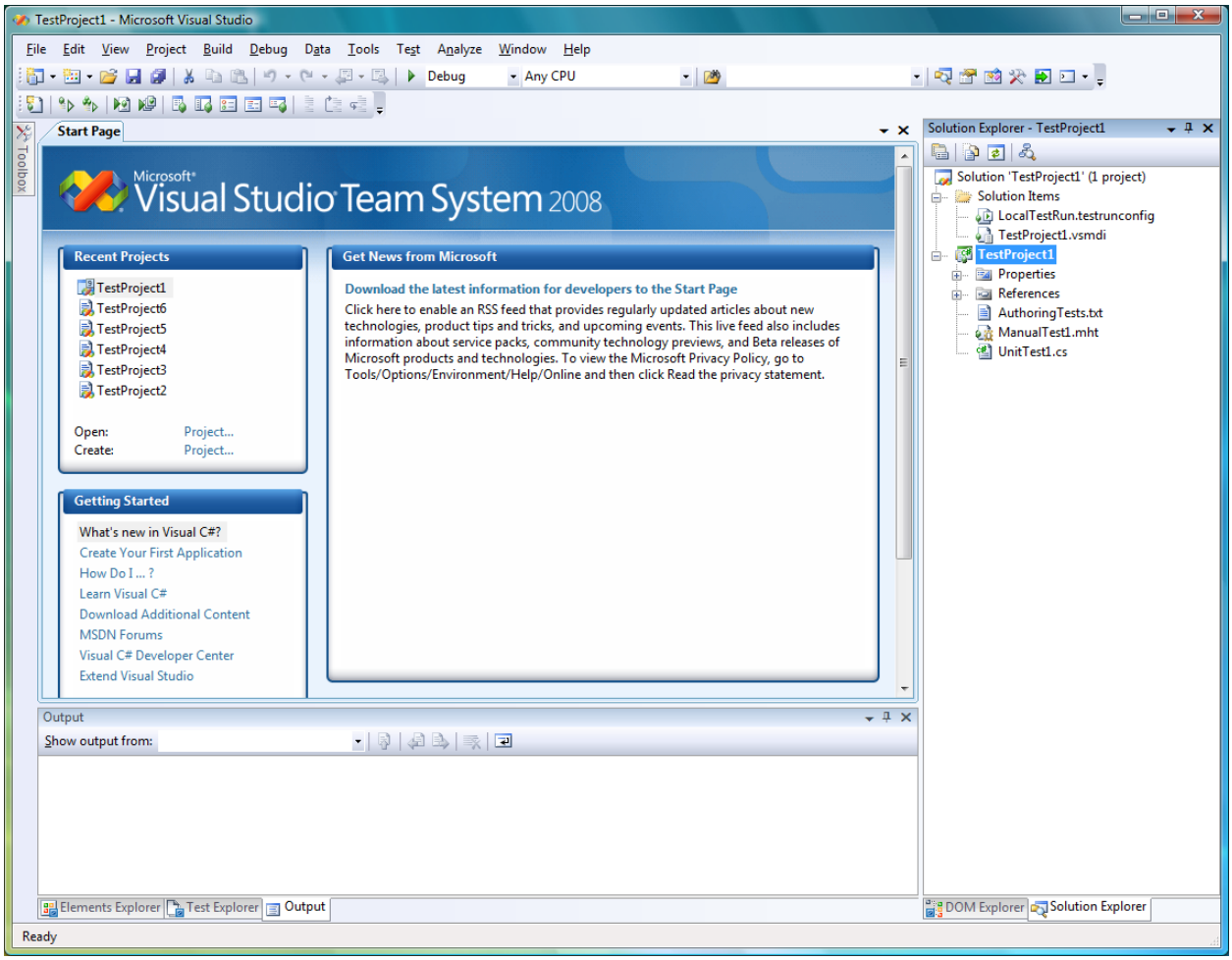
ArtOfTest Inc.

WebAii Automation Design Canvas 1.1 Quick-Start Guide

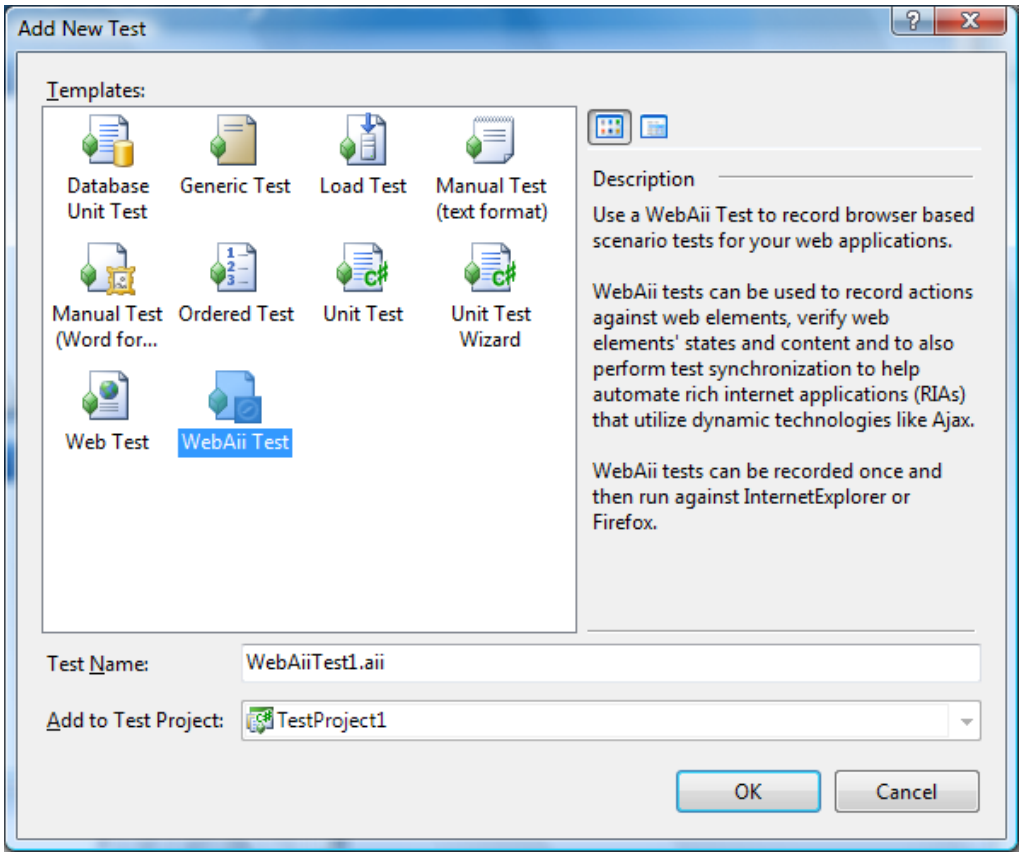
Contents

Creating and Running Your First WebAii Test

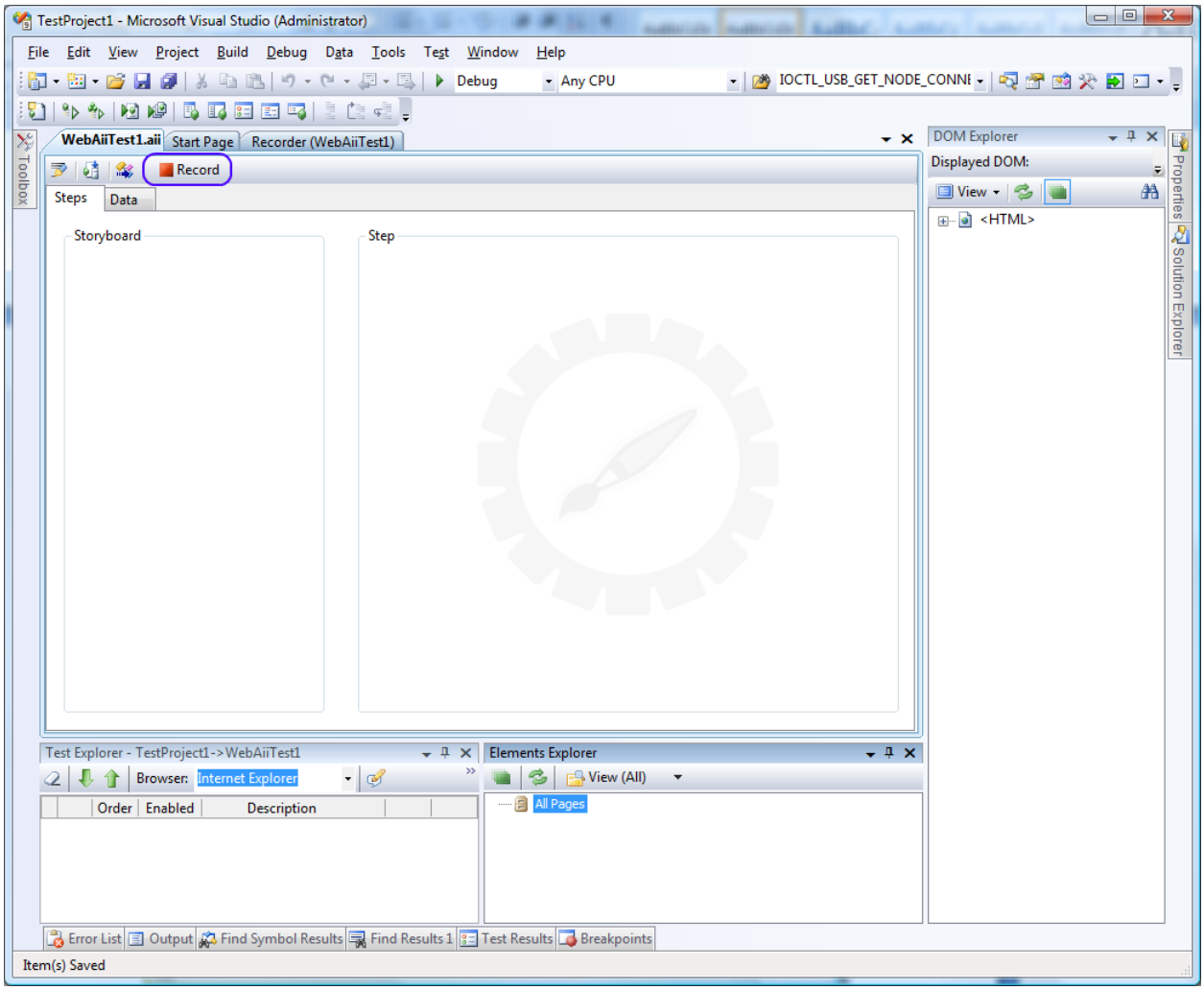
1) Open Visual Studio and create a new test project.



