Telerik Corp.

Test Studio
Standalone & Visual Studio Plug-In
Quick-Start Guide

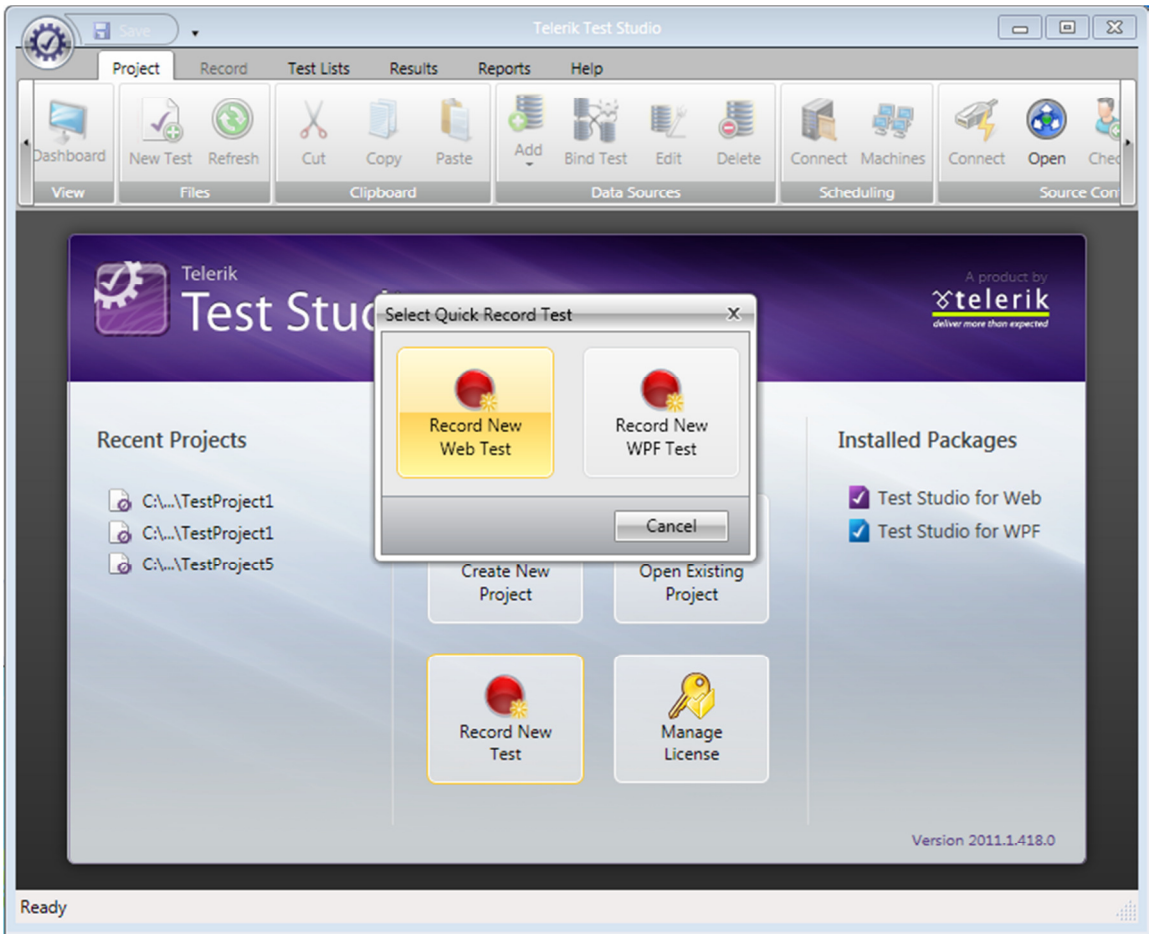
Contents

Create your First Test

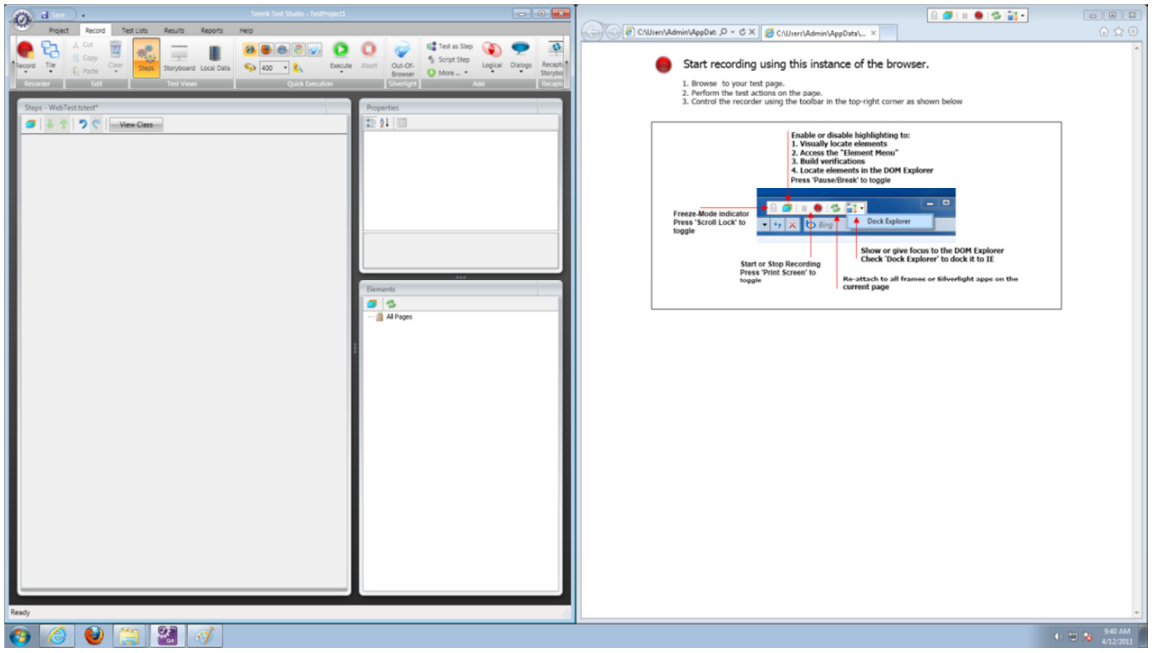
Let's jump right in and show you how easy it is to record test. We'll show you how to record a Web, WPF, and Silverlight test with the Standalone and Visual Studio plugin versions. Ready? Here we go.

