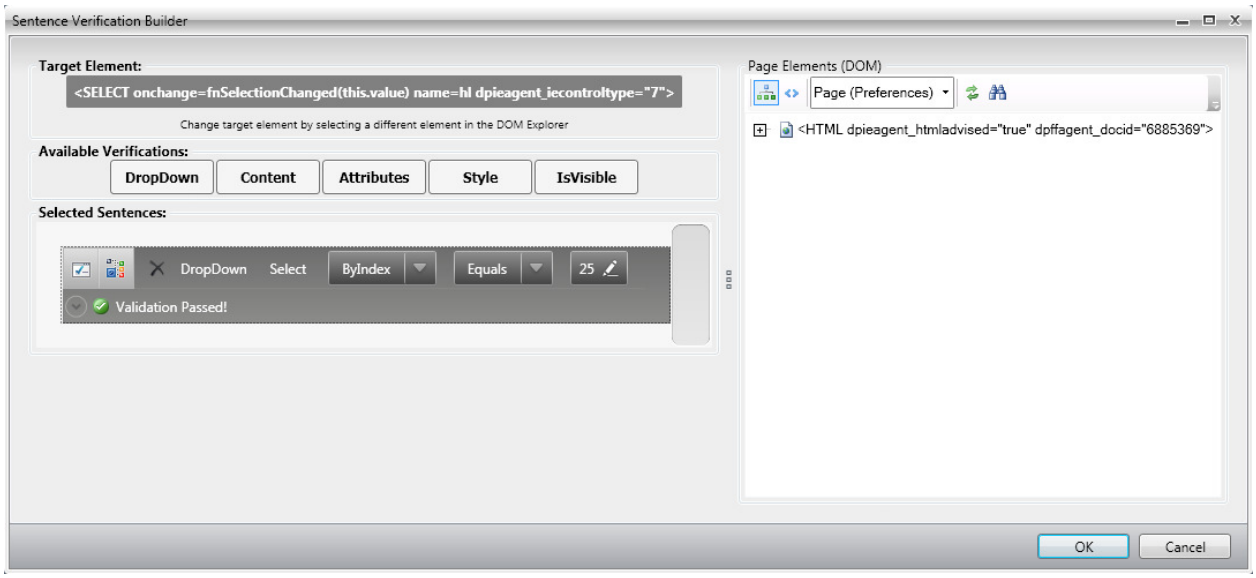


Notice that it has “Verify table has” items.

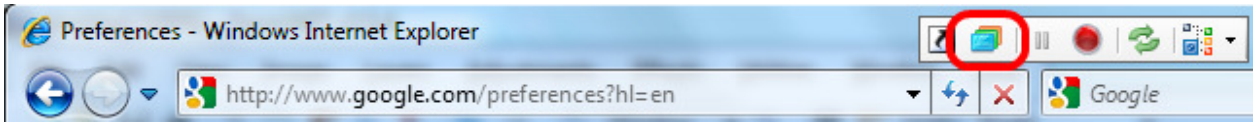
6. Double clicking one of these items will add that Verify/Wait For as a step to your test list similar to this:



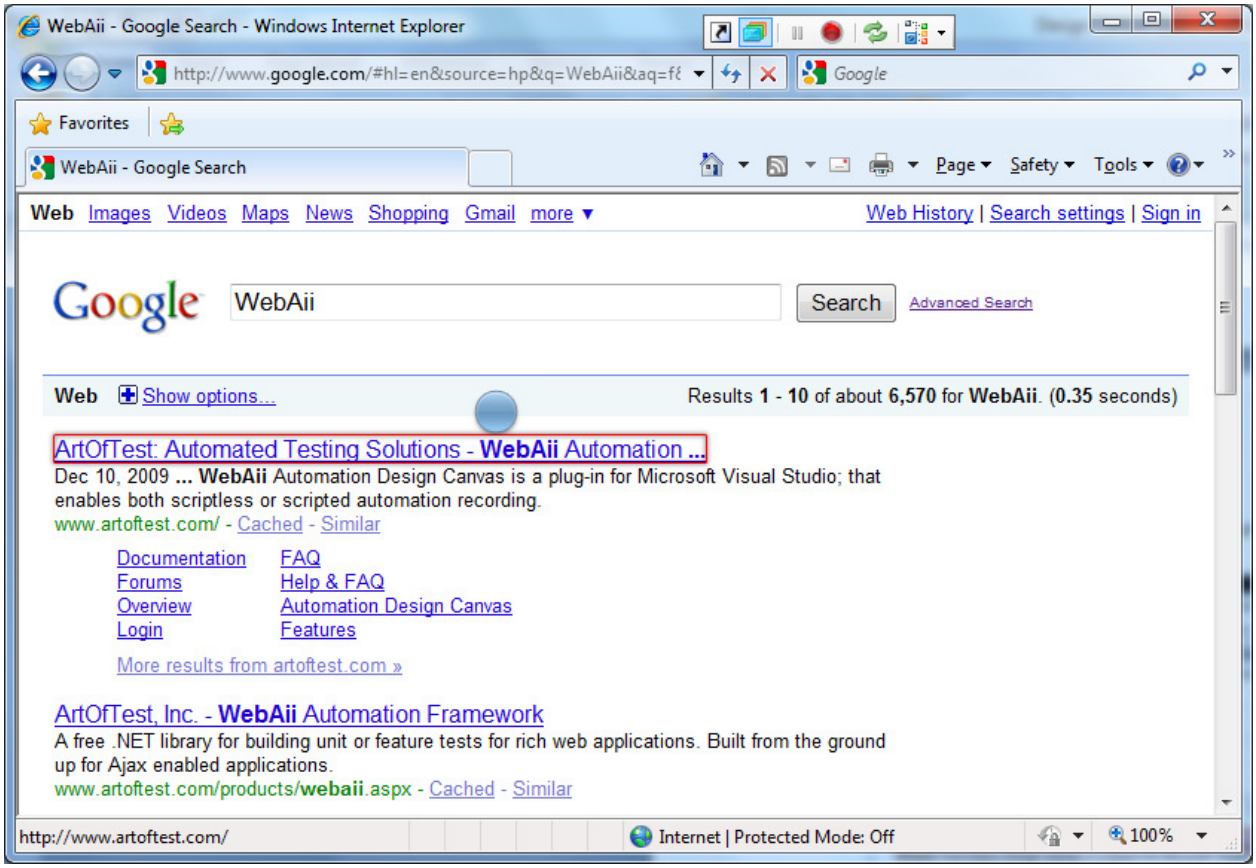
Creating Advanced Test Verifications



1) Test Verifications are added to a test as an ordered step.
2) To craft a verification step, enable the Automation Overlay Surface by clicking on the "Enable overlay" button.

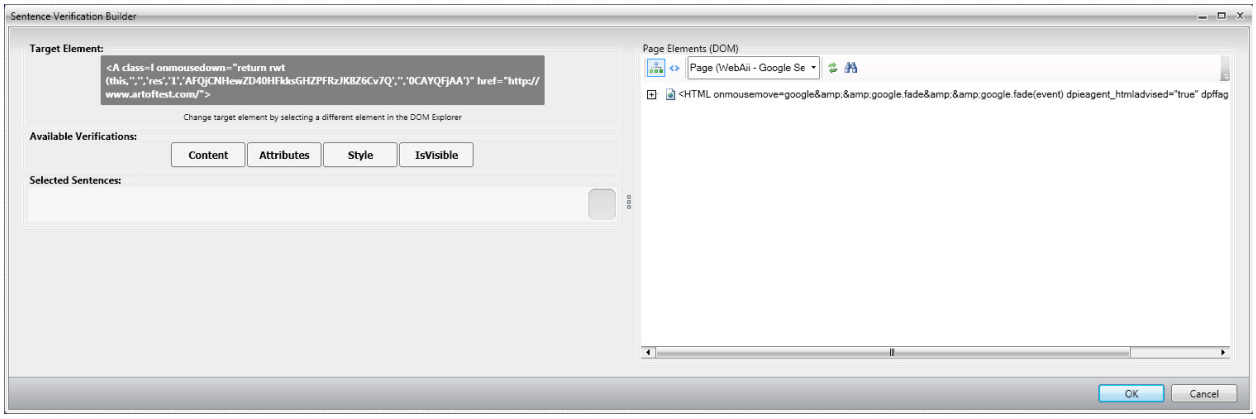


3) You will know if the Automation Overlay Surface is on by hovering over the web page and seeing each element being highlighted.



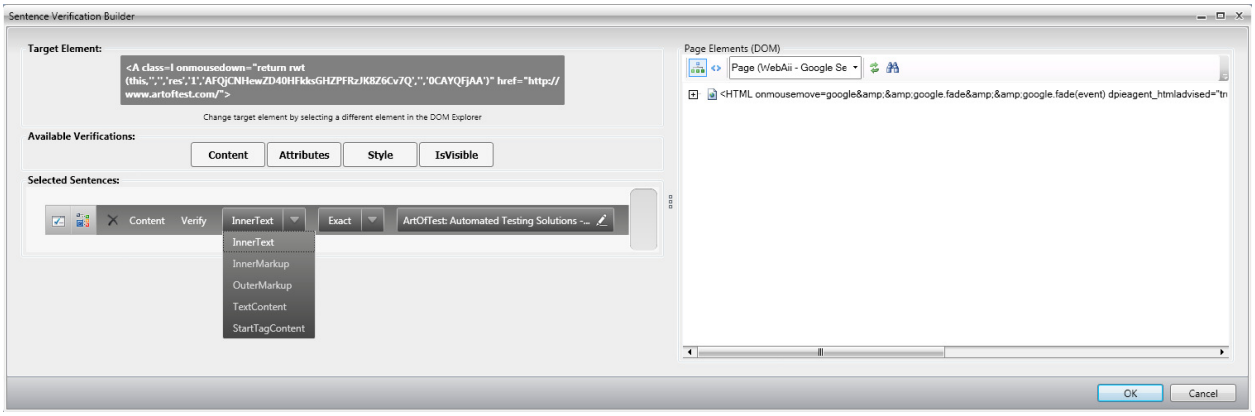
4) Select an Element that you want to verify a property of, wait 1 second for the little blue nub to appear, and then click on the nub. This will open the Element Menu.