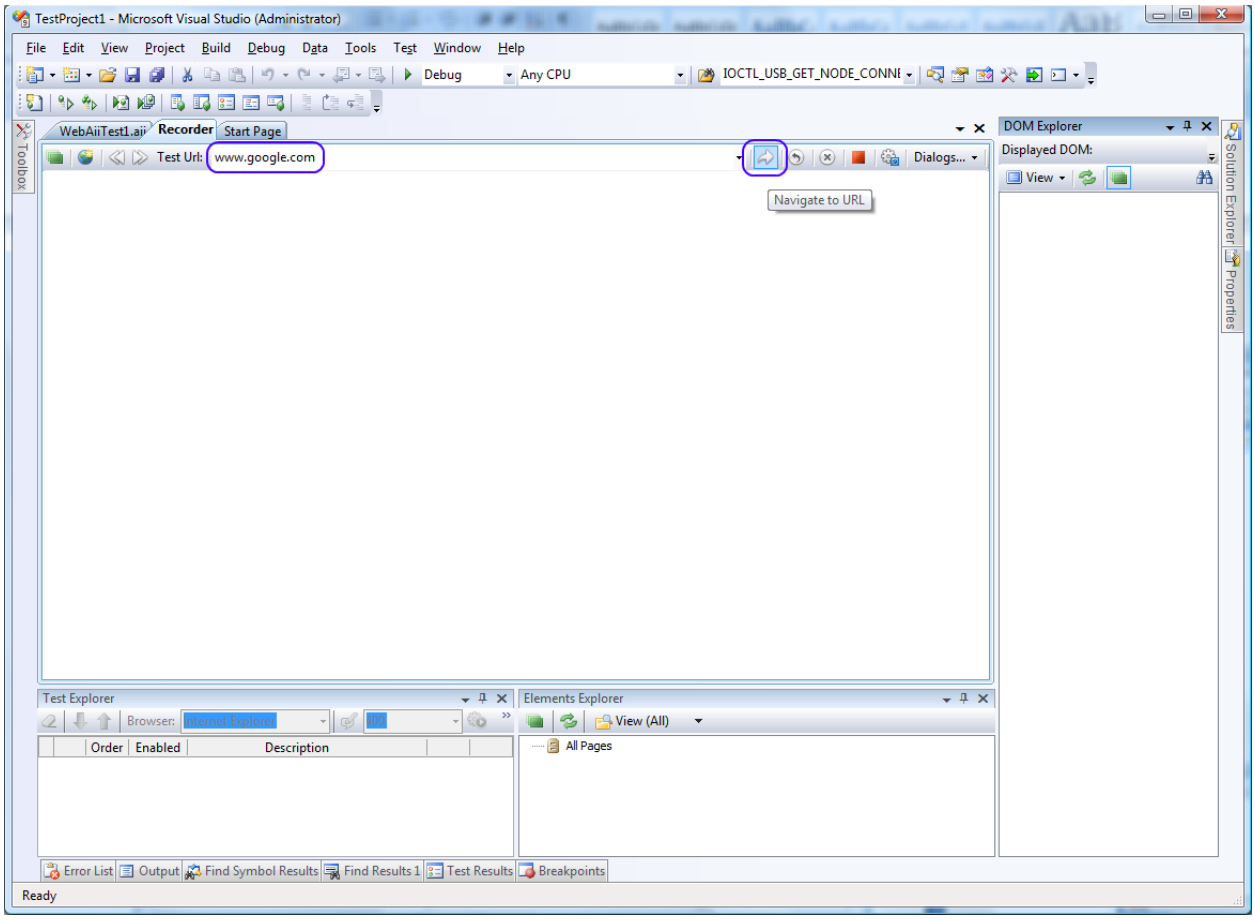
2) Right click on the project node in the solution explorer and select “Add -> New Test…”.



3) Click on the “WebAii Test” type and select OK.
4) Your new empty WebAii test should automatically open. If not double click the new WebAii Test in Solution Explorer or click on the WebAii Test tab in the document windows.

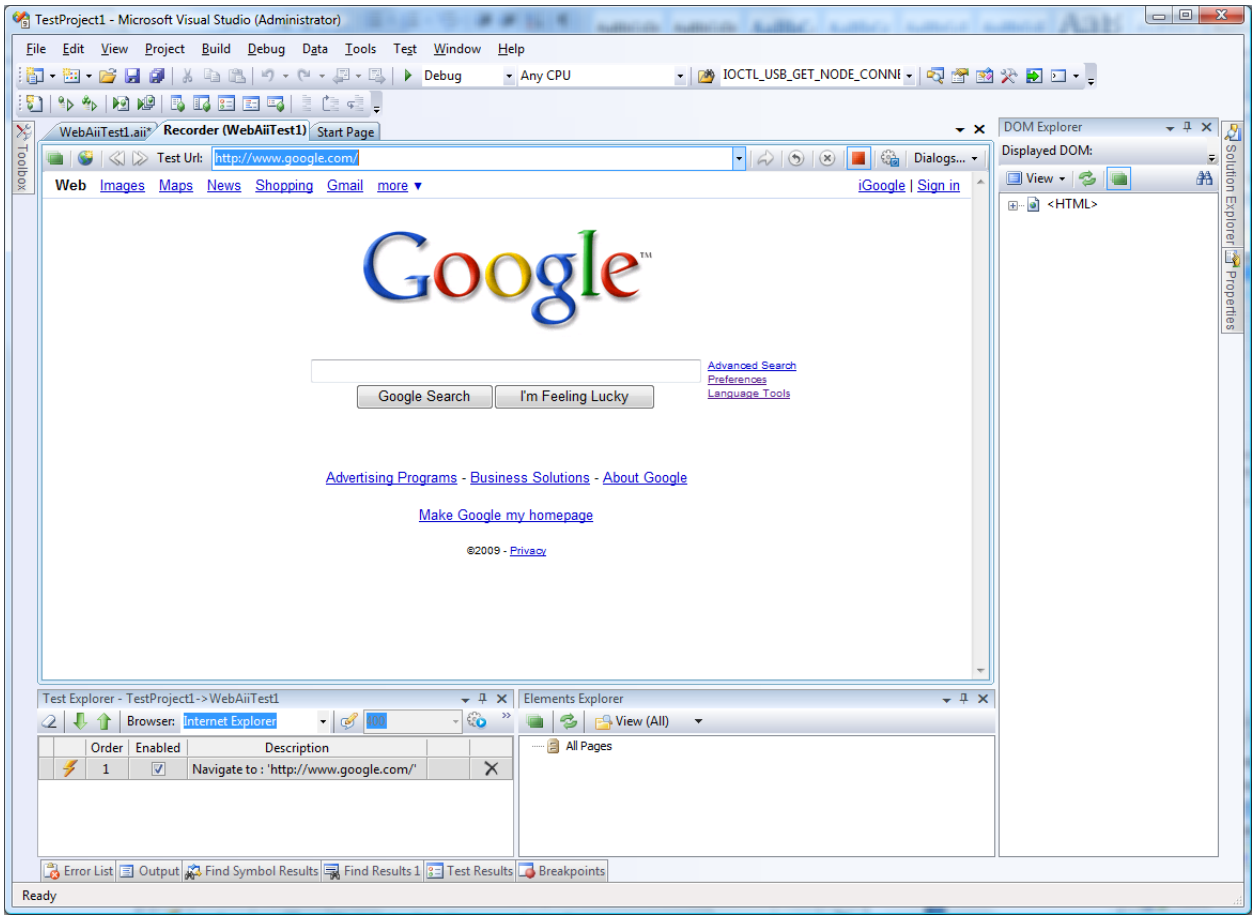


5) If this is your first time to load a WebAii Test please see the section “Recommended Tool Window Layout”.
6) To begin recording the steps of your test, dick on the “Record” button (circled in blue above).
7) The “Recorder” tab will automatically activate.

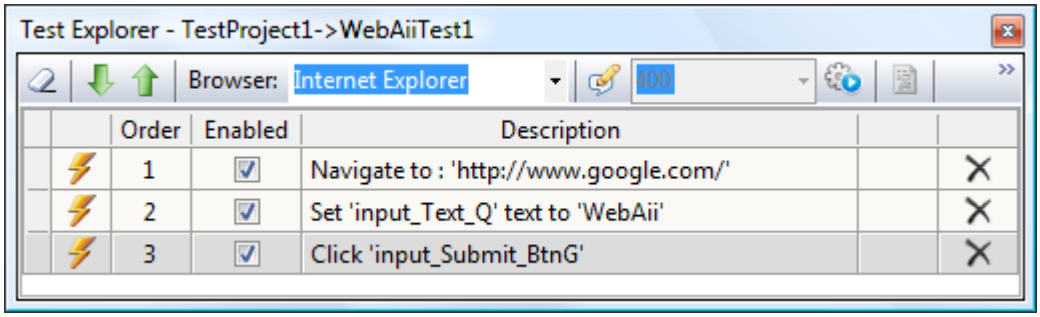


8) In the “Test Url” box, enter www.google.com.

9) Click the “Navigate to URL” button.

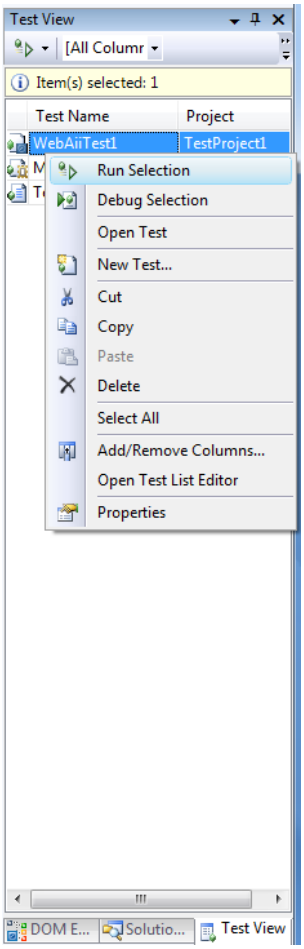


10) You will notice a recording step has been added to the “Test Explorer” tool window.
11) Type in “WebAii” in the Google search box and click the search button.
12) You will notice that two more steps have been added to the “Test Explorer”.

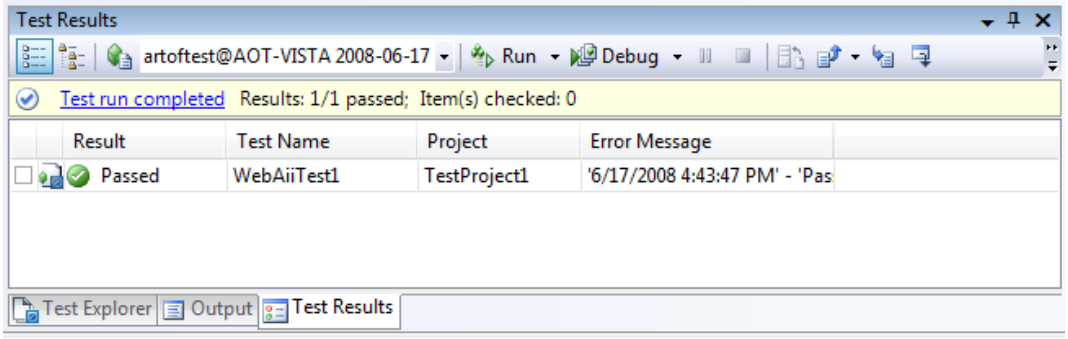


13) That’s how easy it is to create your first WebAii test!
14) Save and build your project.
15) To execute your test, open the “Test View” tool window. This can be found by clicking on “Test” on the top menu then -> “Windows” -> “Test View”.

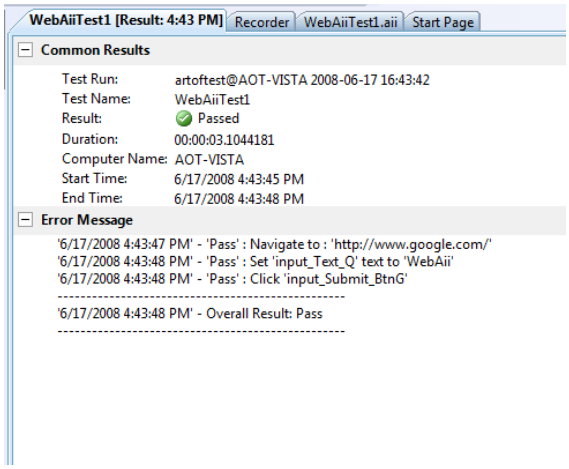
16) Right click on your WebAii Test and click “Run Selection”.