Standalone Web Test

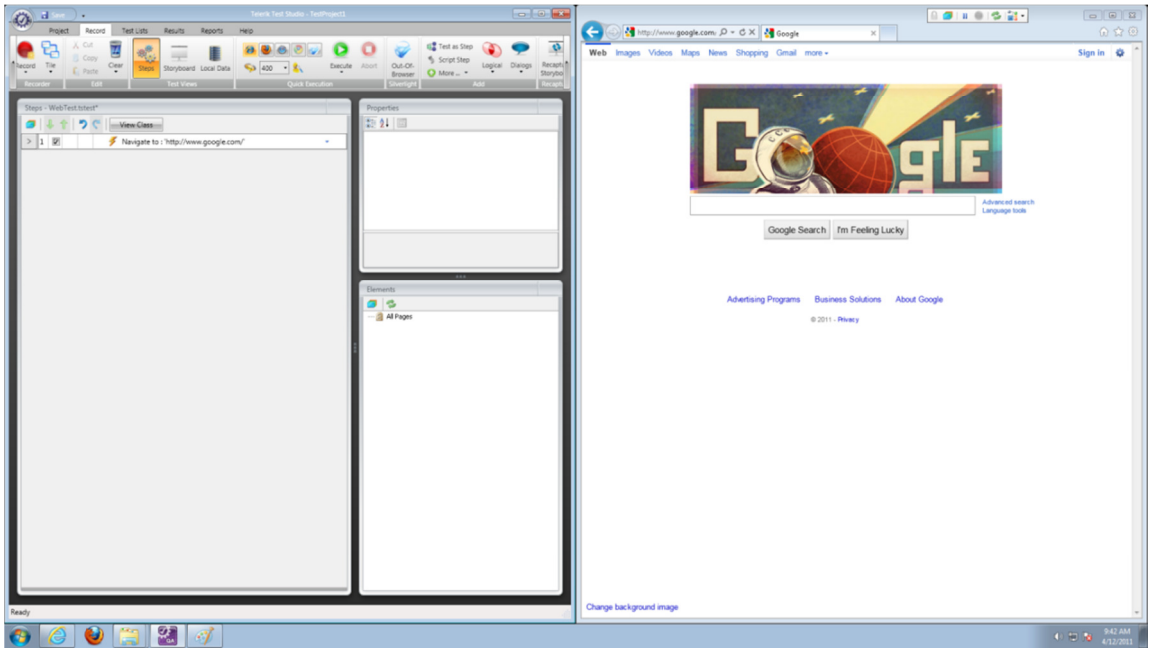
1. Launch Telerik Test Studio.
2. Click "Record New Test" and then "Record New Web Test."



3. Recording is activated and an IE recording window opens.

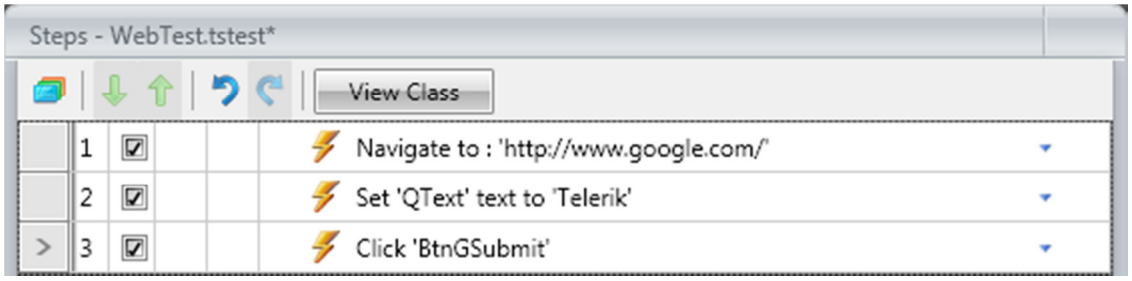


4. Type www.google.com in the URL box and hit Enter.
5. Notice a recording step is added to the Steps pane.



6. Type “Telerik” in the Google search box and click the Search button.

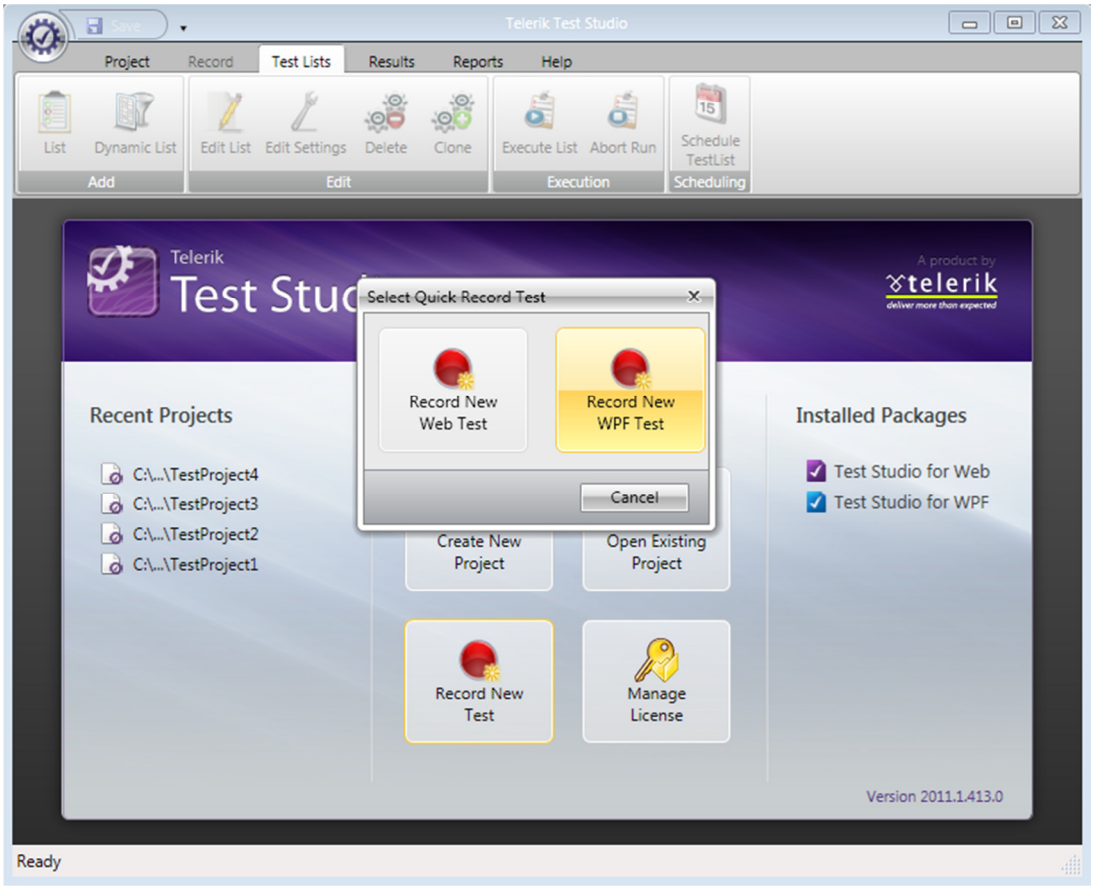
7. Notice that two more steps are added.