5) From the Element Menu, click the “Build Verification” button to open the “Sentence Verification Builder”.

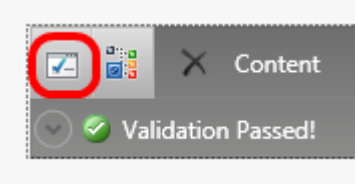


6) The verification builder offers sentence based verifications of elements.
7) Start by selecting a rule category from the drop down box.
8) During verification crafting, the content is dynamically built against the currently selected element. As you make choices, default values will be populated according to the values the element contains.

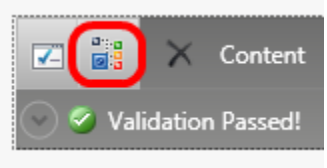
9) For example, if you choose “Content” as your verification rule then you will see three menu options. Click on the down arrow next to each option to see a list of possible values.



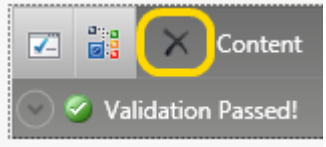
10) You can validate that your verification is valid by clicking on the “Verification” icon.



11) You can also locate the current element in the DOM tree by clicking the “Locate in DOM” icon.

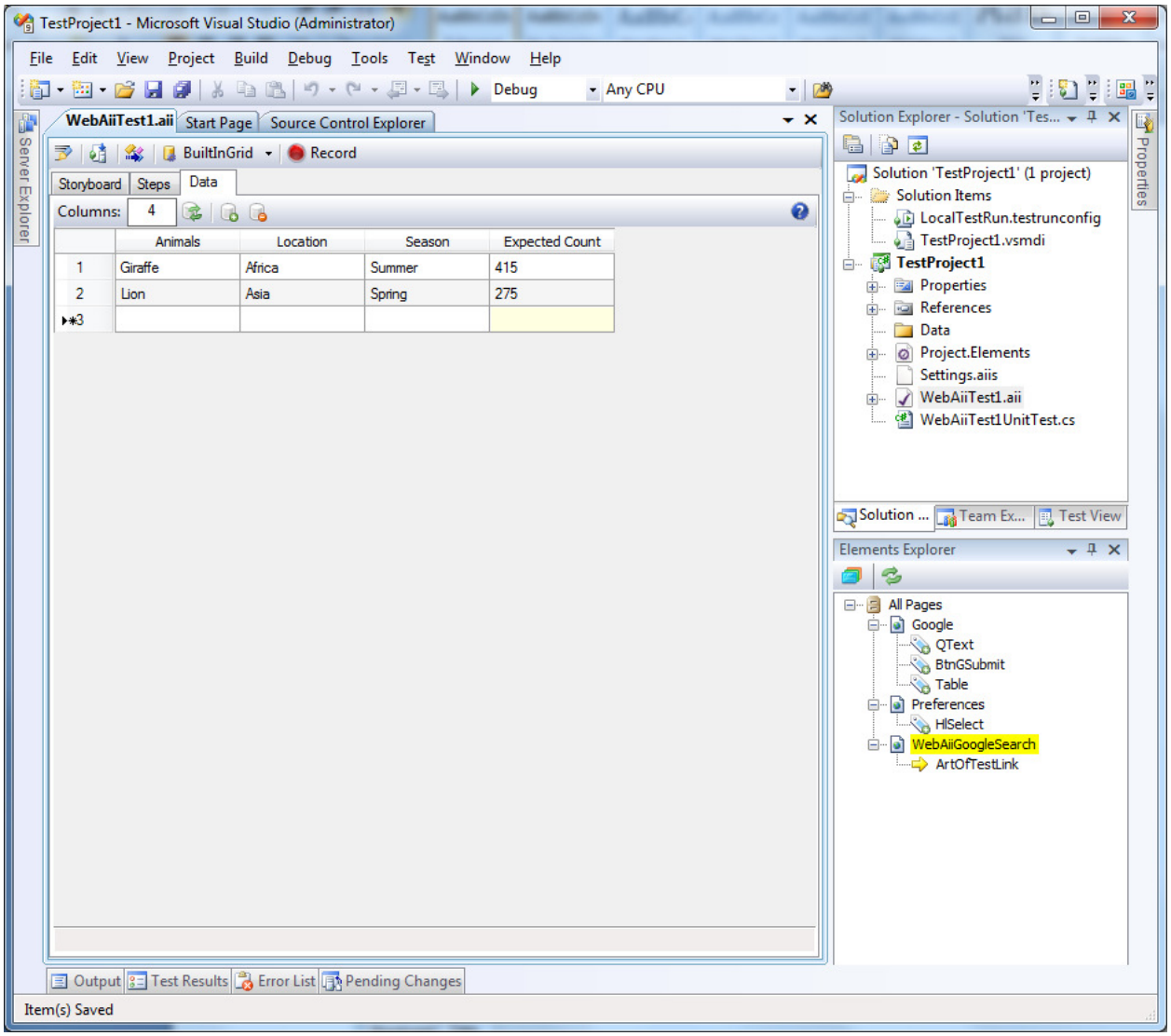


12) To delete or start over, click the “Delete” icon.

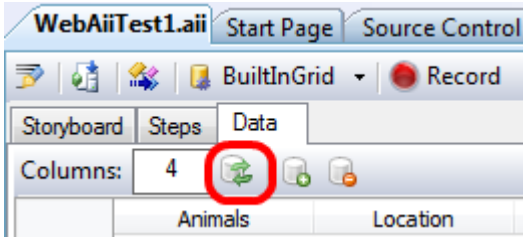


13) Verifications can be crafted to verify many different values, styles or attributes of an element. You can craft multiple sentences by selecting a rule and filling in the verification criteria. NOTE: Each sentence will add a separate verification step.
14) When you are finished building your verification, select OK to add it as a step to your current test.

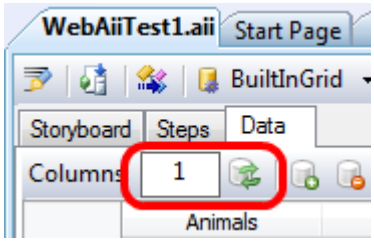
Creating a Data Driven Test



1) Create a new test as outlined above.
2) Record the Google search steps as outlined above.
3) Click on the test tab. There are three tabs at the top of this tab, one labeled “Storyboard”, another labeled “Steps” and the last labeled “Data”.
4) Click on the “Data” tab.
5) At the top of this tab are 3 buttons. Only the “Create a new data table” will be enabled. Click this button to add a new grid for your data.



6) The default grid will have 5 columns. For this data test we are going to go through five iterations of the test with different search text for each iteration.
7) Change the columns text box to 1 and click “Update”.

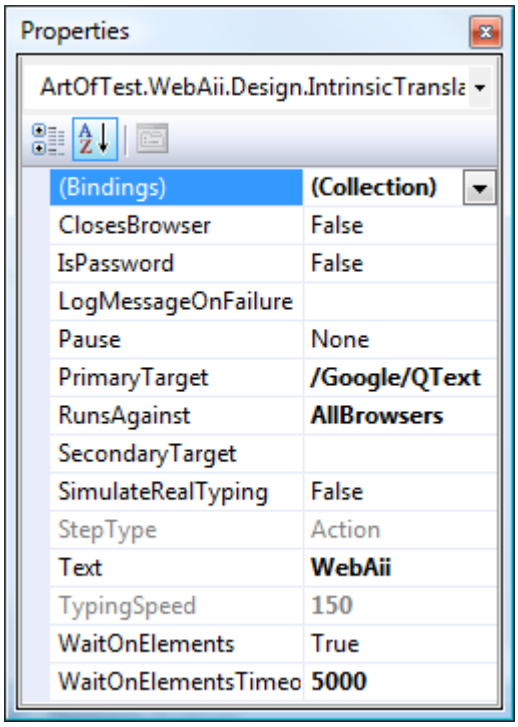


8) You now have a grid of just one column.
9) Enter any text into the first grid cell and hit enter or tab. The input will move to the second grid cell.
10) Continue entering text for 4 more grid cells. New rows will automatically be added as you type.
11) Save your test.
12) You can use data from a data array in both recorded steps and code behind methods. To use reference data from the data array in a code behind method continue with the following steps. To bind data from a data array to a recorded step, jump to step 18.
13) Convert the step to code that sets the value of the Google search text box from.
14) To reference the data from your grid use the “Data” property followed by the index of the column. For example:

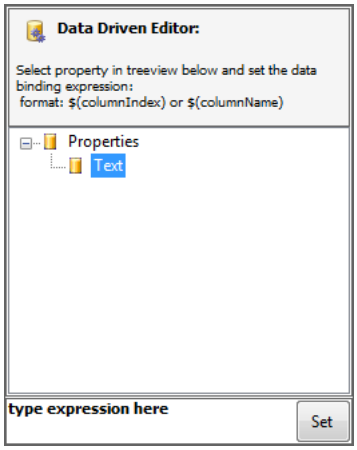
```
[CodedStep(@"Set 'QText' text to 'WebAii'")]
public void WebAiiTest1_CodedStep()
{
    // You can reference the column by index
    Pages.Google.QText.Text = (string)Data[0];
    // Or by name
    Pages.Google.QText.Text = (string)Data["Animals"];
}
```

15) Save and build your project.
16) Execute your test. Note that the test will be executed for each row in the data array.
17) To bind data to a recorded step, follow these steps.
18) Open the properties window by pressing the F4 key.
19) Click on the recorded step that sets the value of the Google search text box.

20) The properties for this step will appear in the properties window.