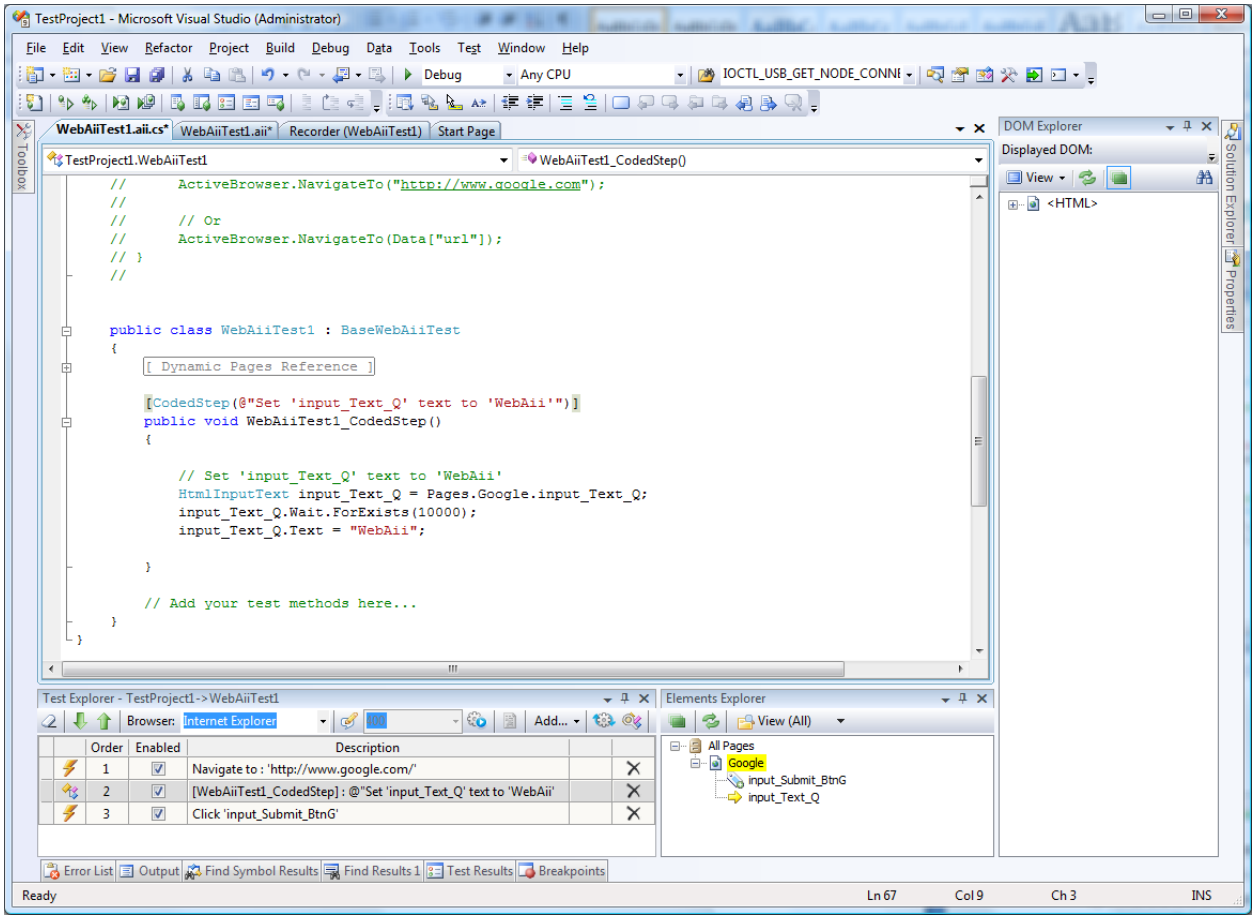
17) An Internet Explorer window will open and the steps of your test will automatically execute in that browser window. When the test completes the browser window will close.



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



Creating a WebAii Test with a code behind file

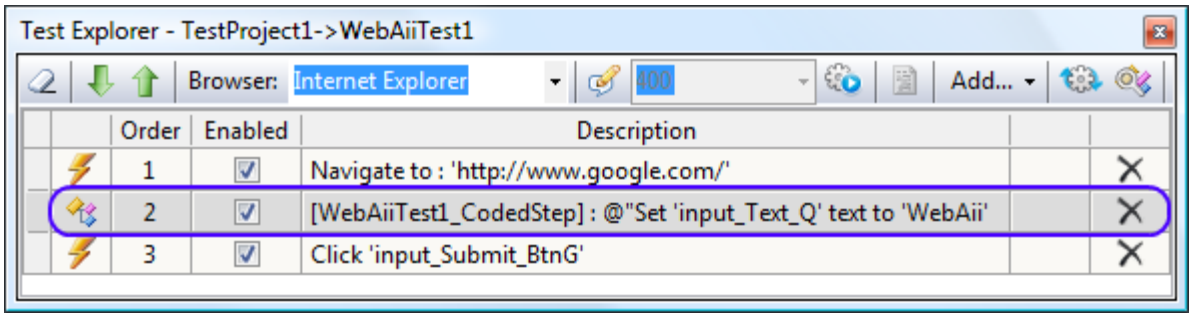


1) Create a WebAii Test as outlined above.
2) Record a few steps as outlined above.

3) There are two ways of creating a code behind file for your WebAii Test file.
   a. On the WebAii Test tab, click the “Add code behind…” button.

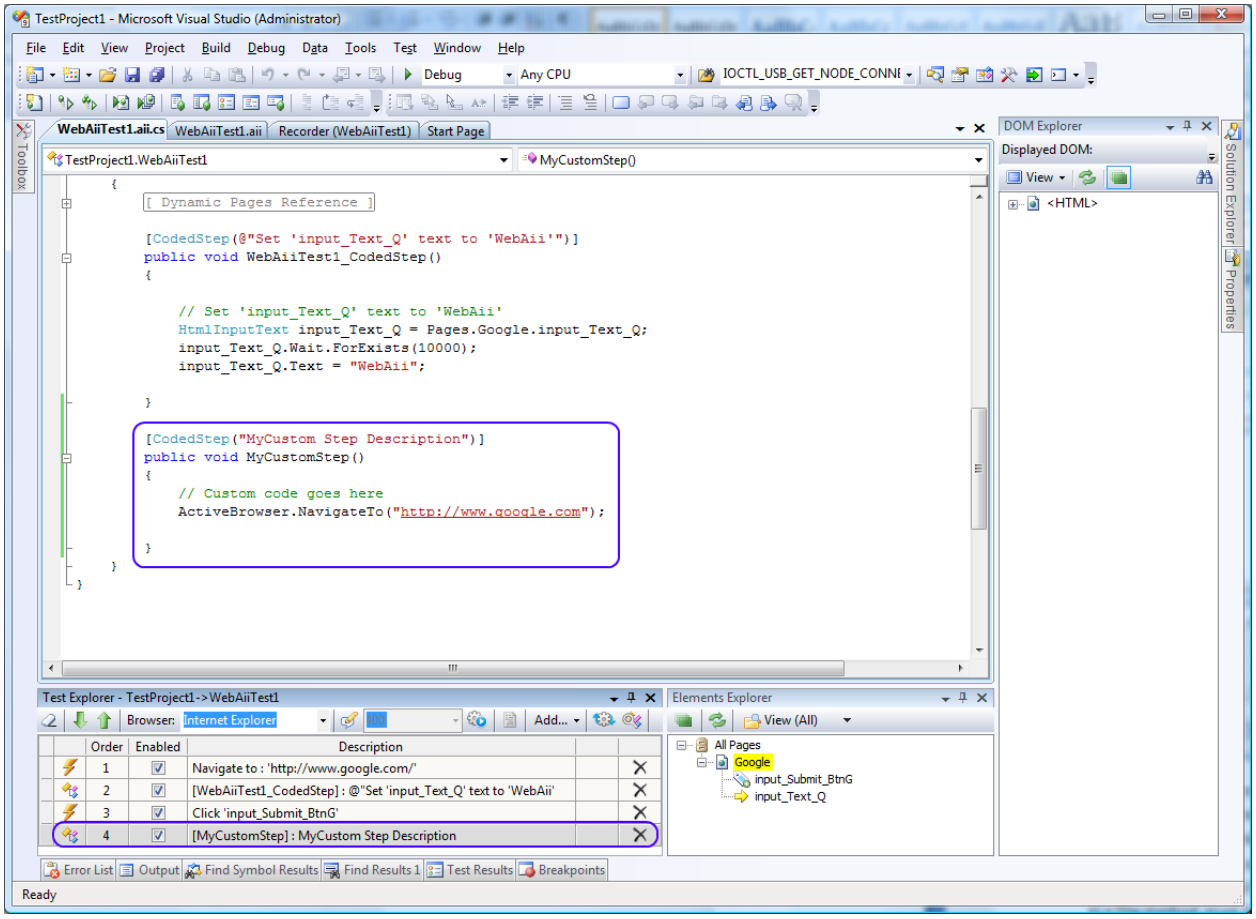
   - Or -

   b. Right click on any step in the “Test Explorer” and select “Convert To 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 “Test Explorer” tool window. Only the description can be changed in the “**CodedStep**” method attribute that appears in the code behind file. A coded step will be indicated with a code icon and will have a white background in the “Test Explorer” tool window.



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 WebAii Test with custom coded steps



1) Create a WebAii Test and add a code behind file using method A as outlined above.
2) Similar to a step that has been converted to code, a method in the code behind file can be represented as a step in the "Test Explorer" tool window.
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 a "`void`" return type or is a "**Sub**" in VB.NET.
4) The method must be decorated with the "CodedStep" attribute as shown below:

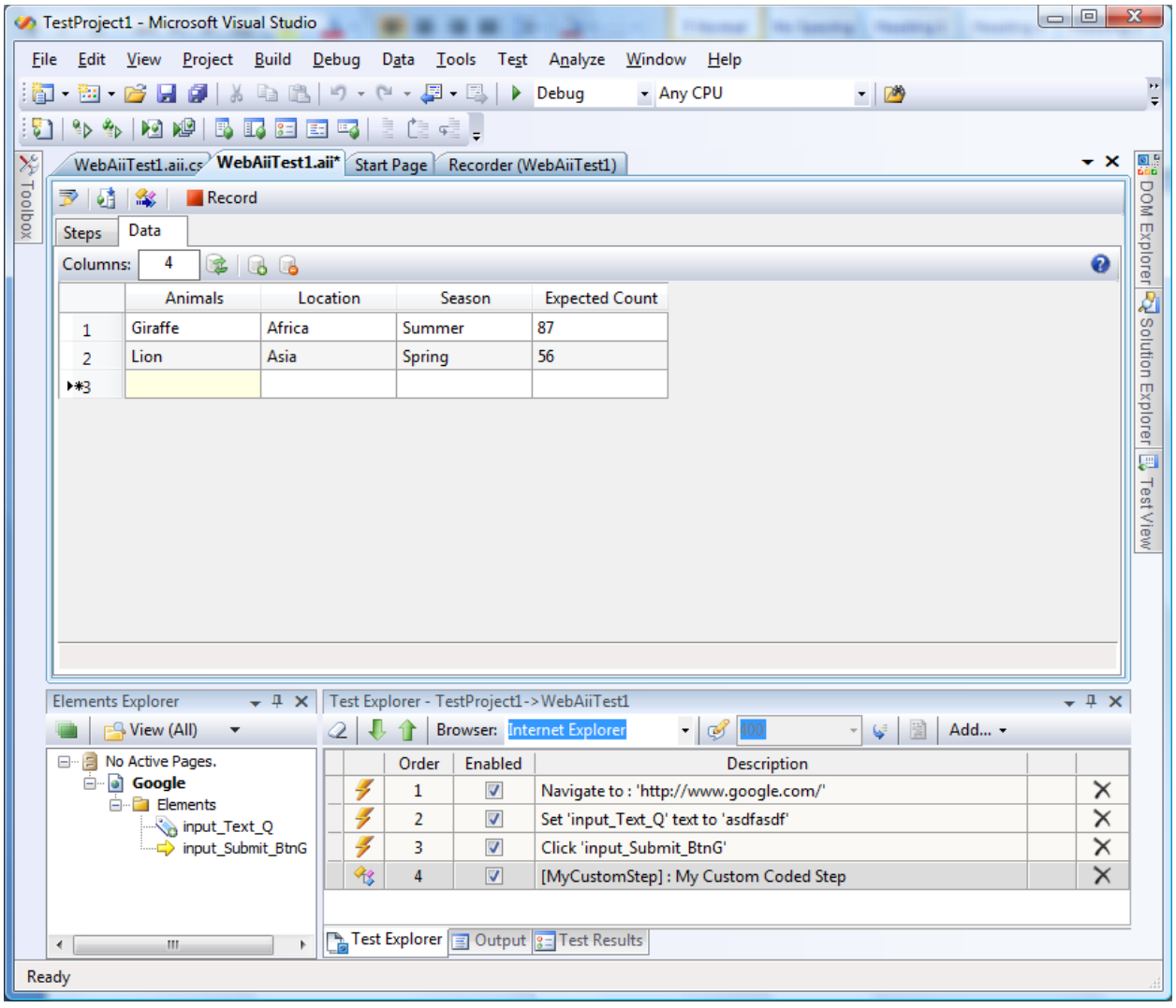
```
[CodedStep("MyCustom Step Description")]
public void MyCustomStep()
{
    // Custom code goes here
    ActiveBrowser.NavigateTo("http://www.google.com");
}
```

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 to the "Test Explorer" with the coded icon immediately upon saving. NOTE: Newly coded steps are always added to the end of the steps in the "Test

Explorer" tool window. You can use the move up/down buttons on the "Test Explorer" tool window to change its order of execution or drag and drop the step to the desired position.