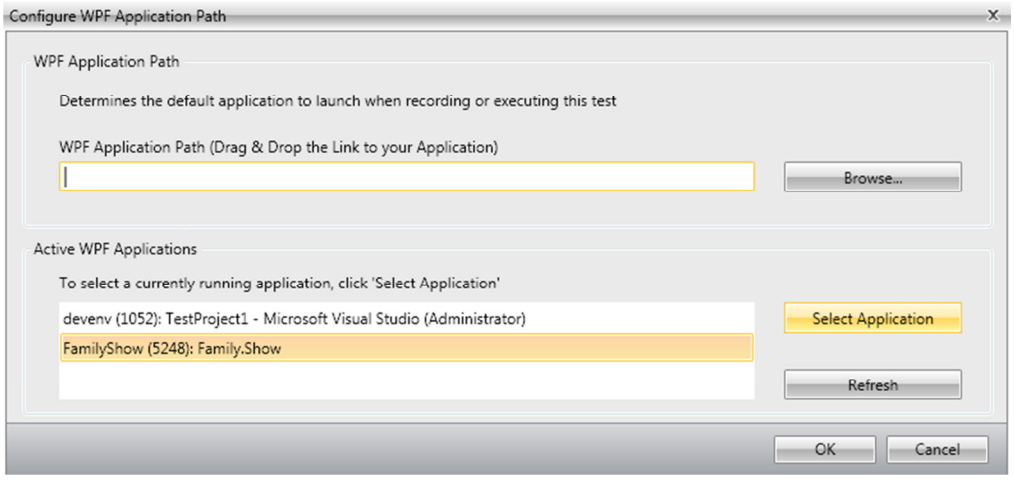
Close the browser to stop recording. That's how easy it is to create your first Web Test!

Standalone WPF Test

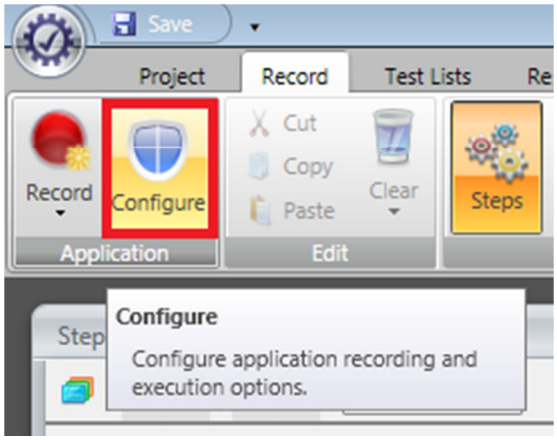
1. Launch Telerik Test Studio.
2. Click “Record New Test” and then “Record New WPF Test.”



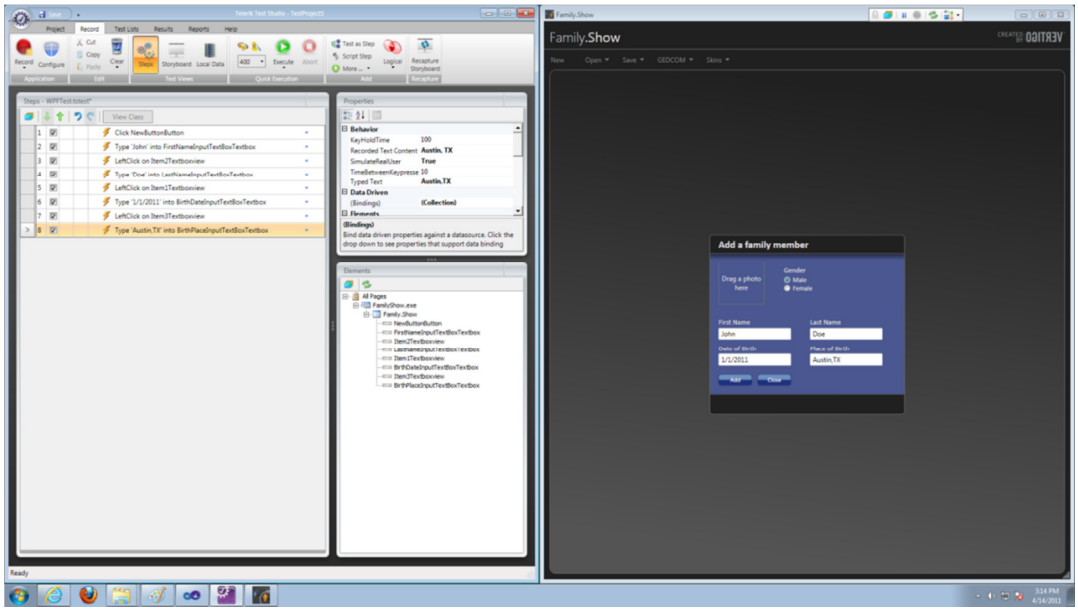
3. The “Configure WPF Application Path” window appears. There are two options to determine the default application to launch when recording or executing this test.
   a. WPF Application Path – drag and drop the shortcut to your WPF application into this text box, or click “Browse” and find it manually.
   b. Active WPF Applications – Telerik Test Studio detects all WPF apps currently running and lists them. Highlight the desired app and press “Select Application.” In this example FamilyShow is used, available from http://vertigo.com/Lab.aspx.