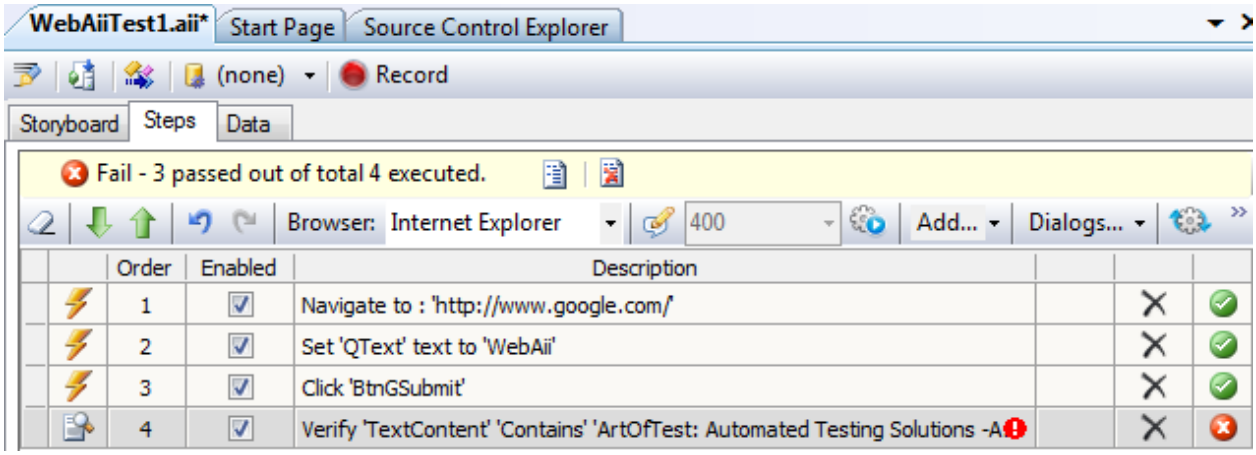
21) Click on the drop down arrow for (Bindings).
22) Click on the "Text" node in the displayed tree.
23) Enter the name of the column you want to draw data from into the text box. In our example we want the first column so we'll enter $(Animals).
24) Click the Set button



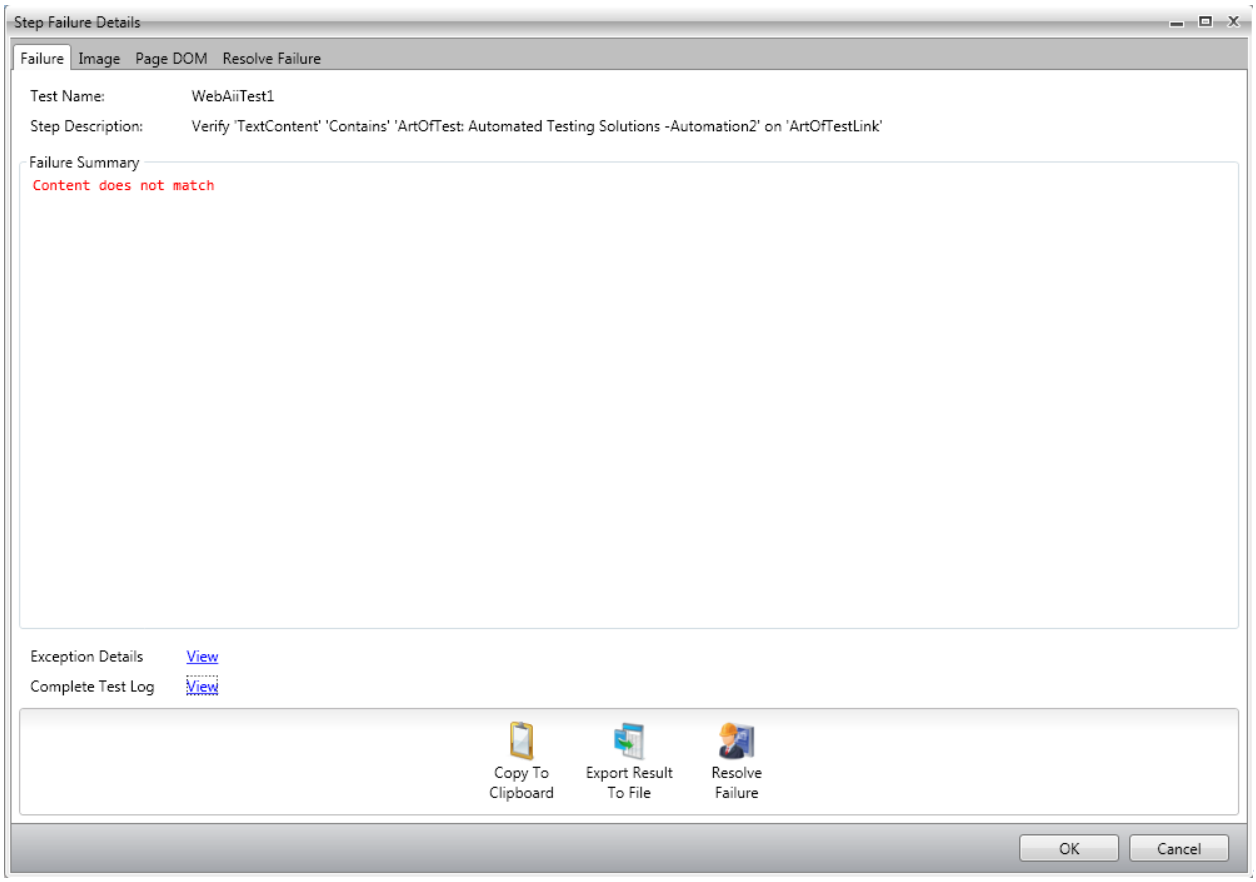
25) The data for the column named "Animals" from the data array is now bound to the Text property for that step. Instead of entering the text "WebAii" into the search box, the data stored in the data array will be entered.
26) Save and build your project.
27) Execute your test. Note that the test will execute for each row in the data array.

How to resolve test step failures

1) After running quick execution for a test, any failed steps will appear have the error icon:



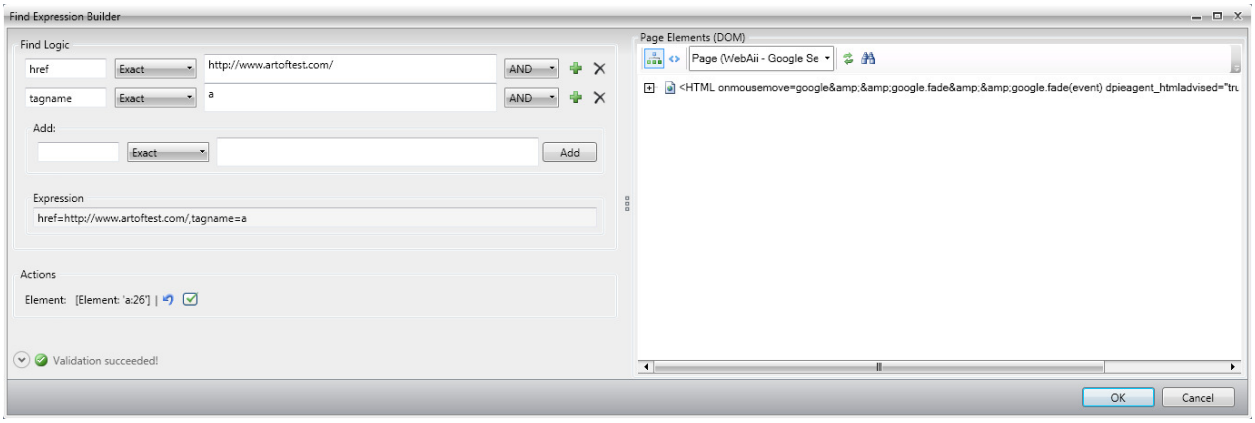
2) Double clicking on the X icon will launch the Step Failure Details UI.
3) The Step Failure Details UI will give a description of the failure as well as details about the failed step.



4) The Step Failure Details UI shows you the details about the failure and offers you the chance to update the verification if that is what is needed.

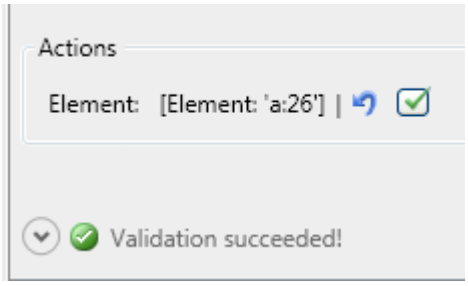
5) At this point the tester needs to decide if the test needs updating or the page under test has a defect. If the test needs updating, then you can simply modify the verification properties and click OK to update the test. Otherwise the test has detected a product defect and a bug should be logged in your bug tracking system.

Changing how an Element is found



1) Whenever a web page element has an action recorded against it, or you explicitly add an element to Elements Explorer, a "Find Expression" is generated that tells the framework how to find that specific element on the web page.
2) In order to change how an element is found, right click on the element in the Element Explorer tool window and select "Edit Element".
3) The "Find Expression Builder" will launch and display how the element is currently being found.
4) The "Find Expression Builder" consists of the following components:
    a. Element Property/Attribute – The first textbox specifies what property or attribute of the element to examine. Some of the things you can enter for HTML elements include: TextContent, NodeIndexPath, TagName, TagIndex, XPath or any HTML attribute (e.g. name, id, visibility, etc.). Some of the things you can enter For XAML elements include: TextContent, XamlTag, Name, TagIndex or any XAML attribute (e.g. Foreground, Content, etc.)
    b. Compare type – The compare type drop down controls what sort of comparison will be performed (e.g. Exact, Contains, StartsWith, etc.).
    c. Value to look for – The next textbox holds the value that will be used in that expression. It can be either a value to look for (such as a string, or a number) or an index path (tag index or XAML index), or a RegEx expression.
    d. And/Then – Find Expression Builder supports the ability to use multiple criteria for locating the correct element. When this drop down is sent to AND it means the current line and the next line must both equate to true. When sent to THEN it means you are creating a chained find expression. In a chained find expression you are specifying how to find element 'A' and then underneath element 'A' find element 'B'. This is very useful for pages in which multiple custom complex controls have been added in different panels. You can use a chained find expression to locate the correct panel and then find the correct element (such as a button) within that panel.
5) After specifying the method and criteria to use for the element find, you can click on the "Validate" icon inside the "Actions" group box to validate that the current find expression finds