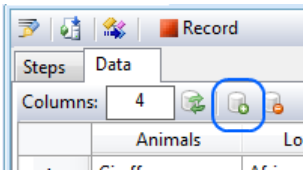
7) The name of the method becomes the step name and the description from the Coded Step attribute is listed as the description in the "Test Explorer" tool window.
8) Save and build the project.
9) Execute the test.
10) The code behind file has access to all the WebAii Framework runtime objects like ActiveBrowser,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

Creating a Data Driven WebAii Test

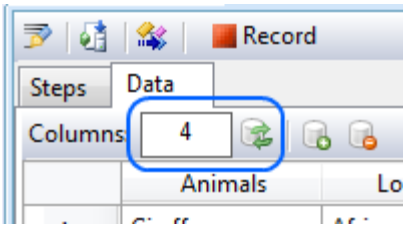


1) Create a new WebAii Test as outlined above.
2) Record the Google search steps as outlined above.
3) Click on the WebAii Test tab.

4) There are two tabs at the top of this tab, one labeled “Steps” and the other labeled “Data”
5) Click on the “Data” tab.
6) 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.



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



9) You now have a grid of just one column.
10) Enter any text into the first grid cell and hit enter or tab. The input will move to the second grid cell.
11) Continue entering text for 4 more grid cells. New rows will automatically be added as you type.
12) Save your test.
13) 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 none coded step, see next section.
14) Convert the step to code that sets the value of the Google search text box from the “Test Explorer”.
15) To reference the data from your grid use the “Data” property followed by the index of the column. For example:

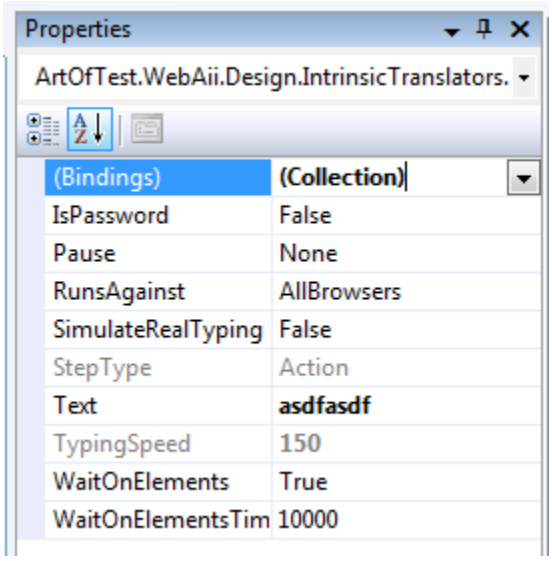
```
[CodedStep("Set 'input_q' text to 'WebAii'")]
public void WebAiiTest1_CodedStep1()
{
    // Set 'input_q' text to 'WebAii'
    HtmlInputText input_Text_Q = Pages.Google.input_Text_Q;
    input_Text_Q.Wait.ForExists(10000);
    // You can reference the column by index
    input_q.Text = Data[0];
    // Or by name
    input_q.Text = Data["Col1"];
}
```

16) Save and build your project.
17) Execute your test. Note that the test will rerun for each row in the data array.

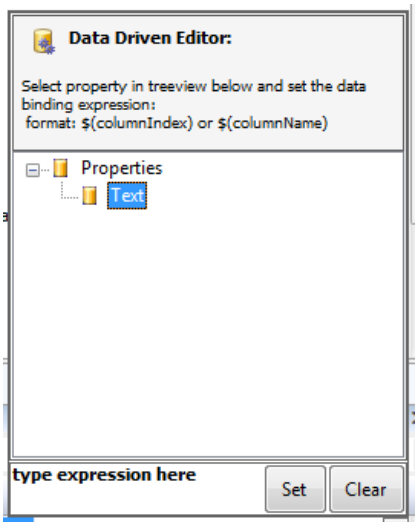
Binding data to a recorded step

To bind data to a recorded step, follow these steps.

1) Open the properties window by pressing the F4 key.
2) Click on the recorded step that sets the value of the Google search text box from the "Test Explorer".
3) The properties for this step will appear in the properties window.

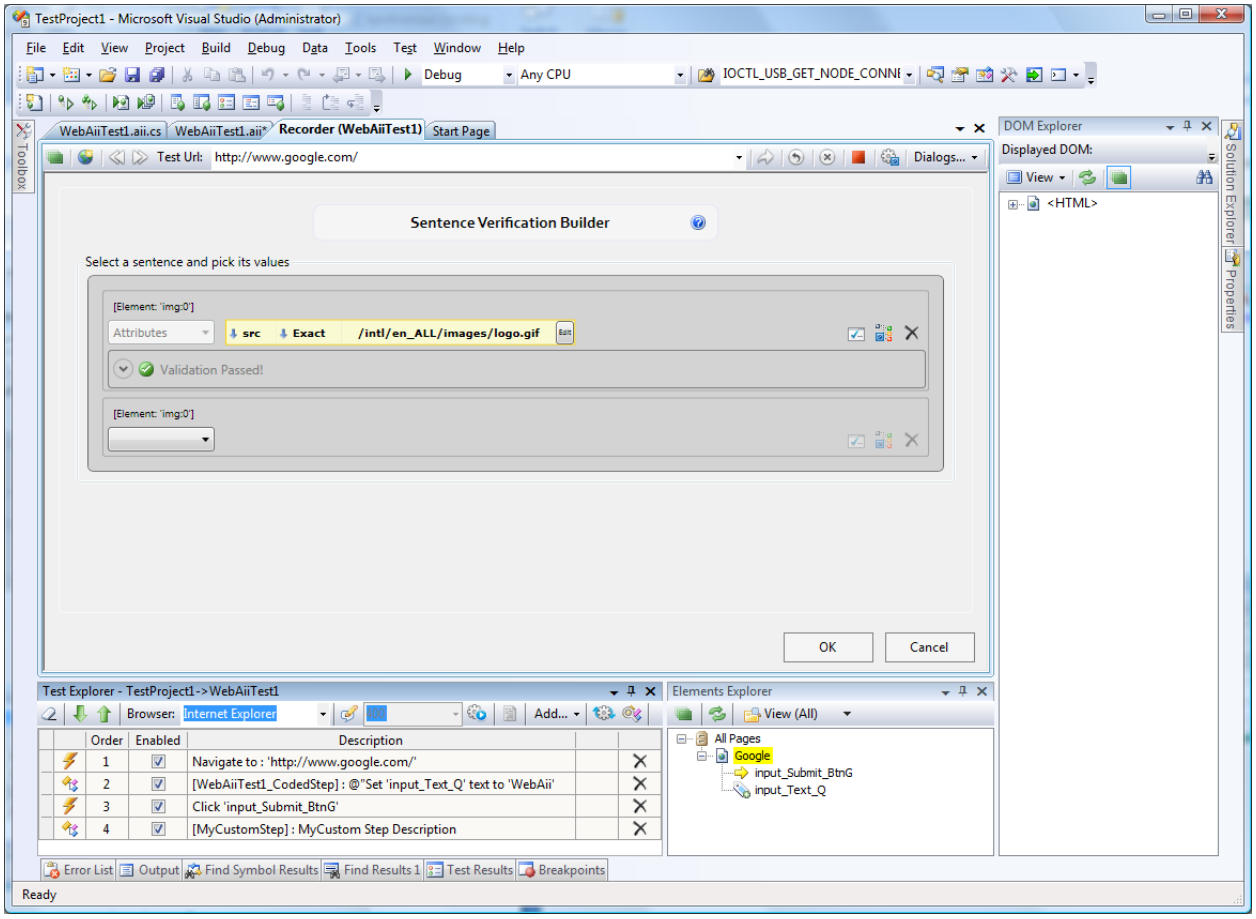


4) Click on the drop down arrow for (Bindings).
5) Click on the "Text" node in the displayed tree.
6) Enter $(Col1) into the text box.
7) Click the Set button

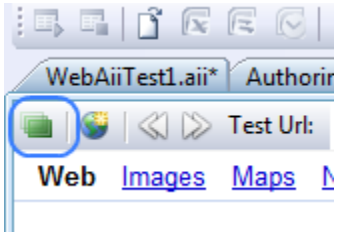


8) The data for the column named "Col1" from the data array is now bound to the Text property for that step. Instead of entering the text "WebAii" into the search box, the data stored in the data array will be entered instead.
9) Save and build your project.
10) Execute your test. Note that the test will rerun for each row in the data array.

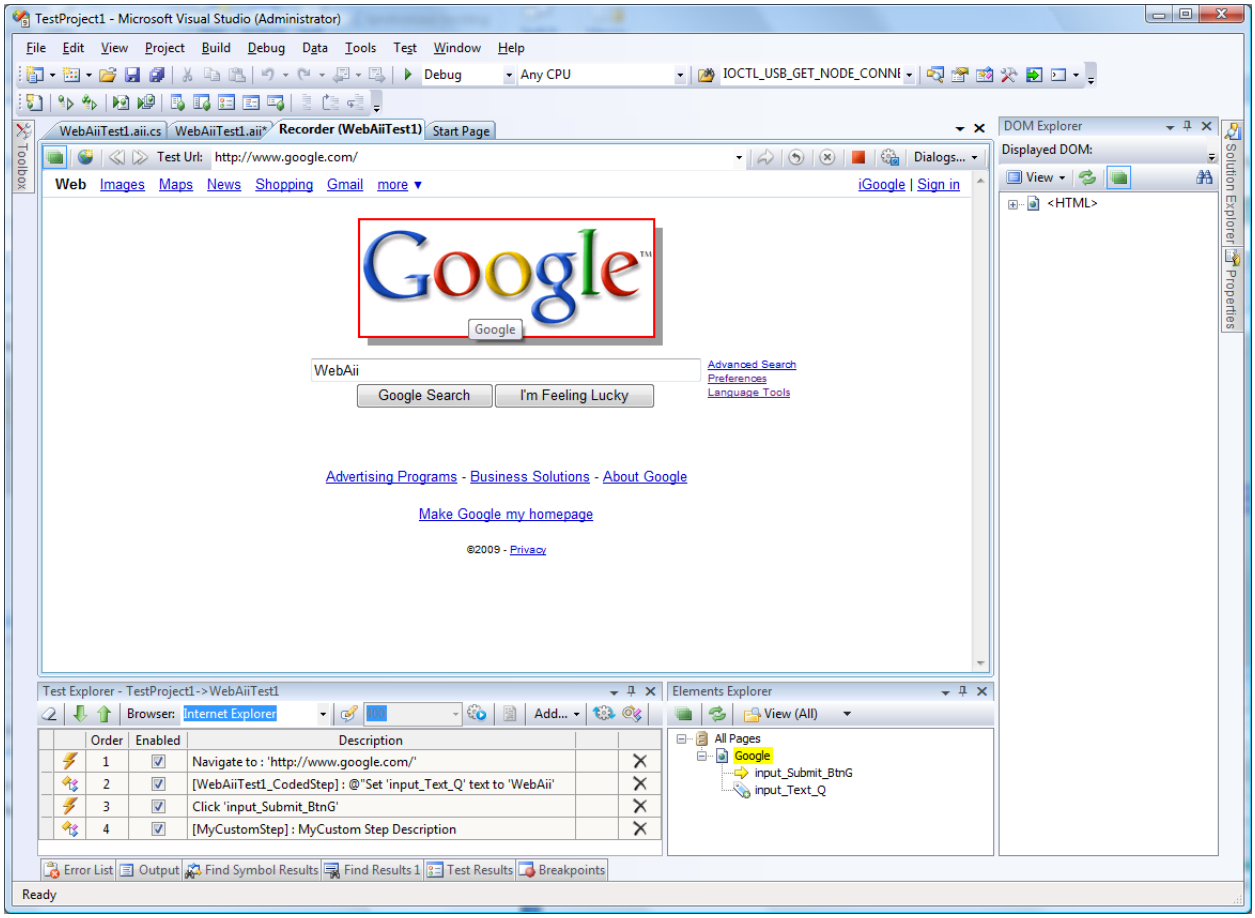
Creating WebAii Test Verifications Overview