4. Click OK. If you need to later modify the WPF application path, choose “Configure” from the toolbar.



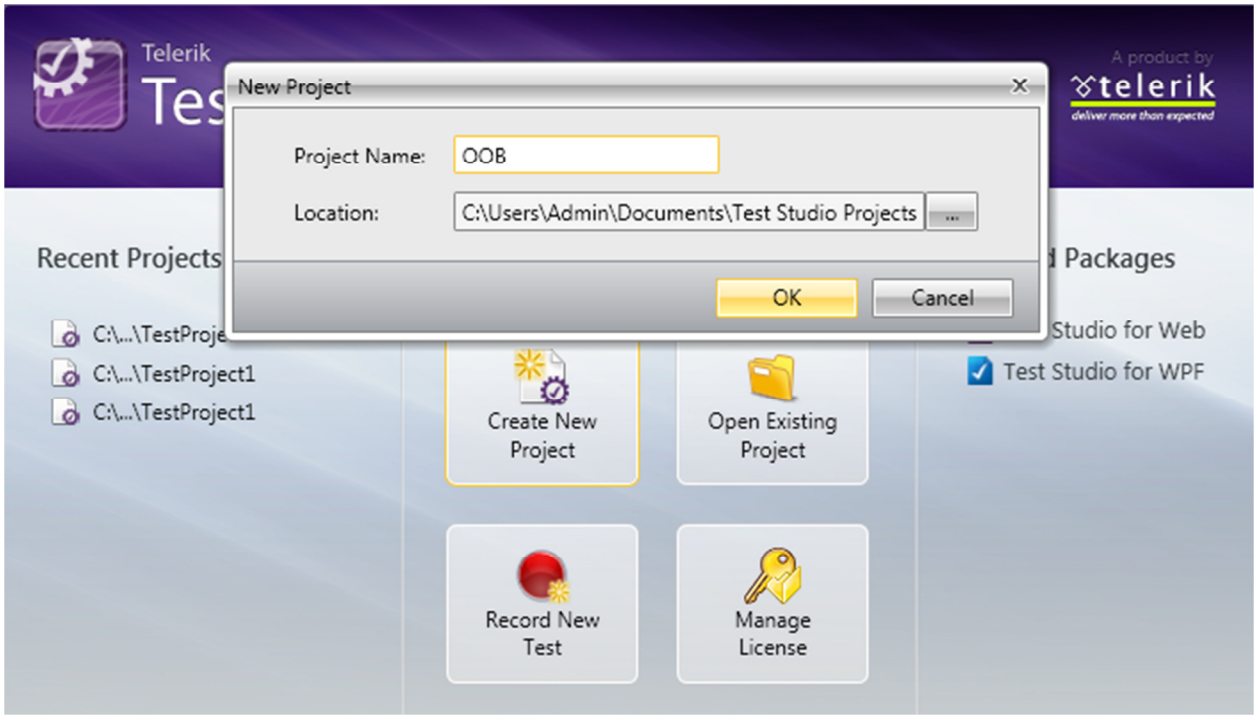
5. Hit “Record” to launch the app with the recording toolbar docked at the top.
6. Notice that steps are added as actions are taken within the application.



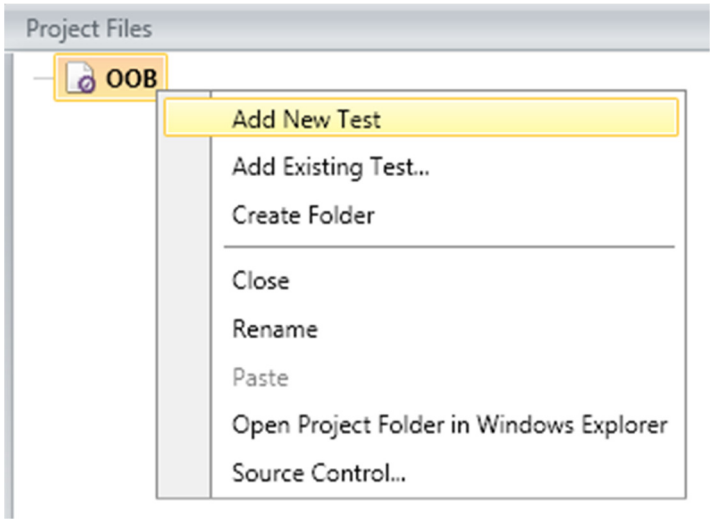
7. Close the browser to stop recording. That’s how easy it is to create your first WPF Test!