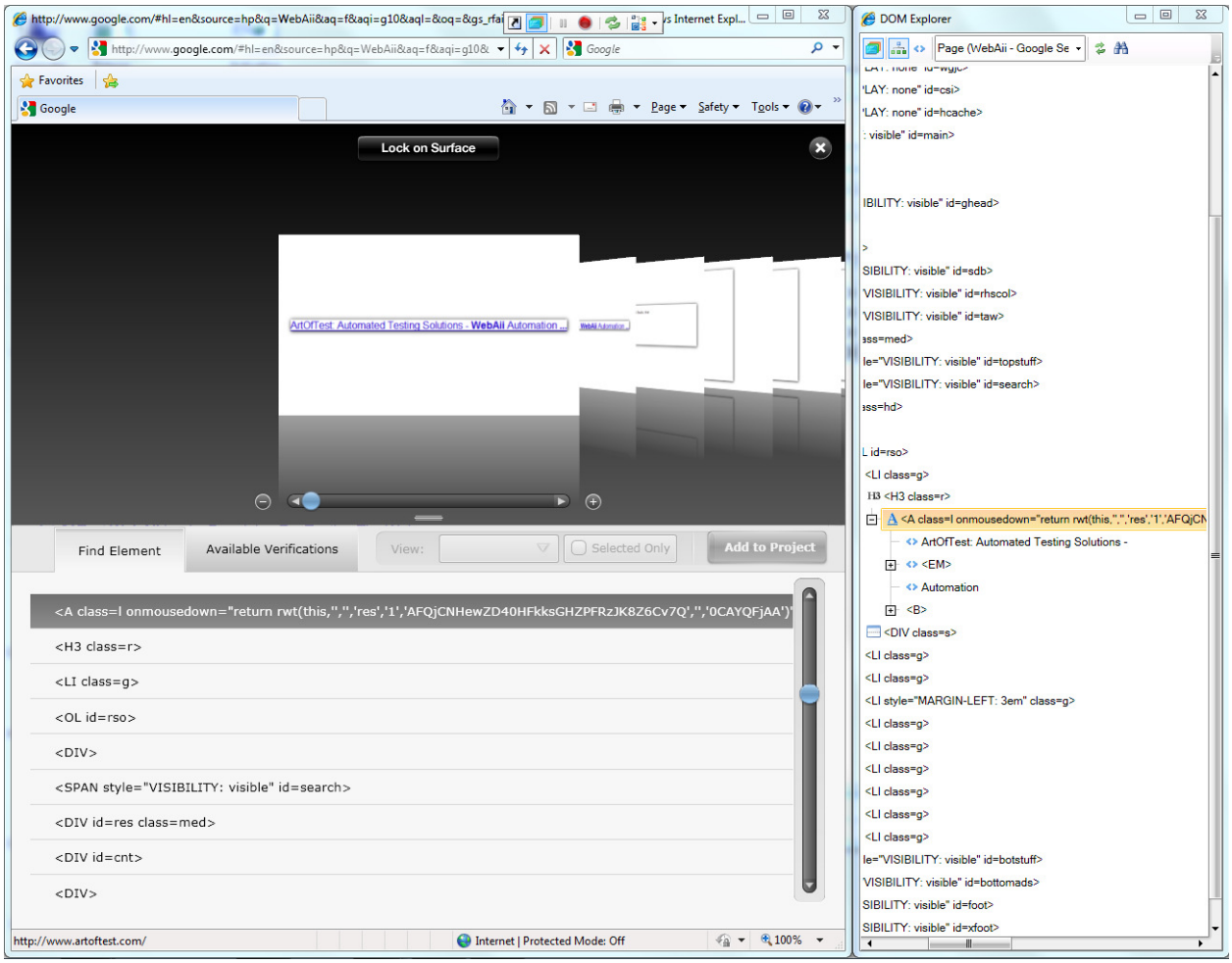
the element within the current DOM.



NOTE: For more information and a detailed discussion on crafting and using find expressions, please visit our website.

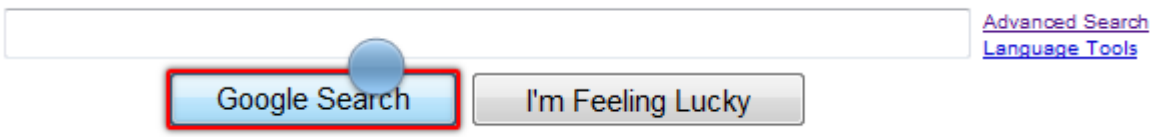
Using the 3D Viewer

The 3D viewer shows a hierarchical view starting at a selected element then traversing up the DOM tree all the way up to the DOM root. It can be used to identify/lock on elements and quickly build verifications. This is what it looks like:



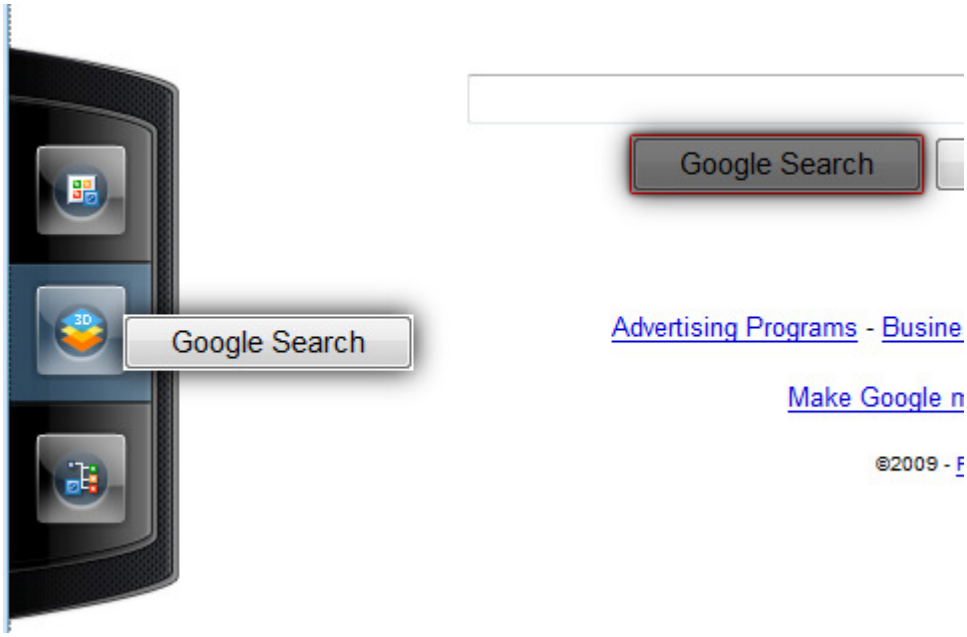
There are two methods of opening the 3D viewer.

1. Using the pop-out menu
    a. In the Recorder window hover the mouse cursor over an element you would like to examine in the 3D viewer.
    b. Wait 1 second for the little blue round nub to appear.



    c. Start dragging this nub. The nub will change into an image of the highlighted element and a pop out panel will appear on the left side of the screen with three buttons in it. Drop the image of the element you are dragging onto the middle button (the 3D viewer

button). The 3D viewer will open and the element you were dragging will be selected in the viewer.



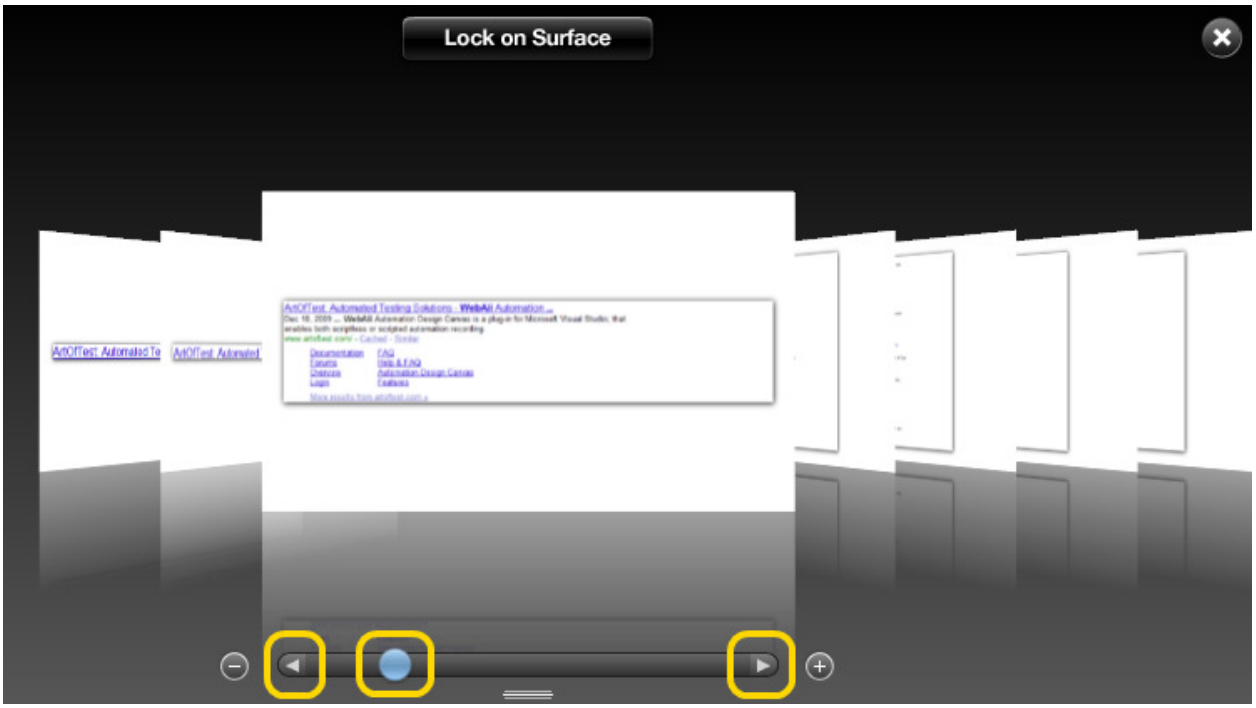
d. If you drag the element onto the top button it adds that element to the Elements Explorer tool window.
e. If you drag the element onto the bottom button it locates that element in the DOM tree and highlights it in the DOM Explorer tool window.