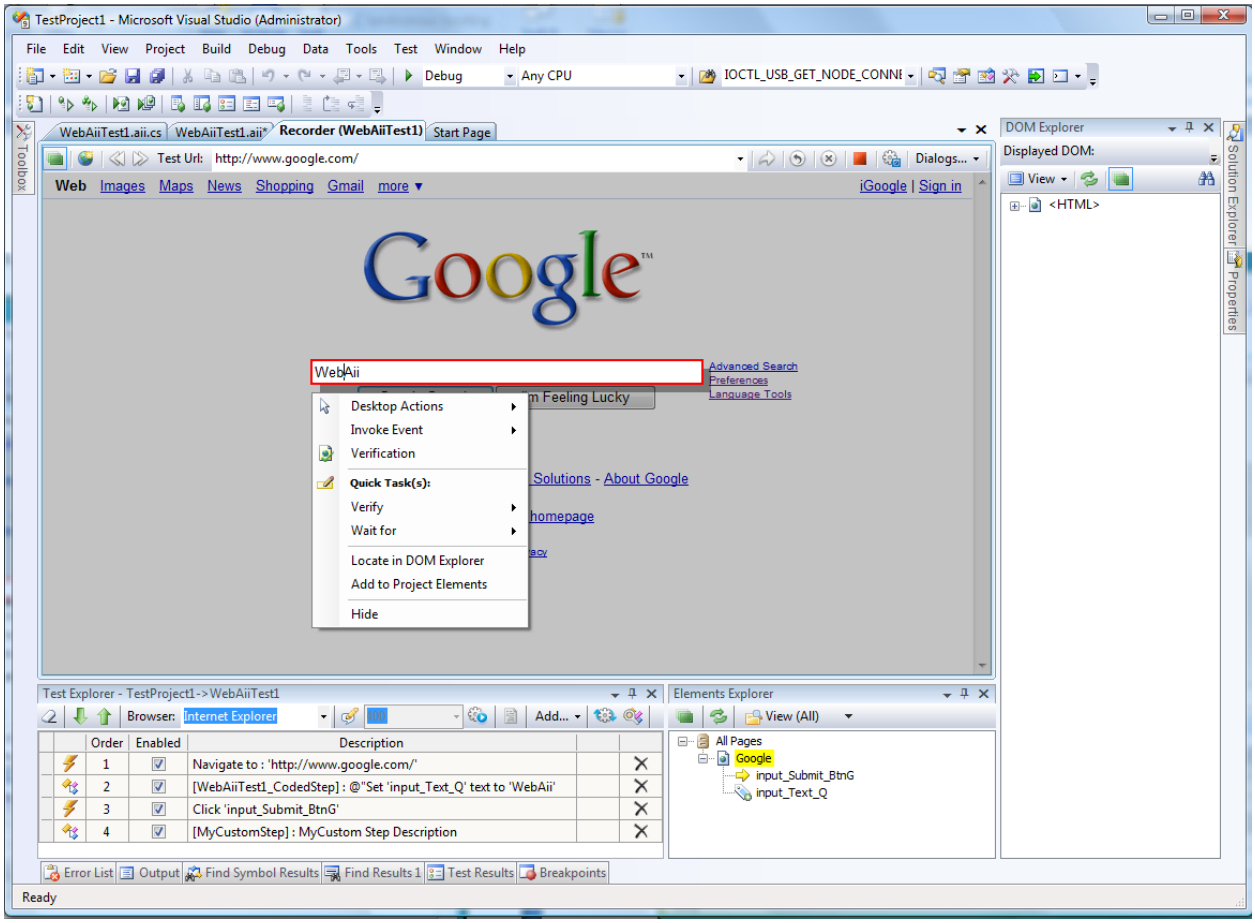
1) Test Verifications are added to a WebAii tests as an ordered step.
2) To craft a verification step, enable the Automation Overlay Surface by clicking on the “Enable overlay” button.



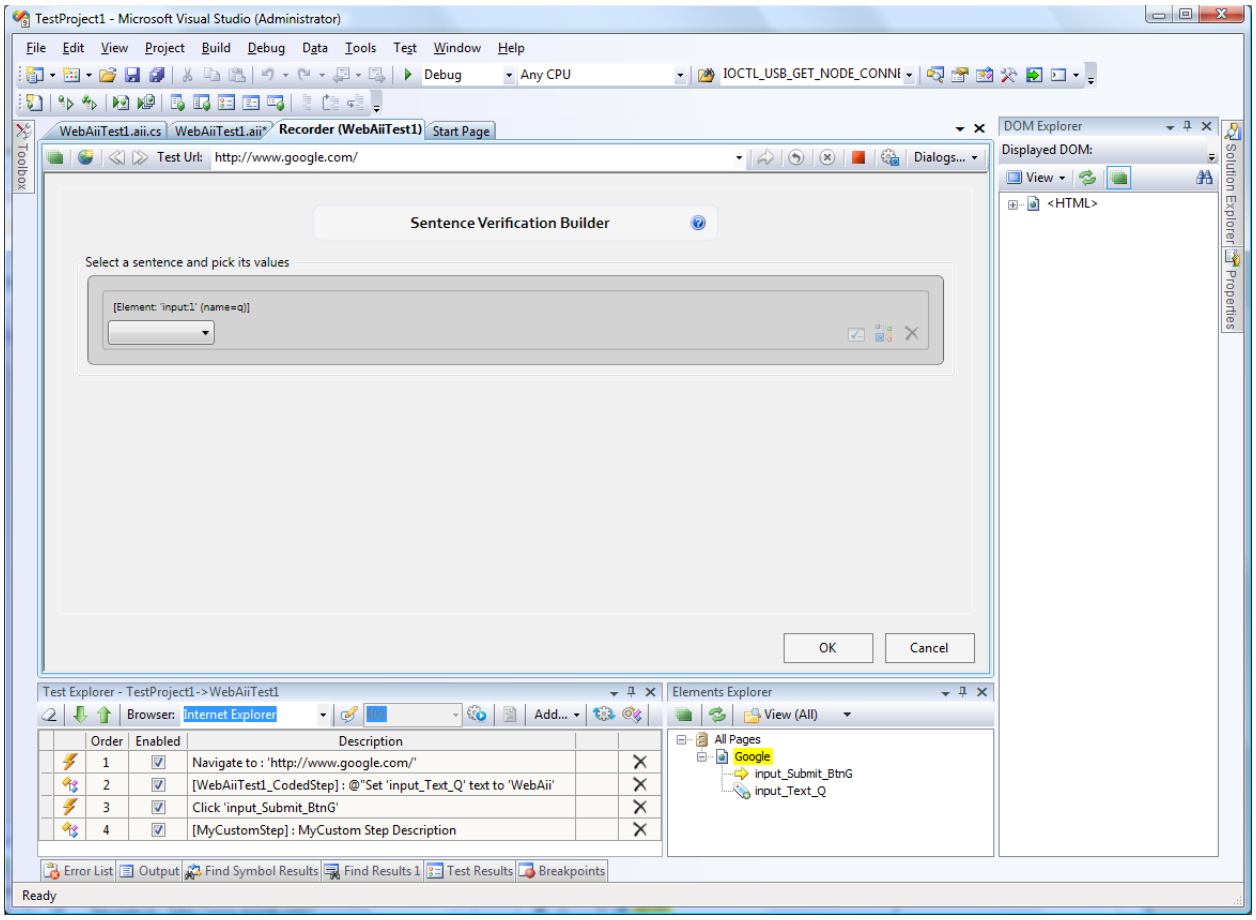
3) You will know if the Automation Overlay Surface is on by hovering over the web page and seeing each element being highlighted.



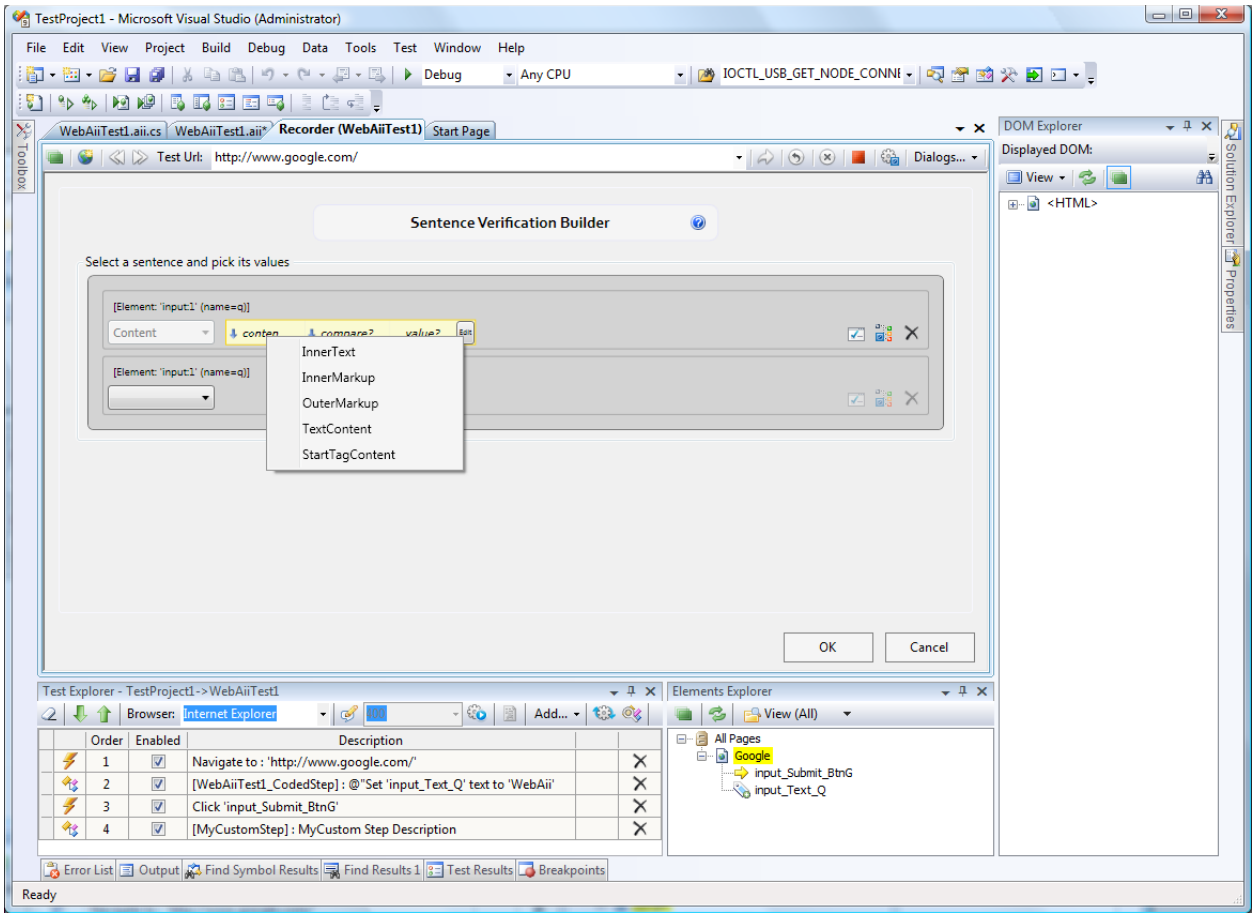
4) Select an Element that you want to craft a verification for and right click on it. This will open a context menu.



5) From the context menu, select “Verification” to open the Verification Builder.