Standalone Silverlight Test

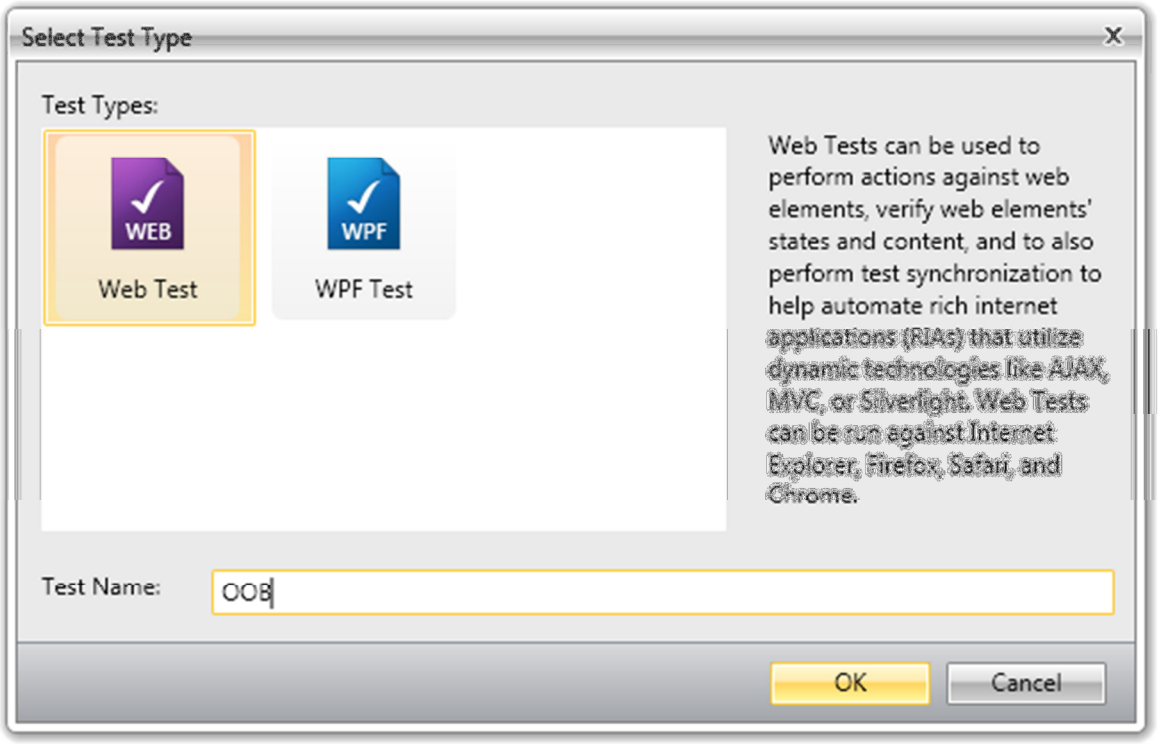
1. Launch Telerik Test Studio.
2. Click “Create New Project,” name the project, and click “OK.”



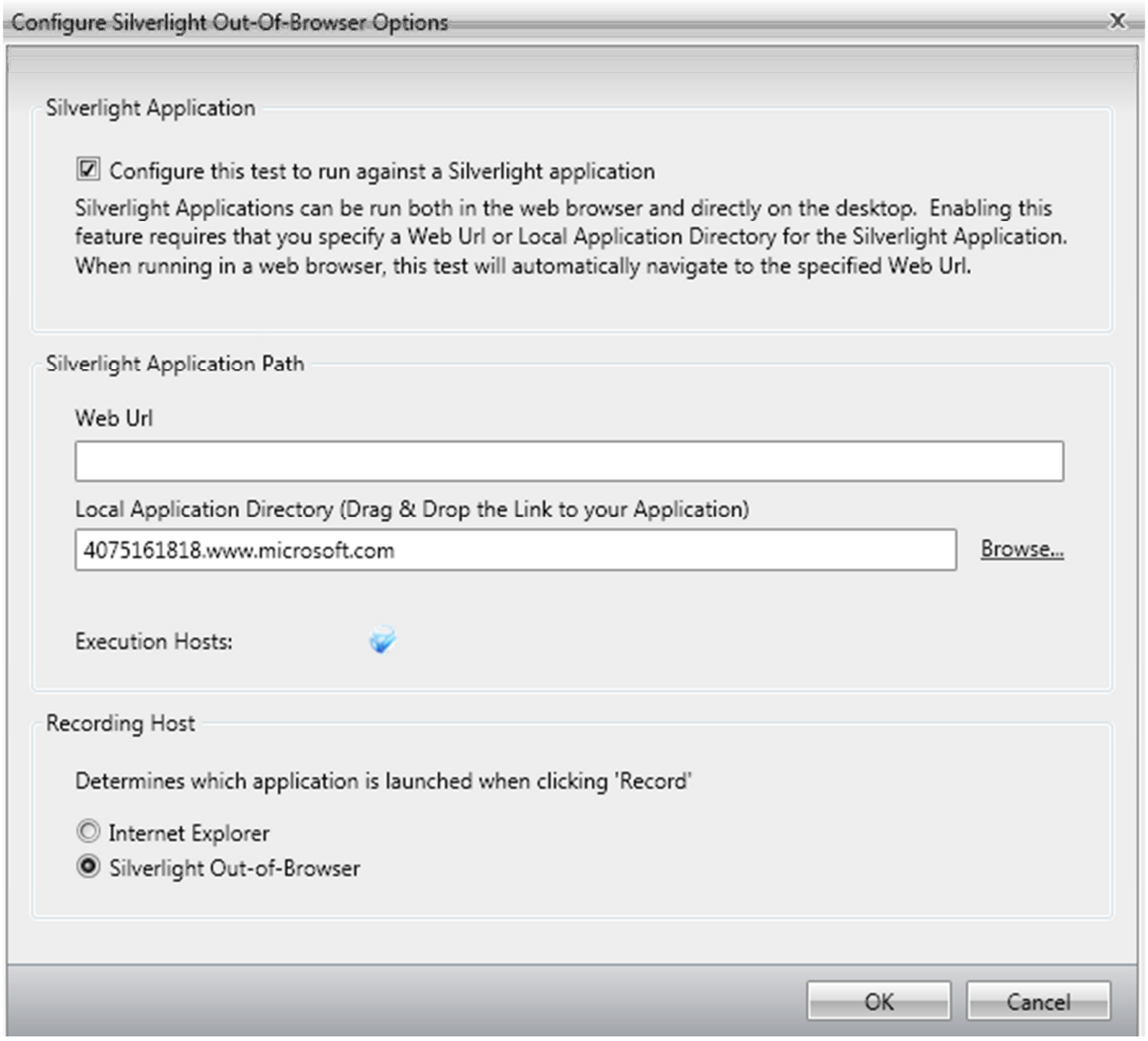
3. Right click the project node and click “Add New Test.”