2. Using the Element Menu
   a. Perform steps a & b above.
   b. Click on the blue nub. The Element Menu will open.
   c. Click the View 3D button.

   d. The 3D viewer will open and the element you were dragging will be selected in the viewer.

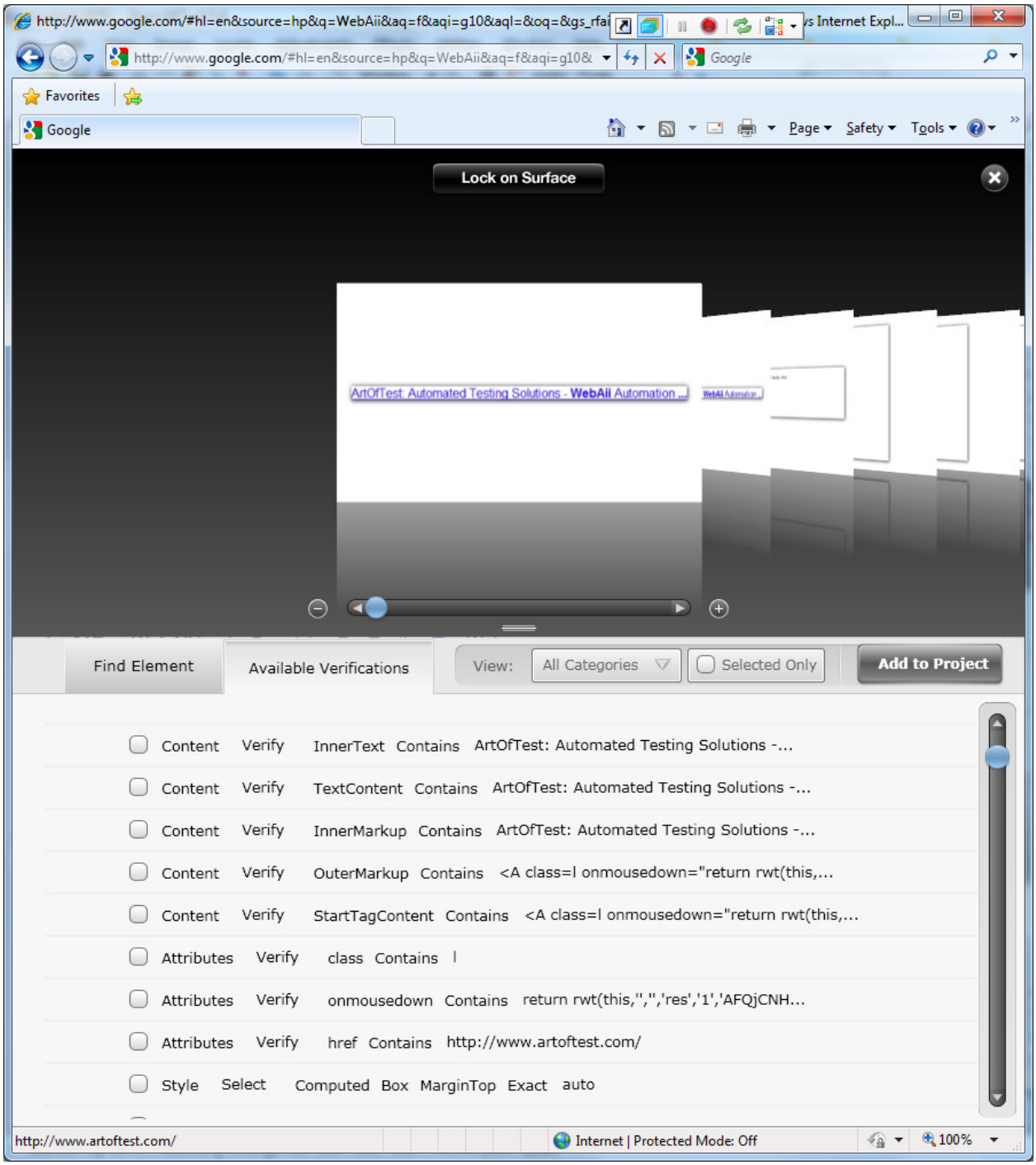
Once the 3D viewer is open you can:

1. Traverse the DOM layers by:
    a. Clicking on an element in the element list.
    b. Dragging the blue nub left/right (shown below).



    c. Clicking on the left/right arrows (shown above).
2. Increase/decrease magnification of the screenshot by clicking the + or – symbol (shown above).
3. Clicking on the “Lock on Surface” button will:
    a. Close the 3D viewer returning you to the IE recording window.
    b. Highlight that element.
    c. Open the Element Menu.

4. When you click on the “Available Verifications” tab a list of prebuilt verifications is displayed.



5. The controls in the View group box control which of the available verifications will be displayed. You can change the category being displayed using the Categories drop down. You can limit the view to only the verifications you have selected by checking the “Selected Only” checkbox.
6. You can quickly add verifications by checking the checkbox (or checkboxes) for the verification(s) that you want and then clicking Add to Project.

7. There are two buttons on each verification in the list. The first button (the one on the left) will test whether or not the verification passes or fails against the currently loaded web page. The other button will locate the selected element in the DOM tree and highlight it in the Dom Explorer tool window.
8. You also have the option of modifying the verification before you add it to the project. Use the drop downs and the edit boxes to modify the verification before clicking “Add to Project”.

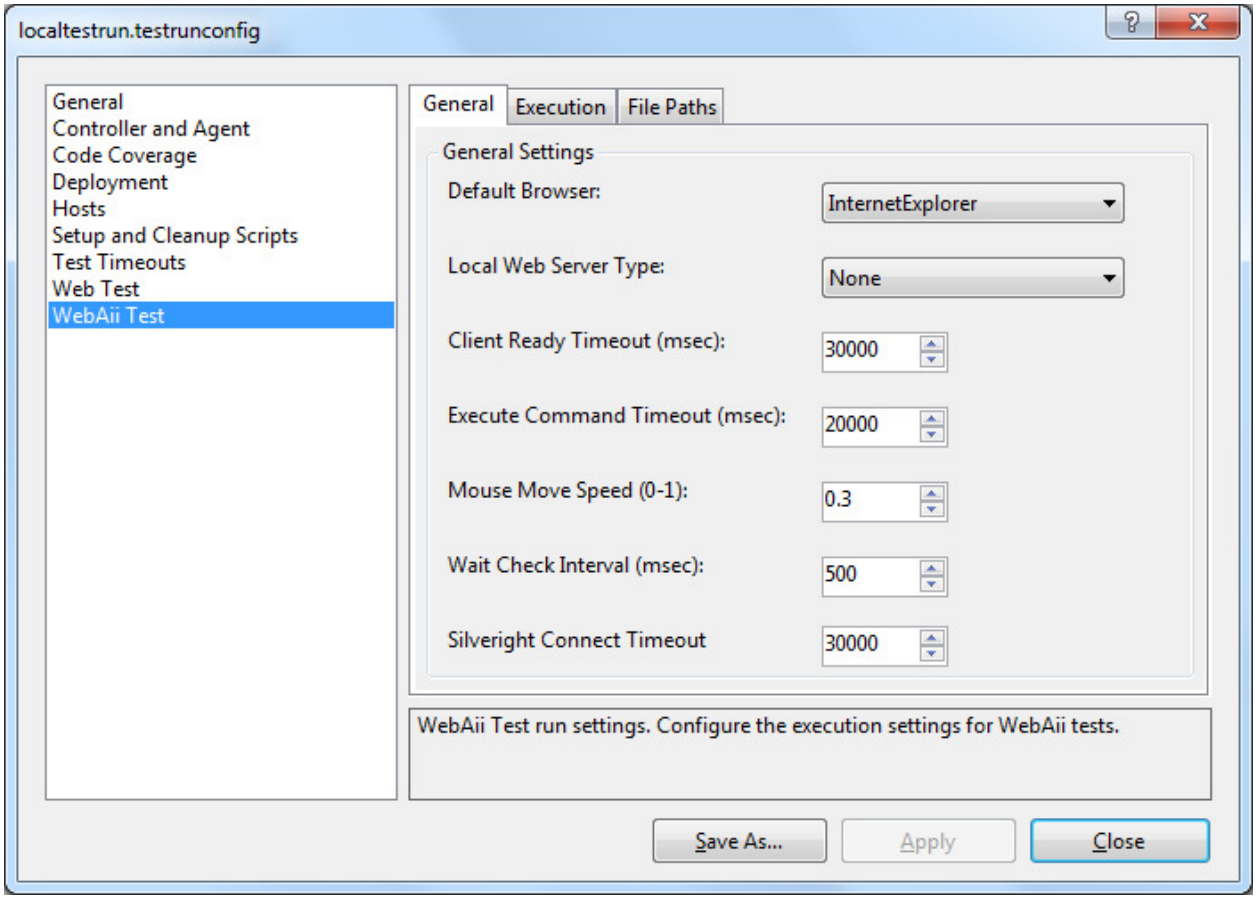
Performing Common Automation Tasks

This picture shows an exploded view of the Element Menu with all of the different tasks you can accomplish with it:

WebAii Test Run Configuration

To configure WebAii Framework settings for execution:

1) Double click on “localtestrun.testrunconfig” under “Solution Items” in the “Solution Explorer”.
2) On the left side of the Test Run Config dialog click on the item labeled “WebAii Test”.
3) This will give you a page with three tabs to configure all the run time configuration settings for all test executions within this project.