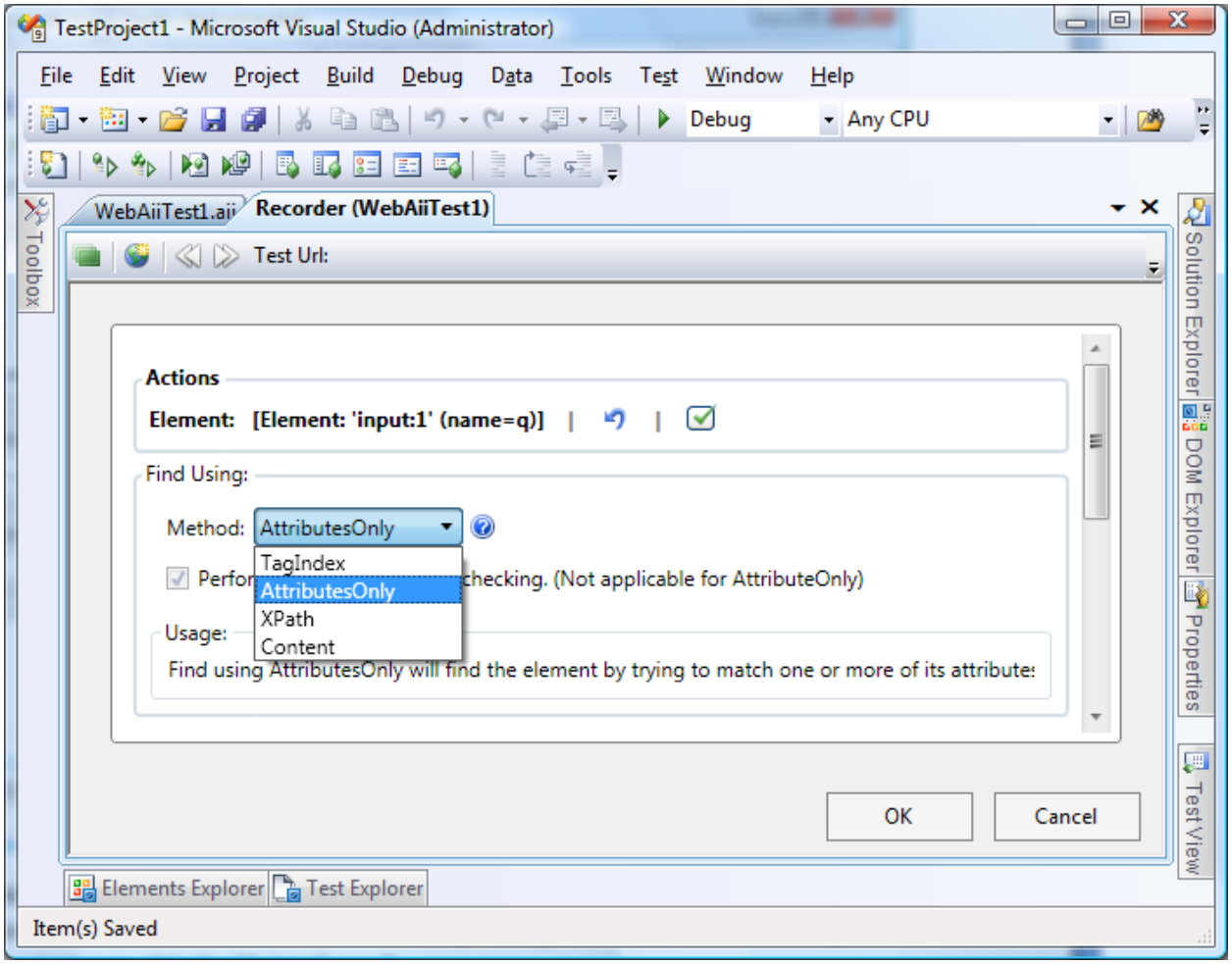


6) The verification builder offers sentence based verifications of elements.
7) Start by selecting a rule category from the drop down box.
8) During verification crafting, the content is dynamically built against the currently selected element. As you make choices, default values will be populated according 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 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 to the “Test Explorer” tool window.
14) When you finish building your verification, select OK to add it as a step to your current test.

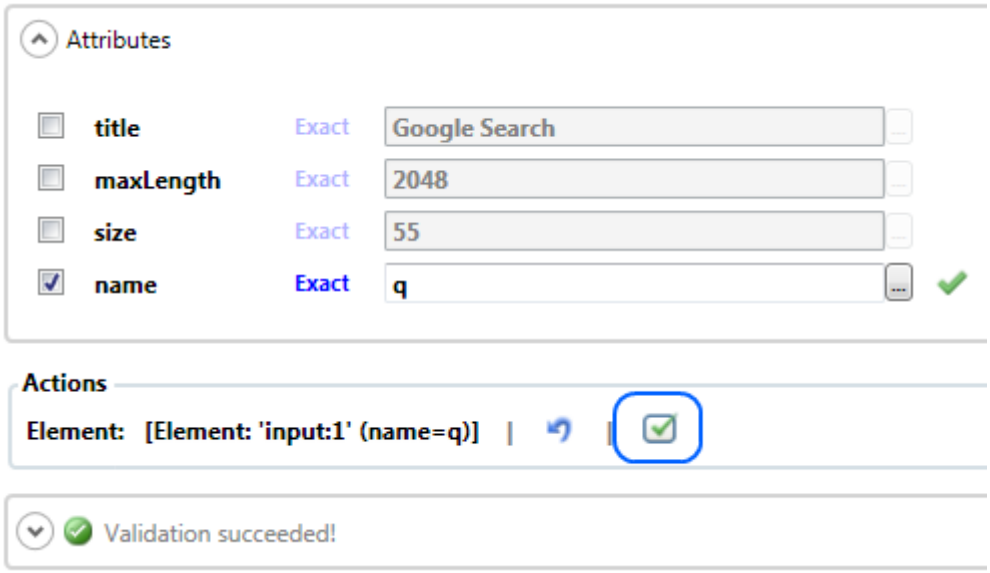
Changing how an Element is found



1) Whenever a web page element has an action recorded against it or you explicitly add an element to the Elements Explorer, a "Find Param" is generated that tells the framework how to find that element.
2) In order to change how an element is found, right click on the element in the Element Explorer tool window and select "Edit…".
3) After clicking "Edit Element", the "Find Param" Editor will launch and will display how the current element is being found.
4) Inside the "Find Using" box there is a drop down box entitled "Method". This drop down contains:
    a. TagIndex – TagIndex will find the element with the specified tag name at the specified index within the DOM.
    b. AttributesOnly – AttributesOnly will find the element by trying to match one or more of the element's attributes.
    c. XPath – XPath will find the element using the specified XML XPath specified.

d. Content – Content can be used to find an element using its TextContent, InnertText… etc.

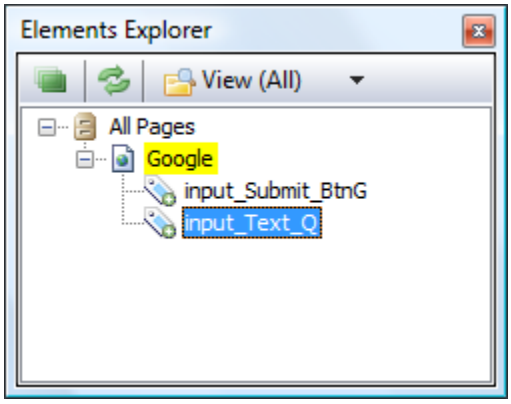
5) After specifying the method and criteria to use for the element find, you can click on the "Validate" icon inside the "Actions" group box to validate that the current FindParam finds the element correctly within the current DOM.



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

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

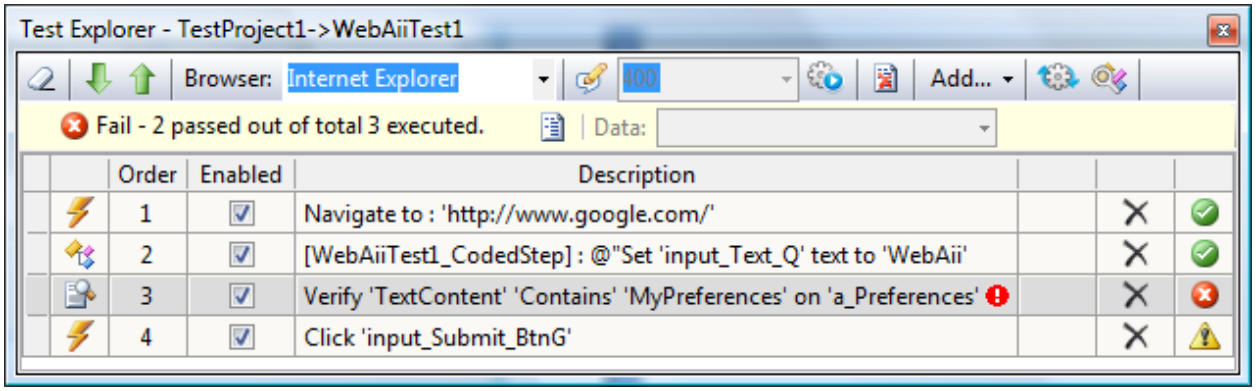
1) Pages can be referenced in the code behind file in the same way they are found hierarchally in the Elements Explorer.



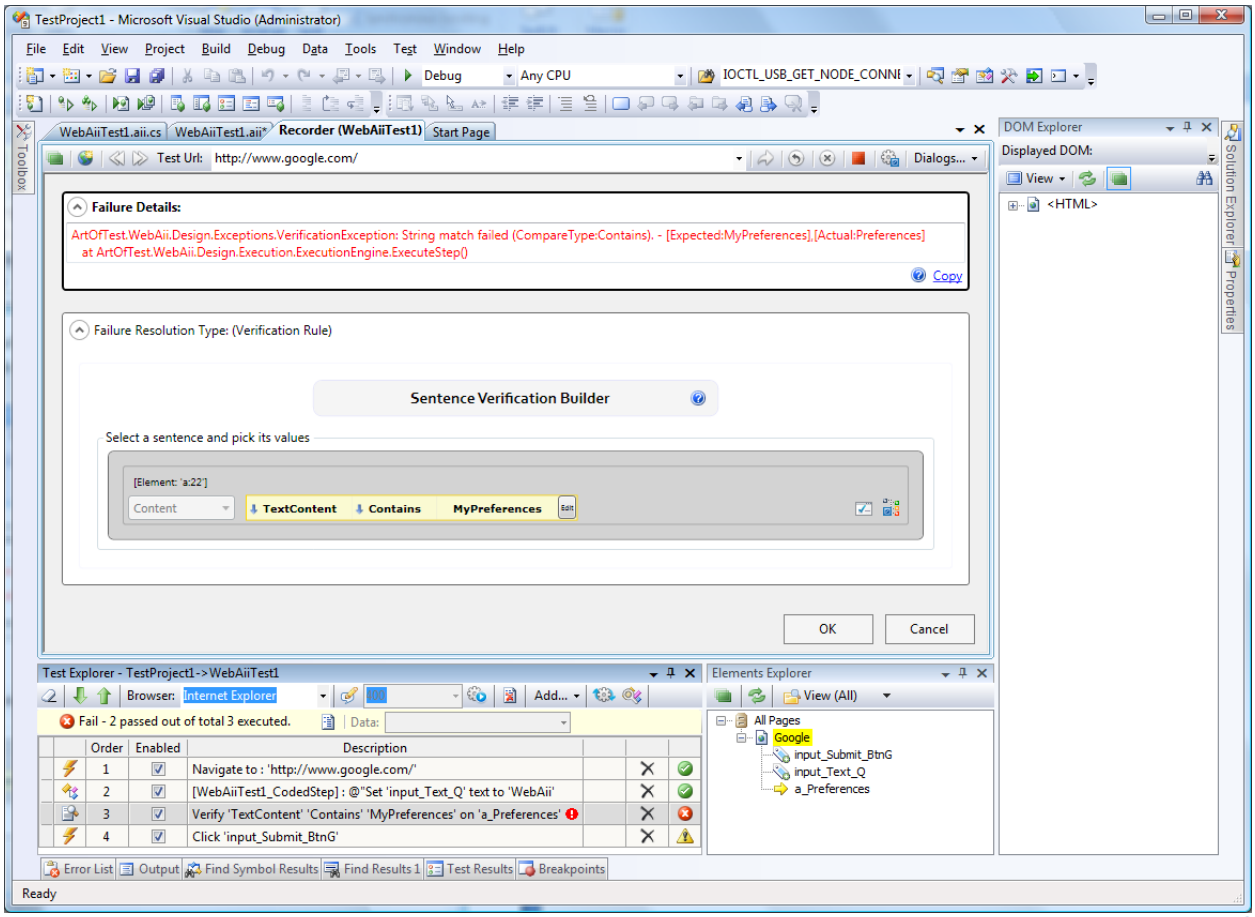
2) For example: If the image above was your current Elements Explorer window, you could reference the "input_Text_Q" element like this:
   a. `Pages.Google.input_Text_Q`
3) This will return the strongly typed WebAii Framework "HtmlInputText" type which is derived from the "Control" type.

How to resolve test step failures

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



2) Clicking the Red X icon will launch the failure debug UI.
3) The failure debug UI will give the textual failure description as well as details about the failed step.