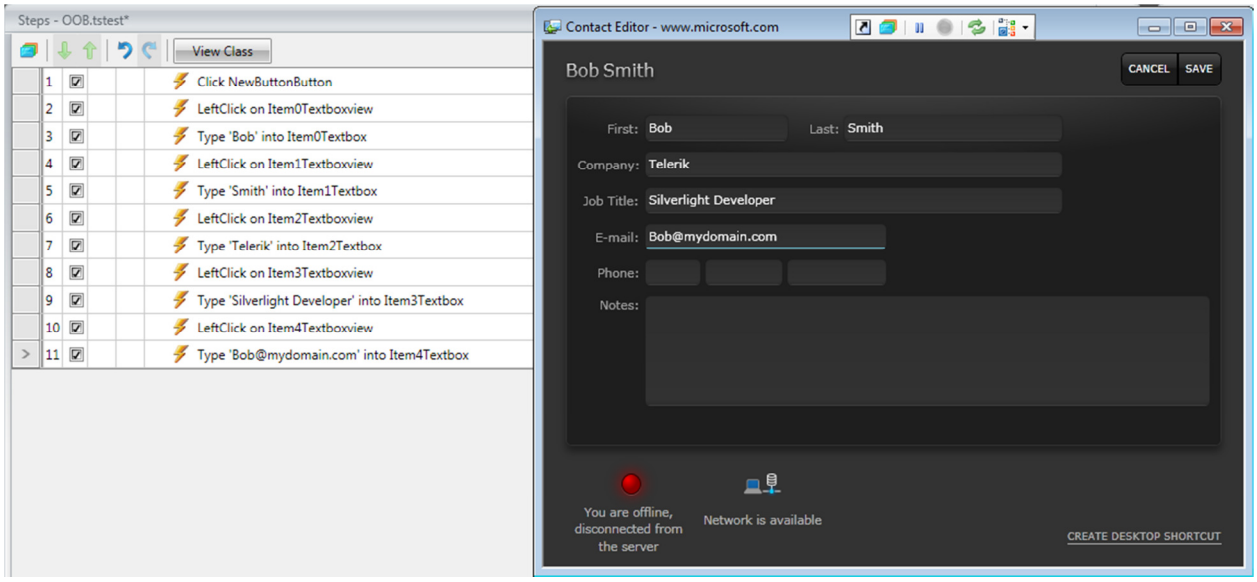
4. Select “Web Test,” name the test, press Enter, and double click it to open it.



5. Click the “Out-Of-Browser” icon in the “Silverlight” ribbon.
6. Check the “Configure this test to run against a Silverlight application” box in Out-Of-Browser options.
7. Enter either the Web URL, browse for the Local Application Directory, or drag and drop a link to your Silverlight application.
8. Select the “Recording Host.”
   a. Select “Internet Explorer” if you entered a Web URL.
   b. Select “Silverlight Out-of-Browser” if you entered a Local Application Directory.



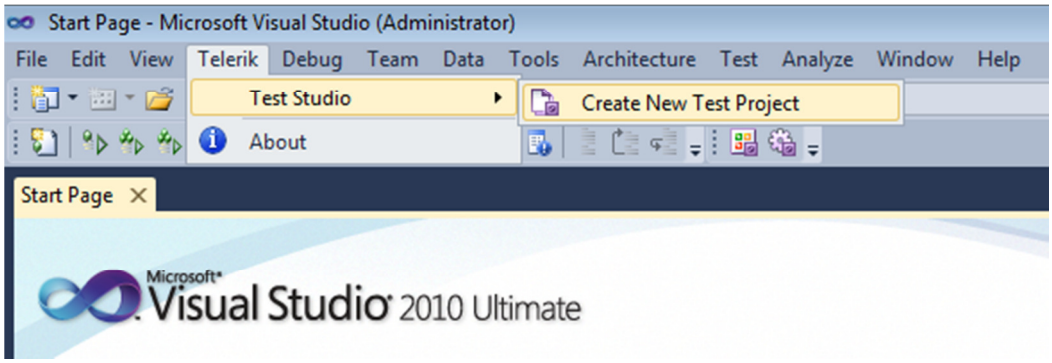
9. Click “Record” to launch the browser or out-of-browser app with the recording toolbar docked at the top.
10. Notice that steps are added to the test as actions are taken within the browser/application.



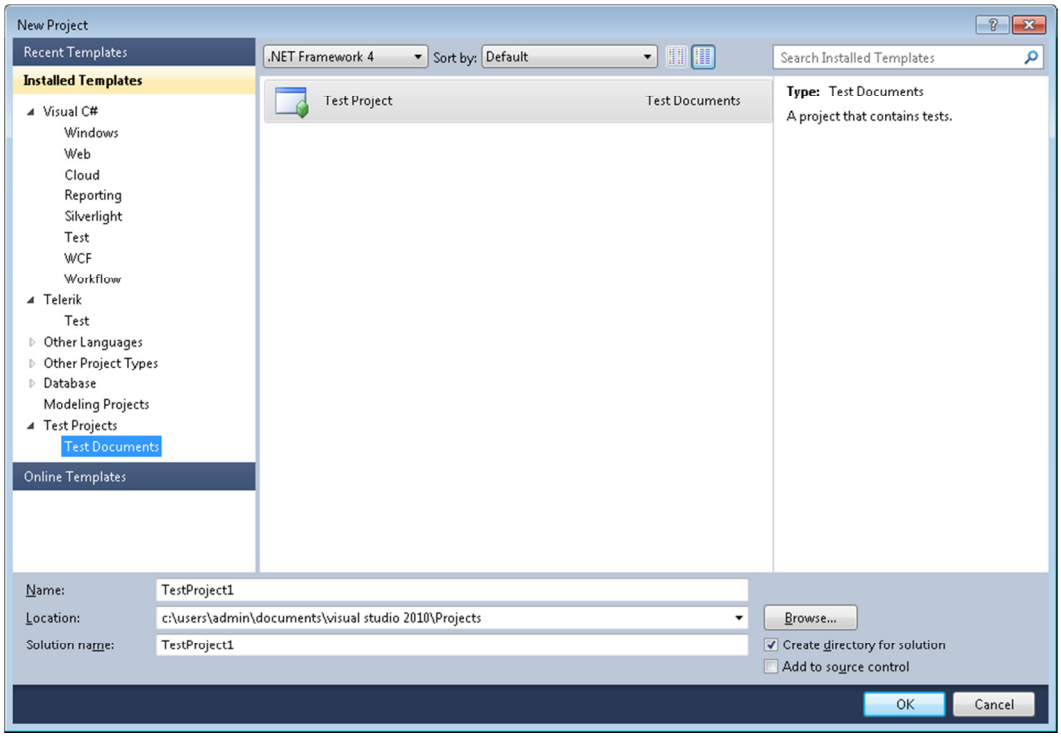
11. Close the browser/application to stop recording. That's how easy it is to create your first Silverlight Test!