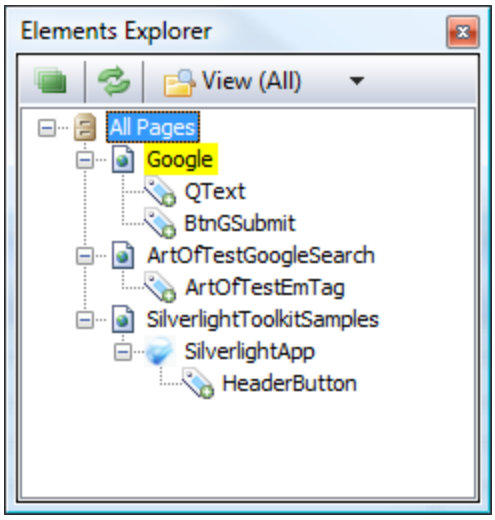


Recording Toolbar Overview

The recoding toolbar attached to the IE recording window contains (from left to right)

a. Go back to host – Makes Visual Studio active
b. Enable / Disable the Automation Overlay Surface
c. Pause recording
d. Start recording (Enable the capturing of actions)
e. Reconfigure recorder – If the communication link between Visual Studio and the IE recording window breaks for some reason, this button will restore the link.
f. Show the DOM Explorer

Element Explorer Tool Window Overview



The "Elements Explorer" tool window maintains a list of all Elements within the current project. It provides a one stop shop to help maintain elements and the way they are found during execution. The Elements Explorer menu bar has these buttons:

a. A green icon at the far left that controls highlighting of elements on the recording surface as they are selected in the tree view. Clicking on the button turns on element highlighting. Clicking on the button again turns off element highlighting.
b. Refresh – Clicking on this icon refreshes the display of the elements in Elements Explorer. You seldom should have to do this because Design Canvas normally automatically refreshes the window properly.
c. View All or View Current Page only.

The tree view is organized by Page -> Frame -> Test Regions -> Elements. The hierarchy is maintained according to where the element was located on the page. For example, if there are no frames or regions then elements for that particular page will be listed under the "Page" node. Each Page Node has a context menu with two choices:

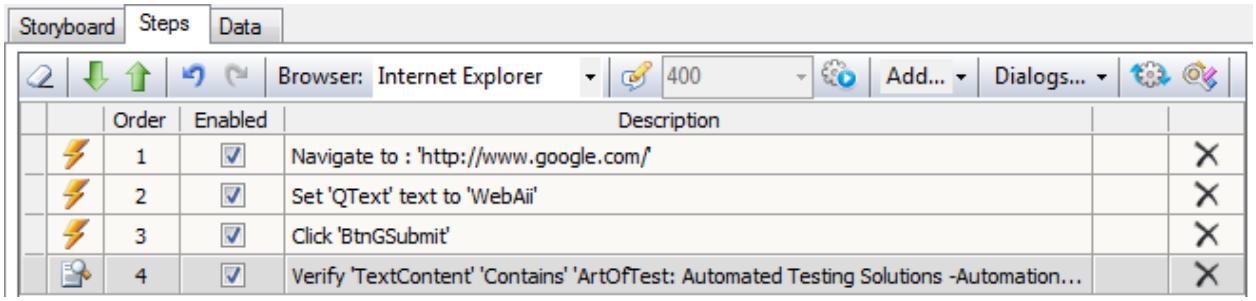
a. Load Page – loads the page in the recording surface.
b. Validate All Elements – validates that all elements can be located on the current page using the current Find Expression settings.

Each Element node has a context menu with three active choices (when the page is loaded in the recording surface):

a. Edit… allows you to change the Find Expression of the Element (the way the element is found).
b. Locate in DOM Explorer highlights the element node in the DOM Explorer tool window.
c. Load Page… loads the URL the element belongs to into the recording surface window.

Clicking each element will display the properties for that element in the Properties tool window.

Steps Tab Overview



The “Steps” tab provides the list of steps contained within the currently selected test. Each step has:

a. Type Icon – represents the type of step: Action, Verification, Coded.
b. Order of the Step.
c. Enabled checkbox – whether the step will run during execution.
d. Step description.
e. Continue on failure indicator – controls whether or not the test will stop if that verification step detects a failure.
f. Delete Button – deletes the step from the test.

The menu bar for the Steps tab contains (from left to right):

a. Clear all test steps – deletes all steps from test.
b. Move Step Down – moves selected step down one in the sequence.
c. Move Step Up – moves selected step up one in the sequence.
d. Undo – Restores a step you just deleted or changed to its previous state & location.
e. Redo – Re-applies the edit that was undone with the Undo button.
f. Browser Drop Down – selects which browser to use for quick execution.
g. Enable Test Annotation – enables test annotation during Quick Execution. If you want to change the execution across all your tests then you need to change the setting in the test configuration as previously explained.
h. Execution Delay – sets (in milliseconds) the time to delay between each step execution.
i. Quick Execute – immediately executes the current test in the selected browser.
j. Add… - This dropdown allows you to add special steps that can’t be recorded. These steps include:
    i. Capture Desktop – Captures a screenshot of the desktop and stores it in a file.
    ii. Capture Browser – Captures a screenshot of the browser window and stores it in a file.
    iii. Custom Annotation – Allows you to add your own annotations to the test.
    iv. Test as Step – With this option you can include another test as a test step in this test. Design Canvas will run the other test when it comes to this step and, when complete, resume running the rest of the steps of this test.
    v. Wait x msecs – Causes the test to pause for number of milliseconds you specify.
    vi. Clear Cookies – Will clear all of the browsers cookies.

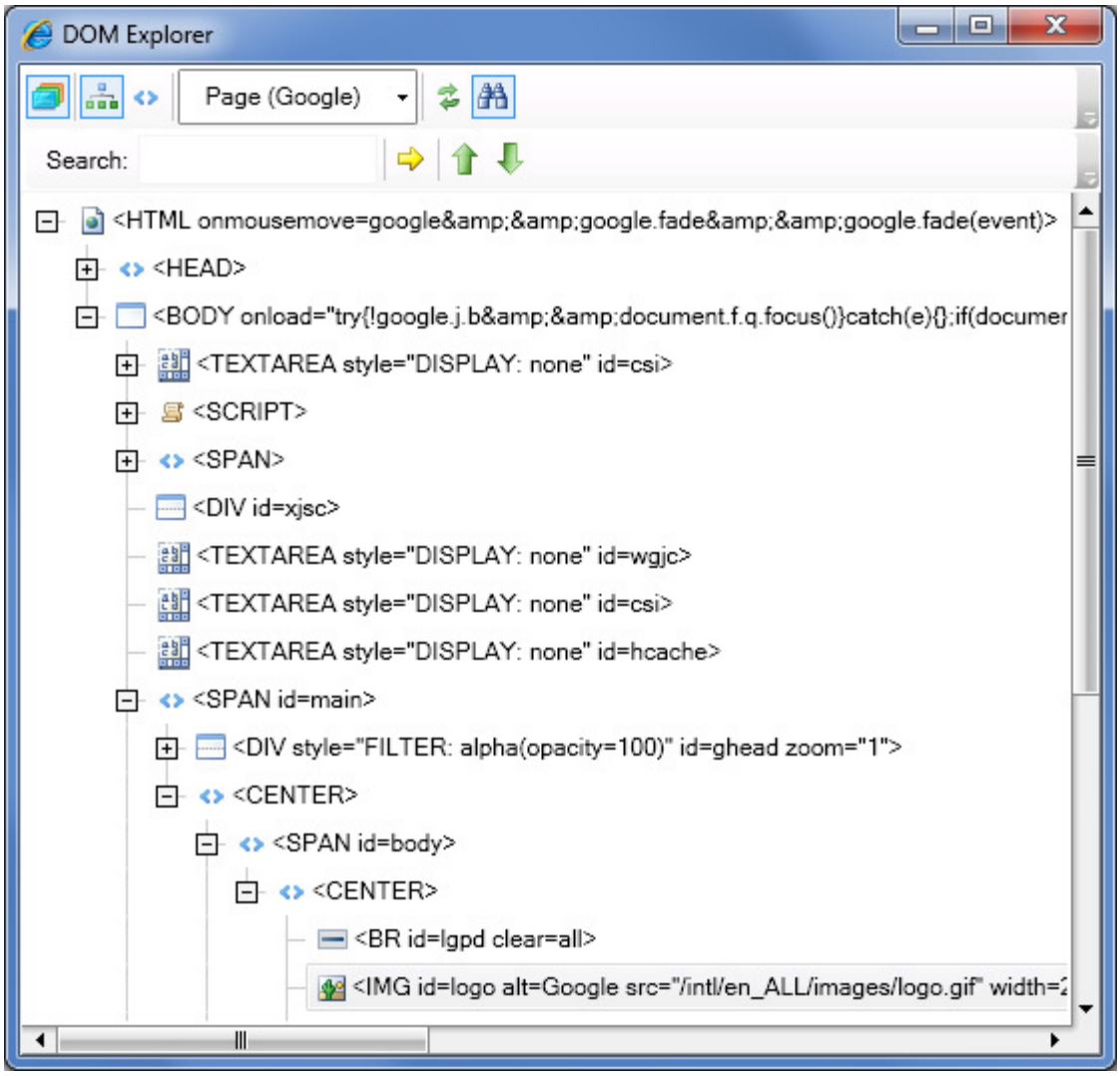
vii. Wait for Url – Waits for a specific URL to appear in the navigation bar of the browser. This is very useful for page redirection or form posting where there can be delay in processing by the web server.

viii. Inspection Point – Pauses the test and opens a DOM window with the current DOM. You can then study the DOM tree. The test will automatically resume when you close this DOM window.

k. Dialogs… - This dropdown allows you to add specific dialog handlers as test steps. The dialogs that can be handled include: Alert, Confirm, Logon, File upload, File download, and a special Generic dialog handler which can simply close most dialogs.

Clicking a step will display the properties for that step in the Properties tool window. You can then edit the properties of that step (e.g. the URL to navigate to, the text to enter into a text box, check or uncheck a checkbox, etc.).