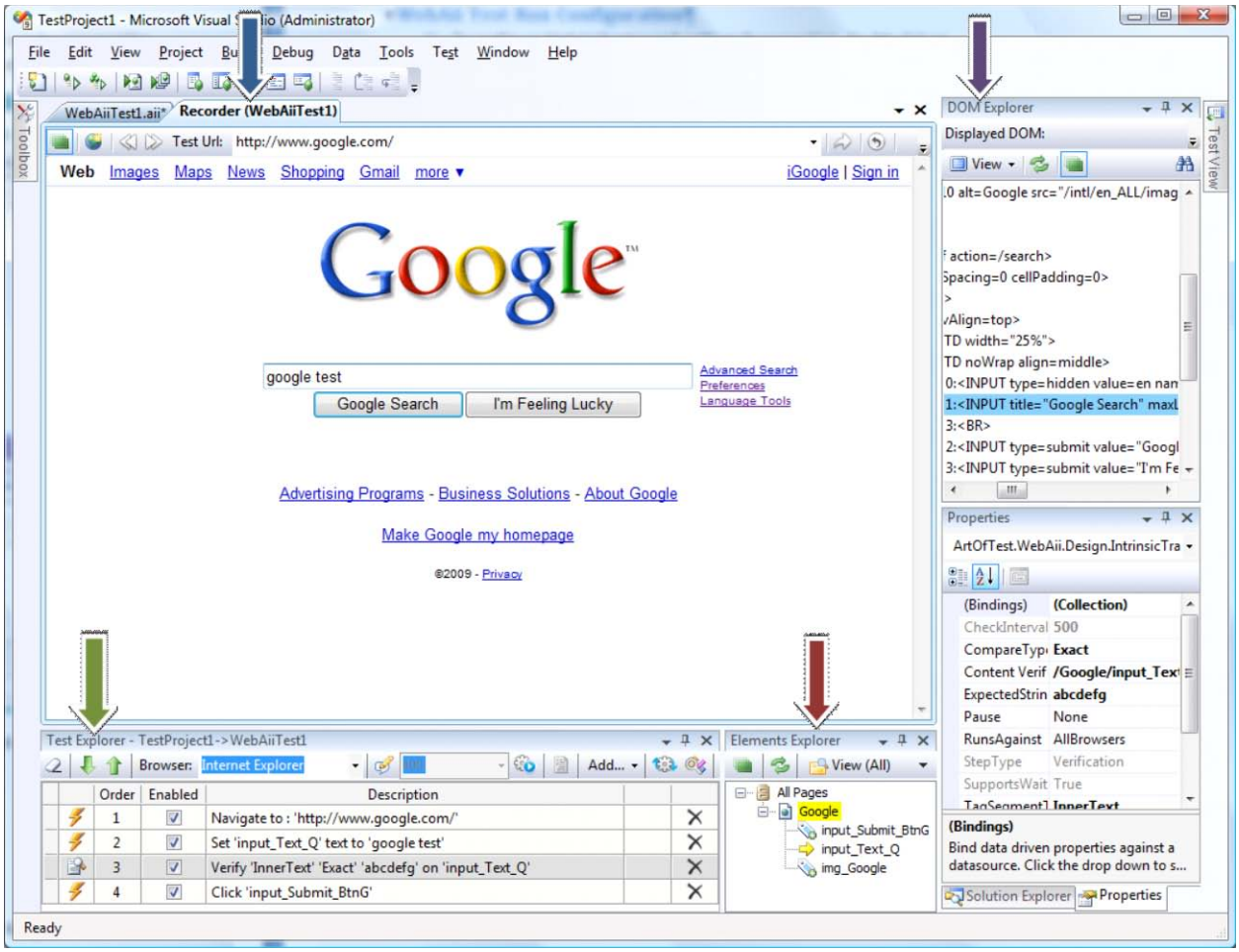


4) The major feature of the failure debug UI is that it loads the "Execution DOM" in the "DOM Explorer" tool window. This allows the tester to compare what the expected result was against the actual result when the test was executed.

5) At this point the tester needs to decide if the test needs updating or the page under test has a defect. If it is the test, then you can simply modify the step and click OK to update the test. Otherwise, the test has detected a product defect.

Automation Design Canvas Recommend Tool Window Layout

The first time you load Automation Design Canvas, all tool windows except for the recording surface might be floating. Below is a recommended layout.



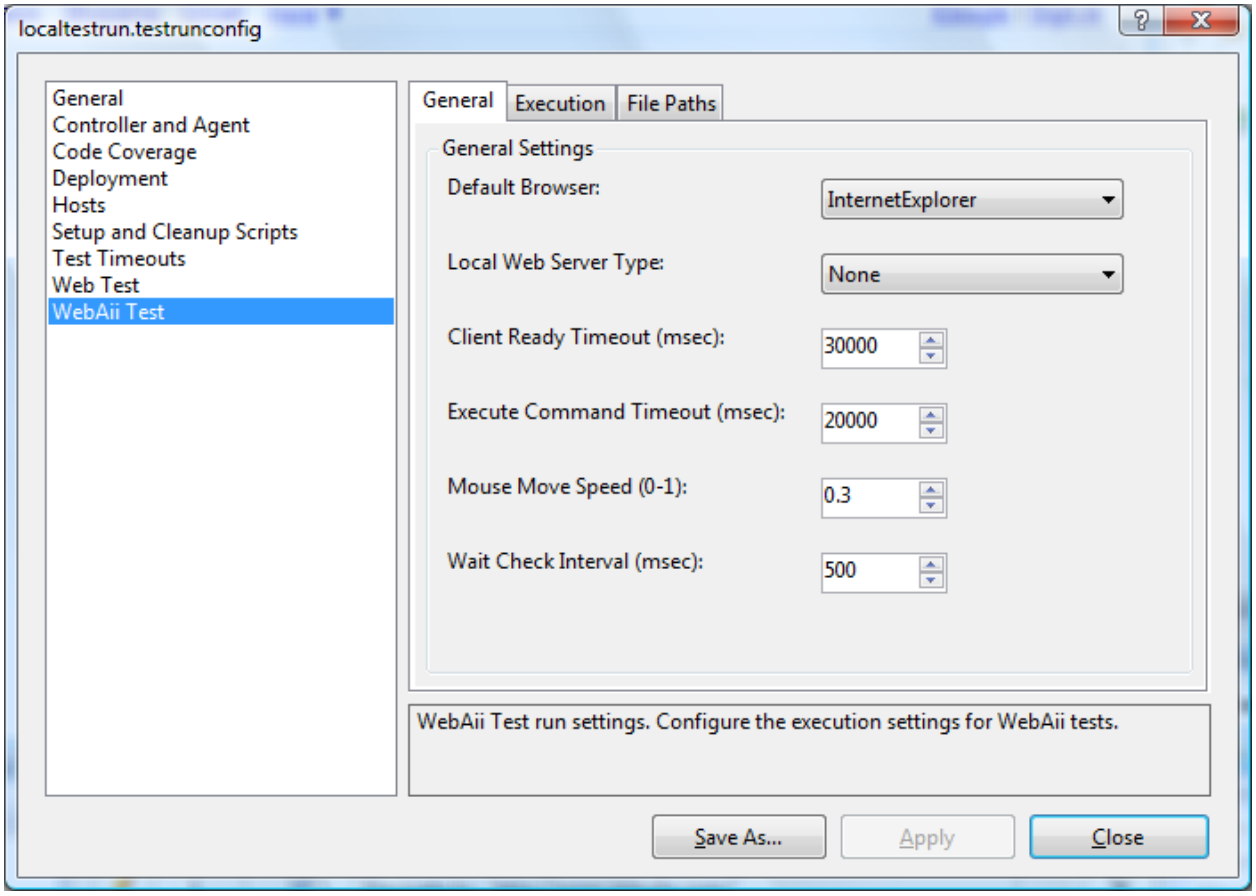
1) The layout above is the recommended tool window layout to optimize space and recording efficiency.
2) The Automation Design Canvas Tool Windows are:
    a. Recording Surface (Document Window Docked)
    b. Elements Explorer (Dock-able floating tool window)
    c. Test Explorer (Dock-able floating tool window)
    d. DOM Explorer (Dock-able floating window)

Note: This is only a recommended layout; you should customize your layout to your liking.

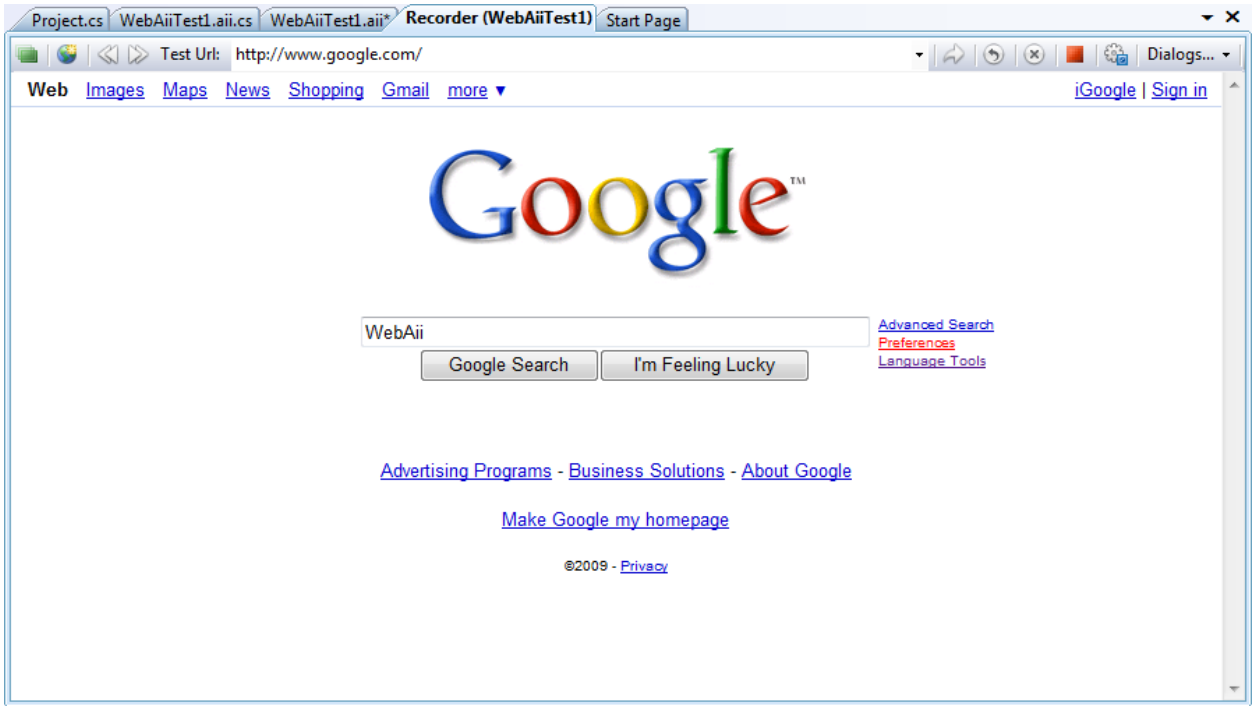
WebAii Test Run Configuration

1) To configure WebAii Framework settings for execution, 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 configuration settings for all WebAii Test executions within this project.