Visual Studio Plug-In Web Test

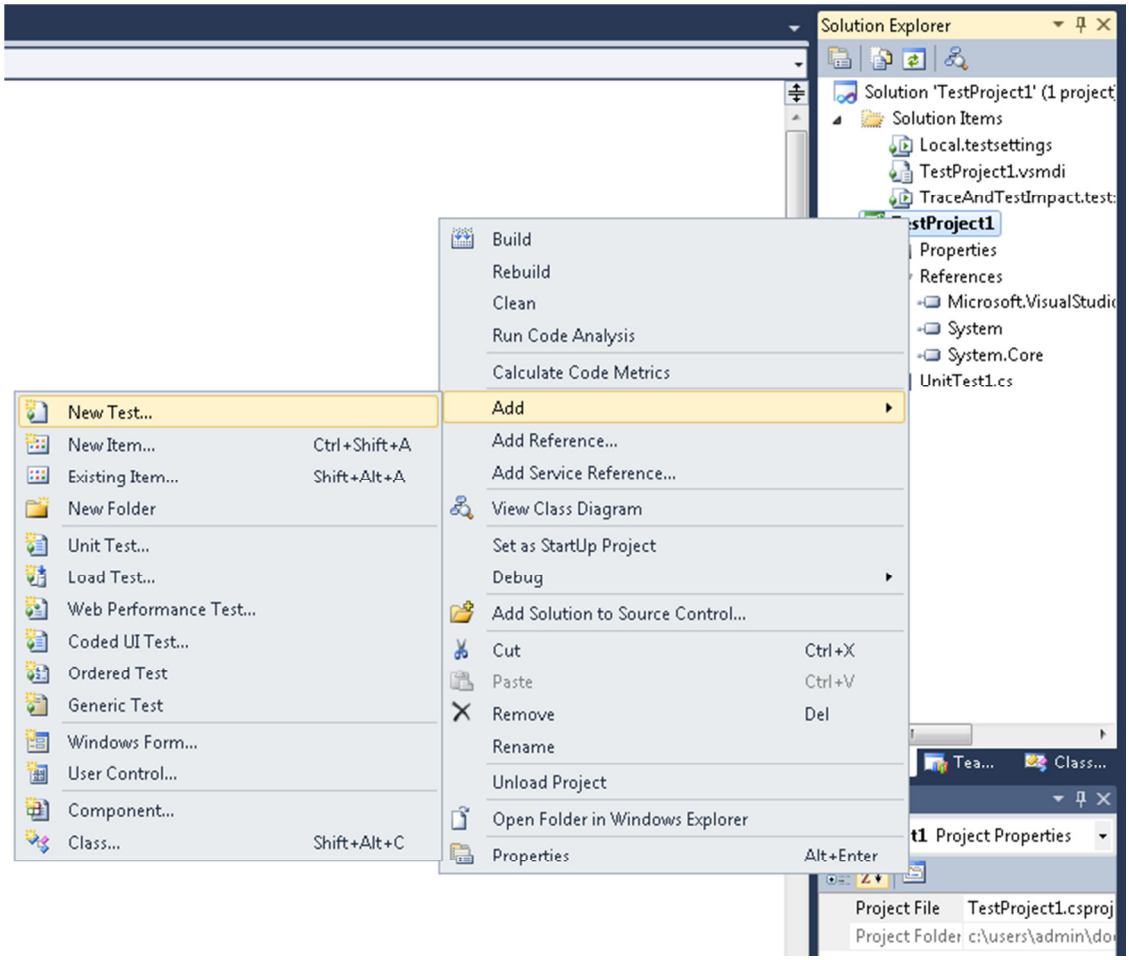
1. Within Visual Studio, click “Telerik”->”Test Studio”->”Create New Test Project.”



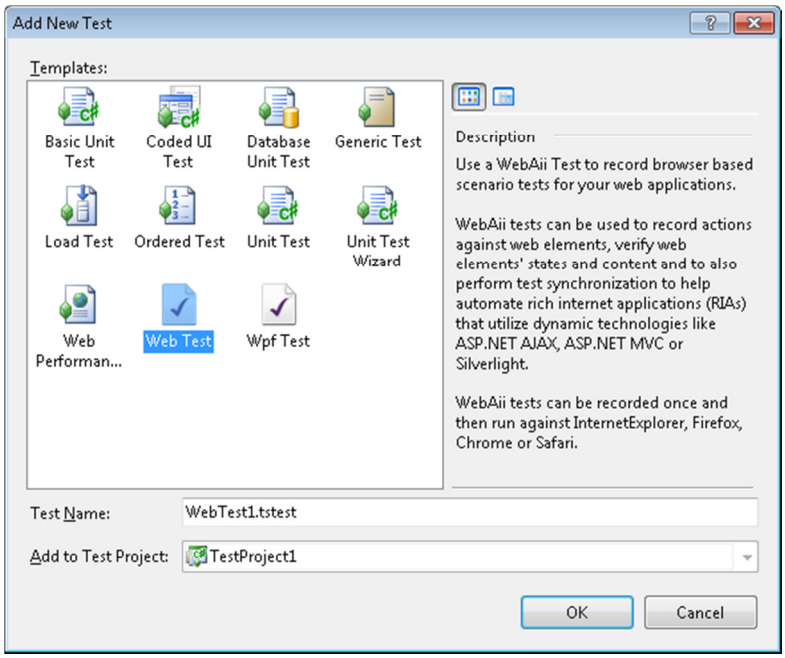
2. Choose “Test Projects” -> “Test Documents” -> “Test Project,” name the project, and click OK.