DOM Explorer Tool Window Overview



The "DOM Explorer" tool window displays the Document Object Model for the currently loaded page in the IE recording window. The nodes are a hierarchical representation of all the HTML/XAML elements that make up the page. Each node in the tree is listed by:

<[Tag Name] [attributes]>

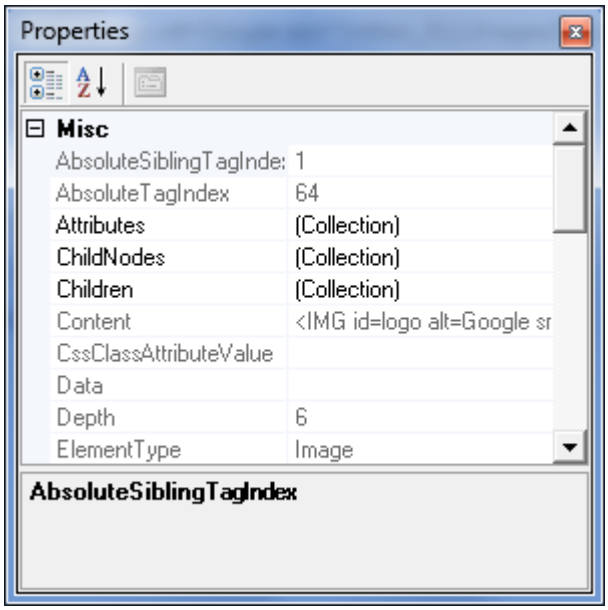
The toolbar for the DOM Explorer contains (from left to right, top to bottom)

a) Enable highlighting – When enabled, the element currently selected in the DOM Explorer will be highlighted in the IE recording window by having a red rectangle drawn around it.
b) Hierarchal View – Displays the DOM as a hierarchal tree in the order displayed on the web page.
c) TagName – Displays the DOM sorted by tag names.
d) Move to main element drop down – Lists the root element, any frames and any Silverlight applications found on the current page. Selecting one of these from the drop down highlights that element in the DOM Explorer tool window.

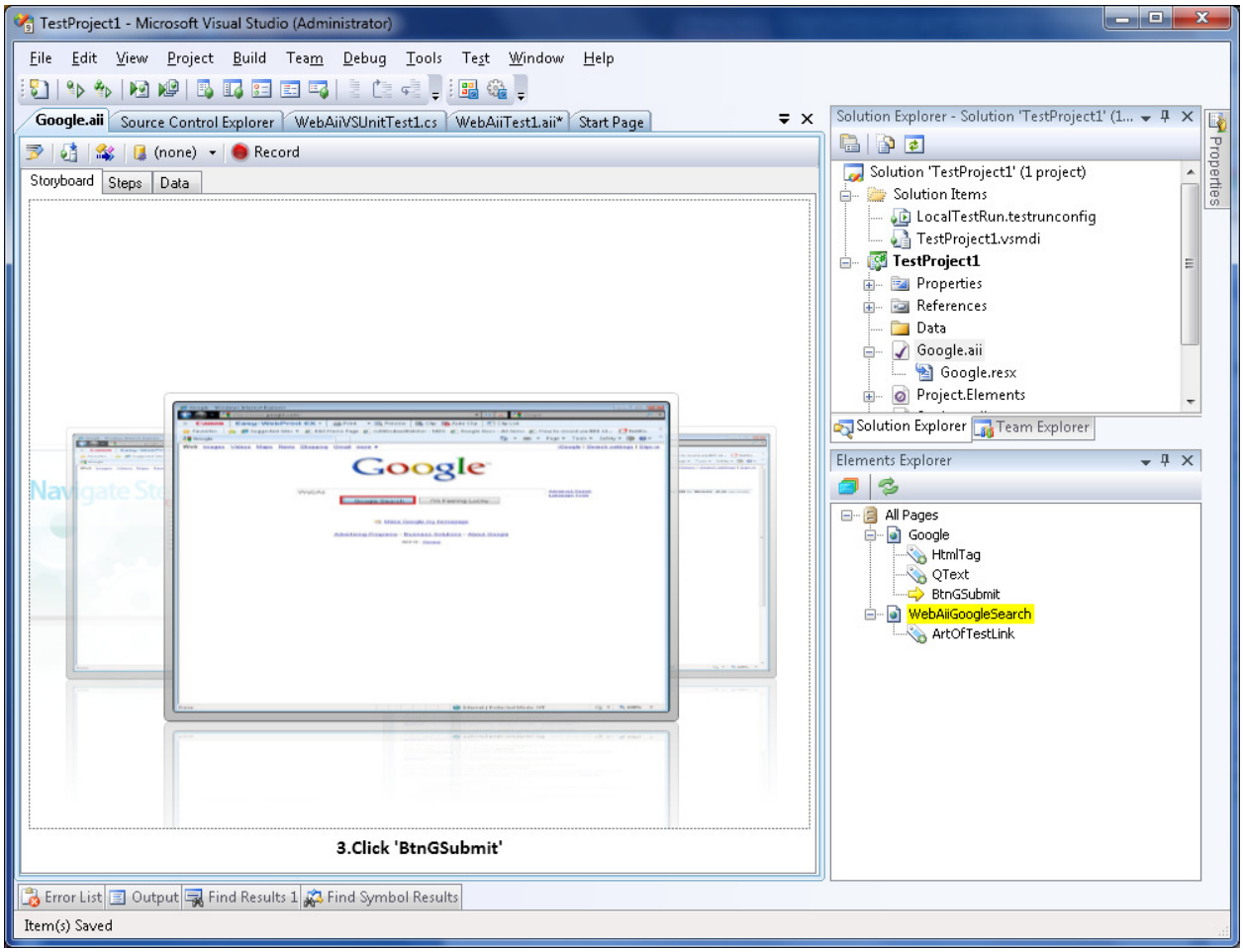
e) Refresh DOM – will rebuild the tree from the current page.
f) Search – will enable another toolbar below the first. This second toolbar can be used to search the DOM tree for elements within it.
g) Find expression text box – enter a valid Find Expression (see http://www.telerik.com/automated-testing-tools/support.aspx).
h) Perform Search – Evaluates the find expression and highlights the first matching element in the DOM Explorer.
i) Move to Previous Result
j) Move to Next Result

Right clicking a node will bring up a context menu containing these options:

a) Goto – provides a shortcut for moving to a significant layer of the DOM tree (Page root element, embedded frames, Silverlight applications).
b) Show Element Menu –Opens the Element Menu with options for the selected element.
c) Add to Project Elements – will add the currently selected element to the "Elements Explorer" tool window and craft a default FindParam for that element.
d) Copy to Clipboard – Copies the HTML/XAML to the clipboard. You can then paste the data into another application or report.
e) Properties – Opens a properties window showing detailed information about the selected element.



WebAii Test Tab Overview



Each WebAii test loaded in Visual Studio will have its own document tab allowing for step visualization and test manipulation. The test tab has a menu bar with four icons:

a) Add a code behind file – Adds a new .cs or .vb code behind file which is then logically attached to the test in Solution Explorer.
b) Convert test to VS WebTest – Generates a Visual Studio WebTest from the test and adds it to the project.
c) Generate Unit Test from WebAii Test – Generates a Visual Studio Unit Test from the test.
d) External data sources – A drop down menu you can use to bind the test to an external data source for data driven testing. You can use Excel files, CSV files, XML files, or external databases as datasources.
e) Record – Opens the "Recorder" window and directly enables recording for this test)

Note: Some recorded steps will not have an image associated with them because of the nature of the action. For example, a "Navigate To" action does not have an image.

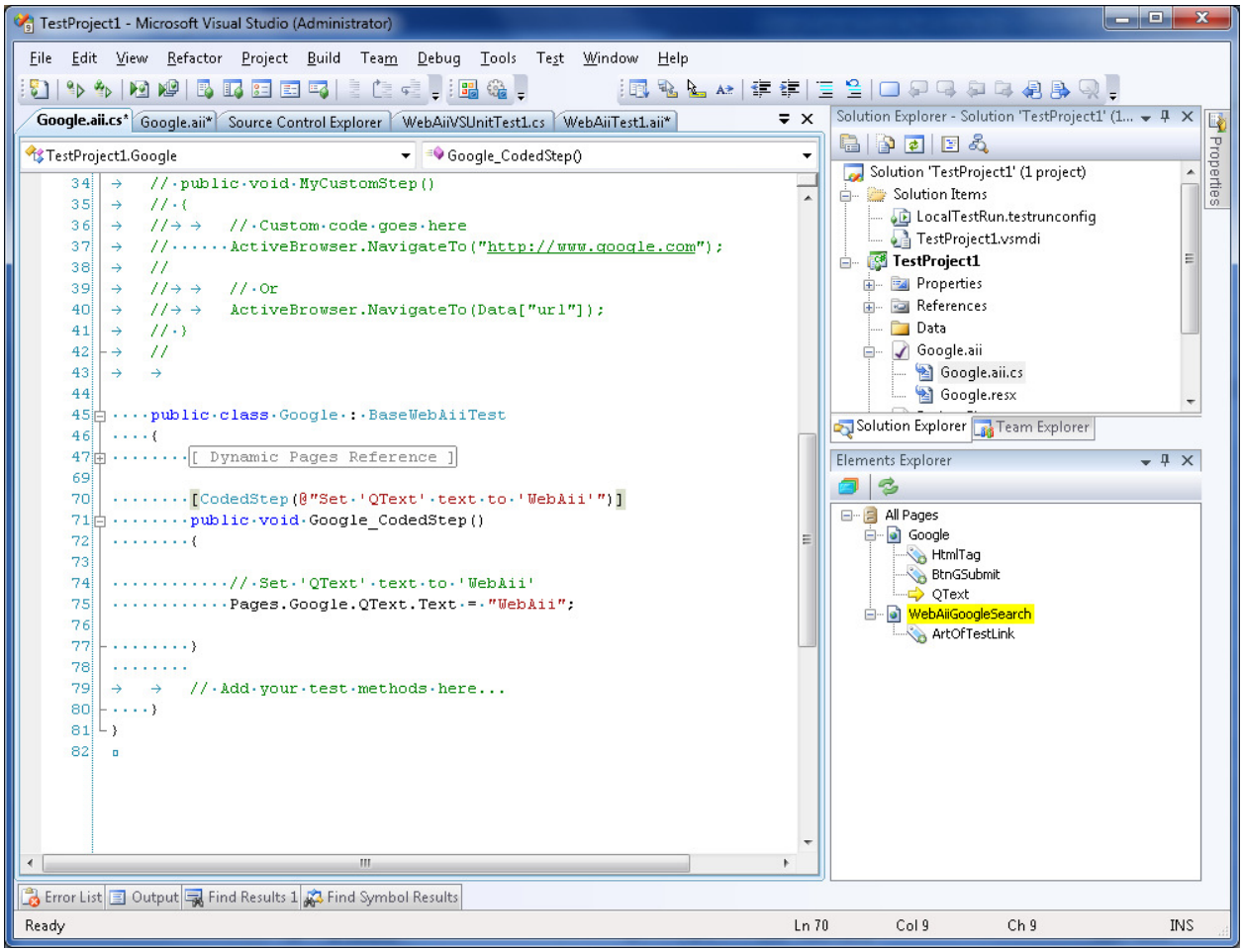
Clicking on any of the step images in the Storyboard tab will bring that image forward and move to that step on the Steps tab. Each recorded image will have the element that it was recorded against highlighted within the image.