Recording Surface Tool Window Overview

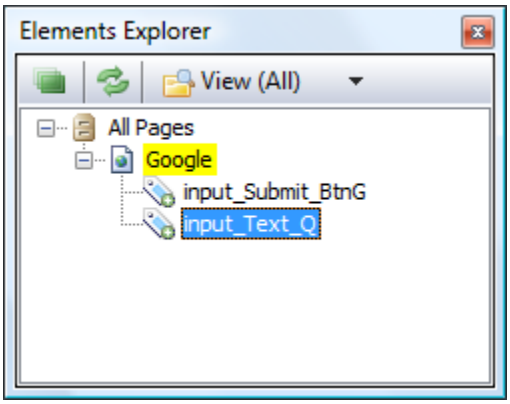


1) The "Recording Surface" tool window contains a web browser and the Automation Overlay Surface.
2) Most actions performed against a webpage can be recorded by just performing the action directly.
3) The menu bar for the Recording Surface contains (from left to right)
    a. Enable / Disable the Automation Overlay Surface
    b. Launch page in Internet Explorer for recording
    c. Navigate Back
    d. Navigate Forward
    e. Test Url
    f. Navigate To
    g. Refresh
    h. Stop navigation
    i. Enable / Disable Recording (Enable the capturing of actions on page)
    j. User Settings (configure settings for Automation Design Canvas)
    k. Dialogs drop down
4) The Automation Overlay Surface (when enabled) provides the user with a visual effect indicating the element you are hovering over.
5) In order to perform more specific action or to craft verifications against an Element, right click a highlighted element to freeze the screen and activate the context menu with the following options:

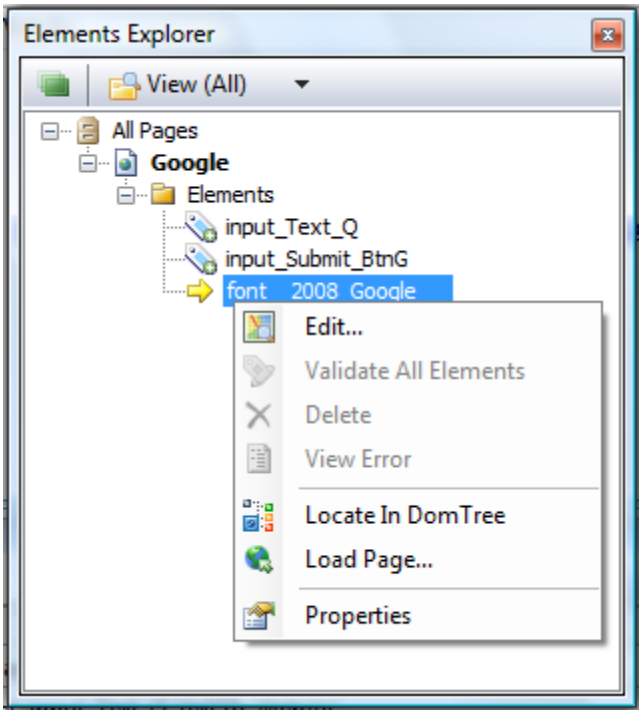
a. Desktop Actions – has options to record specific desktop actions against the highlighted element.
b. Invoke Event – Invoke a specific JavaScript event against the highlighted element.
c. Verification – Enable the verification creation UI.
d. "Quick Tasks" – Quickly record verification or wait for specific to the element type.
e. Locate In DOM Explorer – Highlights the current element in the DOM Explorer.
f. Add to Project Elements – Adds the current element to the Elements Explorer.
g. Hide – Hides the context menu.

Note: The Overlay Automation Surface allows users to record steps that they may not be able to record by simply interacting with the page e.g. mouse hover over.

Element Explorer Tool Window Overview

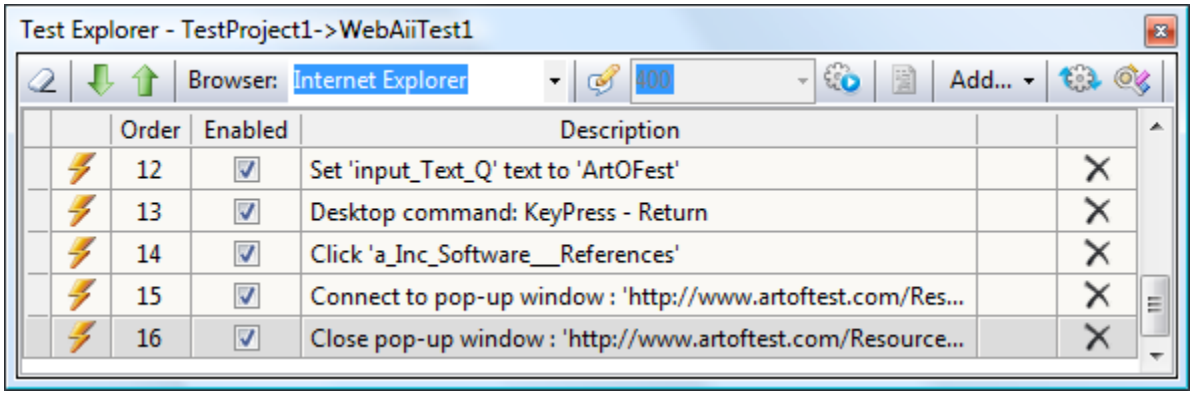


1) The "Elements Explorer" tool window maintains a list of all Elements within the current project.
2) It provides a one stop shop to help maintain elements and the way they are found during execution.
3) The tree view is organized by Page -> Frame -> WebAii Test Regions -> Elements
4) The hierarchy 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 "Elements" folder.
5) Each Page Node has a context menu with three 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 FindParam settings.
    c. Properties



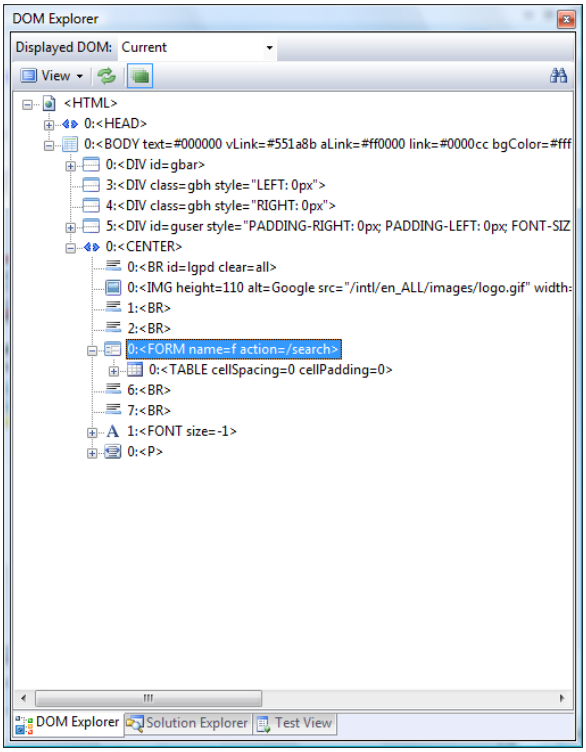
6) Each Element Node has a context menu with four active choices (when the page is loaded in the recording surface):
    a. Edit… allows you to change the FindParam of the Element (the way the element is found).
    b. Locate In DomTree highlights the element node in DOM Explorer.
    c. Load Page… loads the URL into the recording surface window.
    d. Properties
7) The Elements Explorer menu bar has 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.
8) Clicking each element will display the properties for that element in the Properties tool window.

Test Explorer Tool Window Overview



1) The "Test Explorer" tool window provides the list of steps contained within the currently selected WebAii Test.
2) 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.
3) The menu bar for the "Test Explorer" 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. Browser Drop Down – selects which browser to use for quick execution.
    e. 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.
    f. Execution Delay – sets (in milliseconds) the time to delay between each step execution.
    g. Quick Execute – immediately executes the current test in the selected browser.
    h. Clear execution results – clears the previous execution results of the last Quick Execute run.
4) Clicking a step will display the properties for that step in the Properties tool window. You can 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.

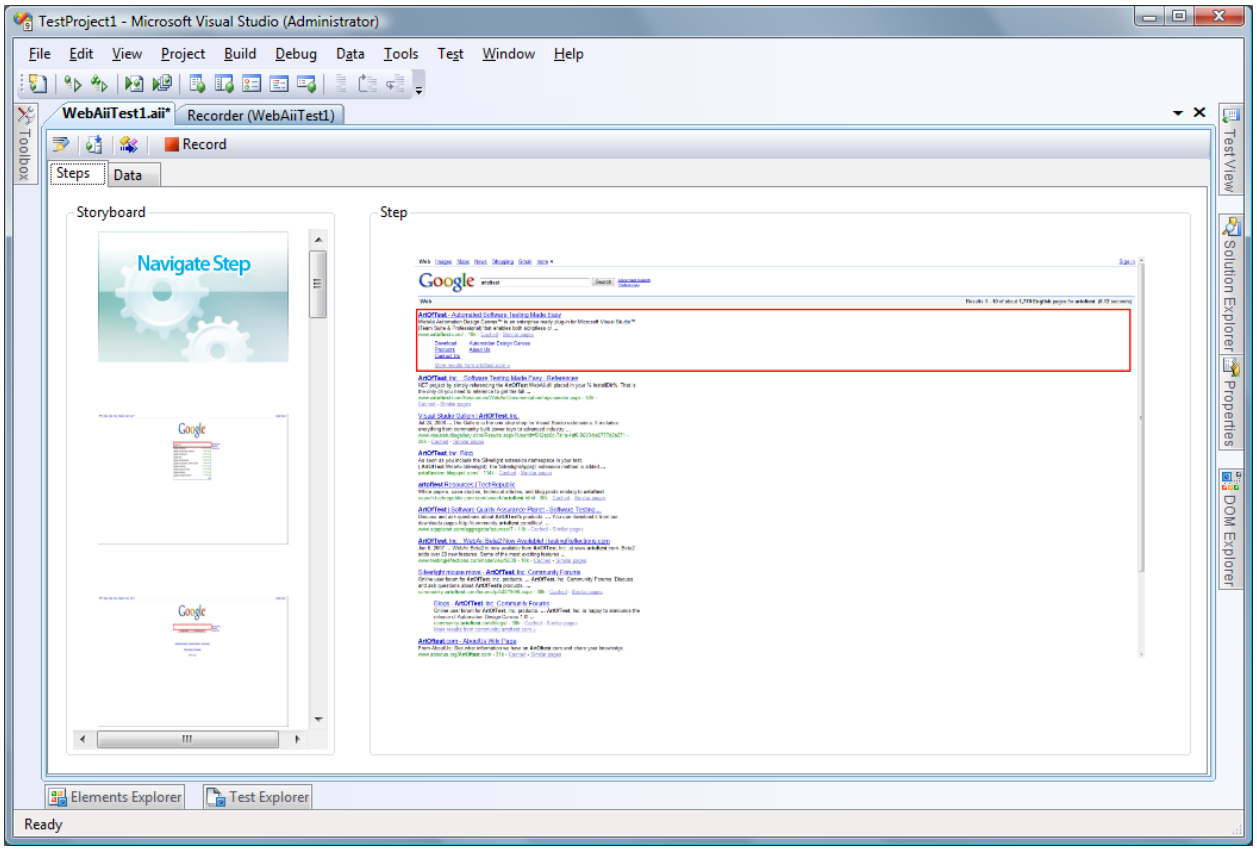
DOM Explorer Tool Window Overview



1) The "DOM Explorer" tool window displays the Document Object Model for the currently loaded page in the "Recording Surface" document window.
2) The nodes are a hierarchical representation of all the HTML elements that make up the page.
3) Each node in the tree is listed by:
    a. Tag Index: <[Opening Tag Name] [attributes]>
4) Clicking on a node in the DOM Explorer will highlight that element (if visible) in the Recording Surface document window.
5) Right clicking a node will bring up a context menu containing two options:
    a. Record Options – will open a context menu in the Recording Surface with recording options.
    b. Add to Project Elements – will add the currently selected element to the "Elements Explorer" tool window and craft a default FindParam for that element.
6) The menu bar for the DOM Explorer contains (from left to right)
    a. View (two options)
        i. Hierarchical View – Displays the DOM in the order of the page.
        ii. By Tag Name – Displays the DOM ordered by tag names.
    b. View Visual DOM State – only enabled during certain operations.
    c. Refresh DOM – will rebuild the tree from the current page.
    d. Search – will enable another menu below to search through the DOM tree.
        i. Drop Down List with search criteria.

ii. Search Operator (exact / partial)
iii. Text box for the search
iv. Find Button to initiate the Search

WebAii Test Tab Overview



1) Each WebAii Test loaded in Visual Studio will have its own document tab allowing for step visualization and test data manipulation.
2) The WebAii Test tab has a menu bar with four icons:
    a. Add a code behind file – Adds a new .cs code behind file which is then attached to the WebAii Test.
    b. Convert test to VS WebTest – Generates a Visual Studio WebTest from the WebAii Test and adds it to the project.
    c. Generate Unit Test from WebAii Test – Generates a Visual Studio Unit Test from the WebAii Test.
    d. Record – Opens the "Recording Surface" and directly enables recording for this test)
3) There are two panels on the "Steps" tab of the WebAii Test:
    a. Storyboard – displays thumbnail for all steps in the test.

b. Step – displays the Recording Surface image for selected step.
   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

4) Clicking on any of the steps in the “Storyboard” group box will display the image for that step.
5) Each recorded image will have the element that it was recorded against highlighted.
6) The WebAii Test tab also contains a “Data” tab.
7) The Data tab allows for the creation of a simple Excel like data array to be used by the steps of the test. See “Creating a Data Driven WebAii Test” section for more details.