The test tab also contains a “Data” tab. The Data tab allows for the creation of a simple Excel like data array to be used by the steps of the test. See the “Creating a Data Driven Test” section in the User’s Guide for more details.

Customizing Your Test Using C# or VB.NET Code

Now that you have the core of your test put together, you may run into a scenario that requires a test step that is more complex than can be put together using the Verification Builder or the actions provided by the Element Menu. WebUI Test Studio supports using a "code behind file". Using a code behind file you can actually write any code you need and have that code executed as a test step. The following sections describe how.

Creating a Test With Custom Code Steps

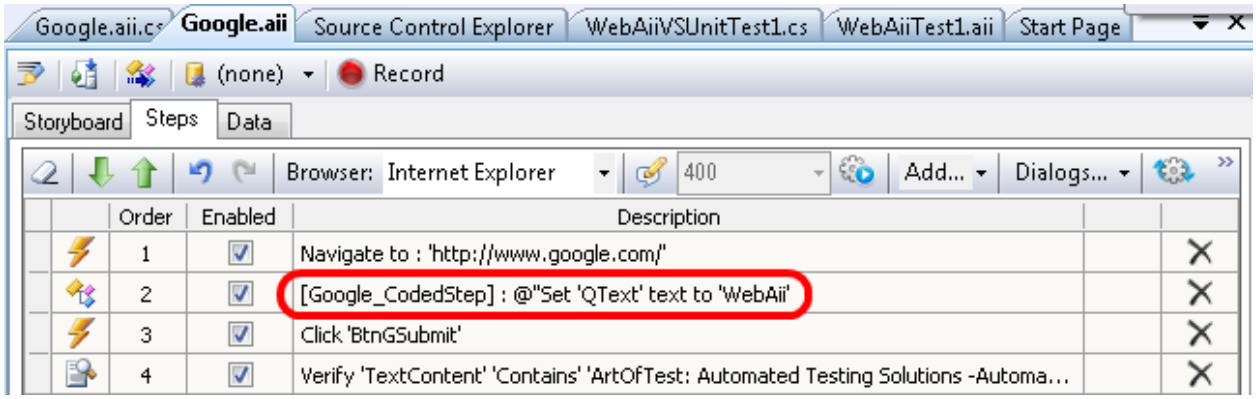


1) Create a test as outlined above.
2) Record a few steps as outlined above.
3) There are two ways of creating a code behind file for your test.
   A. On the test tab, click the "Add code behind…" button.

- Or -

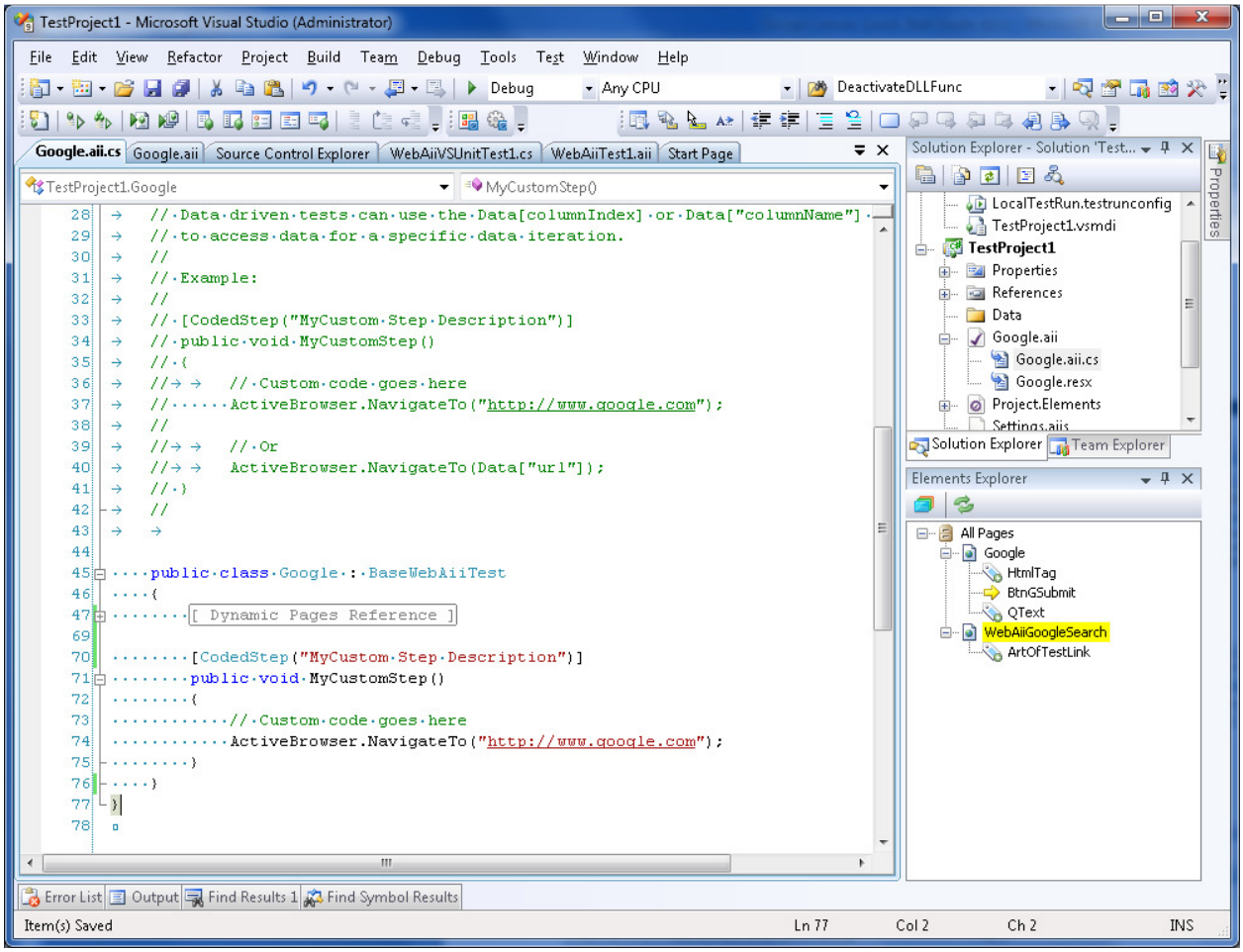
B. Right click on any step in the Steps tab and select “Customize Step in Code”.

4) Method A will create a blank code behind file while method B will create a code behind file with the selected step converted into a method matching the selected step name.
5) Note that a step with generated code will be read only in the Steps tab. The description can only be changed in the methods “CodedStep” attribute that appears in the code behind file. A coded step will be indicated with a code icon.



6) Once a step has been converted to code, you cannot convert it back to a regular step.
7) Save and build the project.
8) Execute the test.

Creating a Test That Uses a Code Behind File



1) Create a test and add a code behind file using method A as outlined above.
2) Just like a step that has been converted to code; a method in the code behind file is represented as a step in the Steps tab.
3) In order to write custom steps in code, you must add a method in the code behind file that takes no parameters and has "`void`" as its return type.
4) The method must be decorated with the "CodedStep" attribute as shown below:

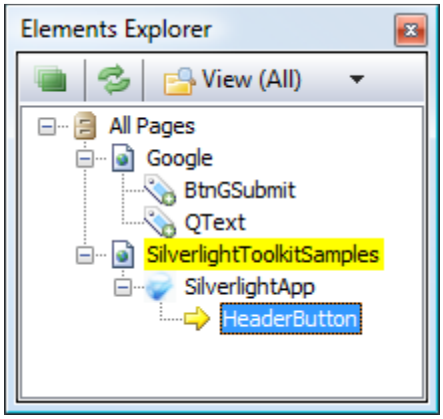
```
[CodedStep("MyCustom Step Description")]
public void MyCustomStep()
{
    // Custom code goes here
    ActiveBrowser.NavigateTo("http://www.google.com");
}
```

5) Add the above method or one similar to your liking to your code behind and click save.
6) Notice that the coded step is added on the Steps tab with the coded icon immediately upon saving. NOTE: Newly created coded steps are always added to the end of the steps. You can use the move up/down buttons to change its order of execution or drag and drop the step to the desired position.

7) The name of the method becomes the step name and the description from the Coded Step attribute is listed as the description in the steps tab.
8) Save and build the project.
9) Execute the test.
10) The code behind file has access to all the WebAii runtime objects like ActiveBrowser… etc. If you are familiar with the WebAii runtime automation framework, this will be an identical coding experience.

How to Reference Elements from the Element Explorer in Code Behind Files

1) Pages can be referenced in the code behind file in the same way they are found hierarchically in Elements Explorer.



2) For example: If the image above was your current Elements Explorer window, you could reference the "BtnGSubmit" element like this:
   a. `Pages.Google.BtnGSubmit`
3) In this example it will return the strongly typed WebAii framework "HtmlInputSubmit" type.