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

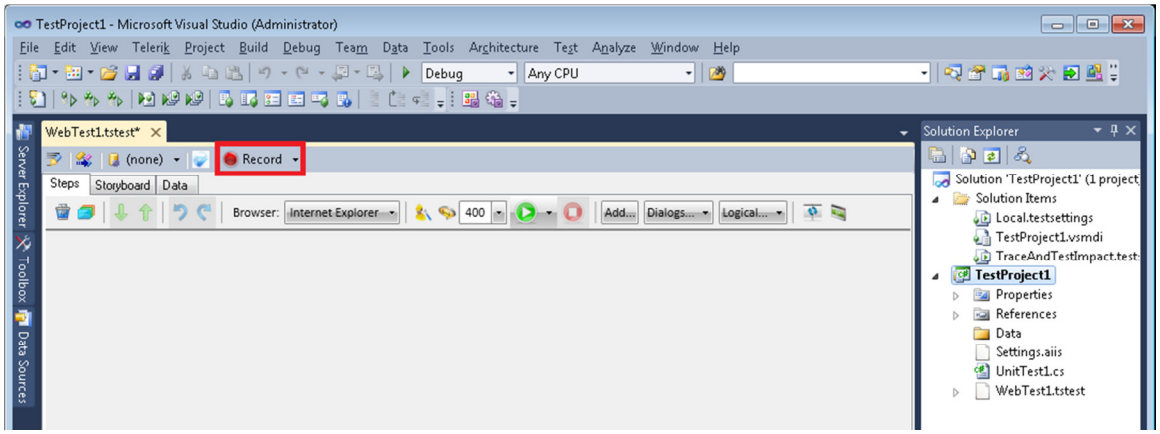


4. Click on the “Web Test” template, name the test, and select OK.

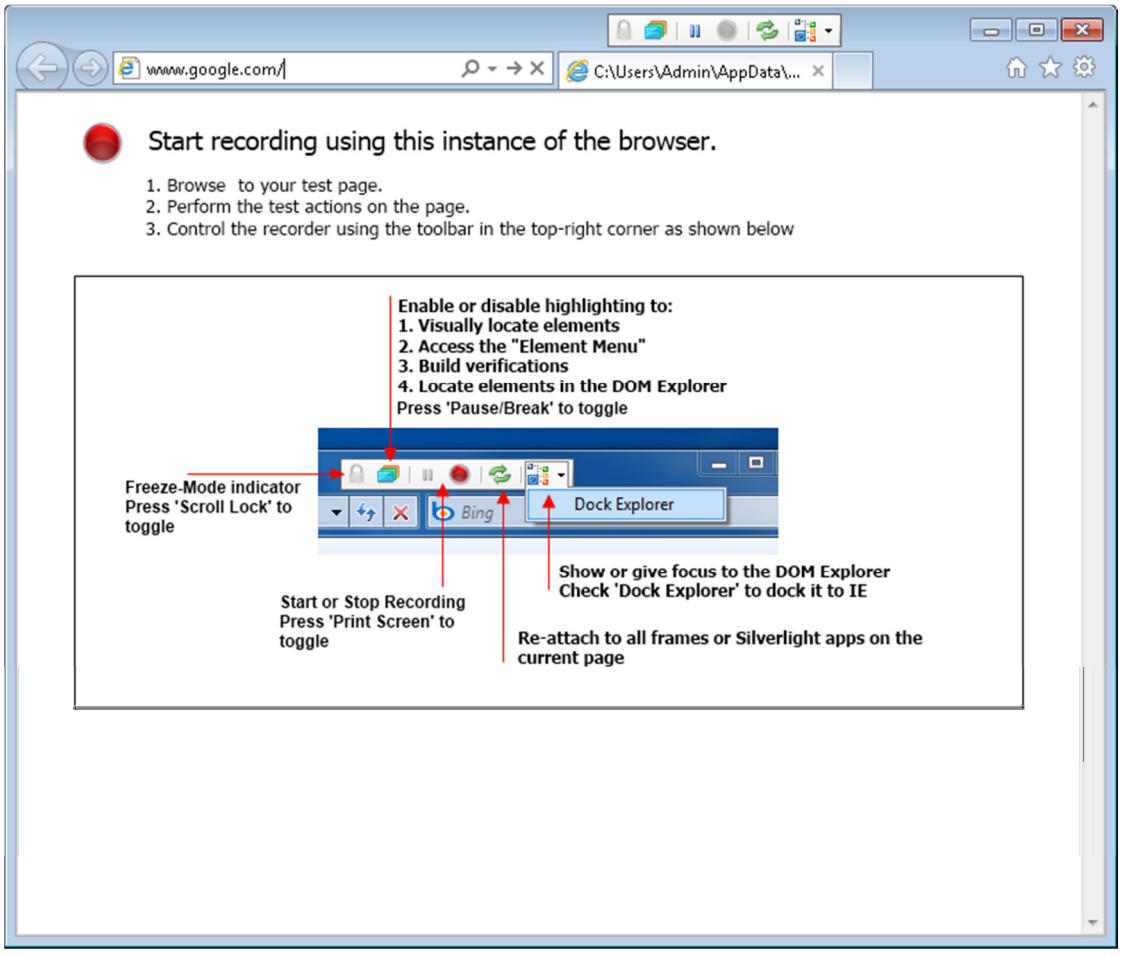


5. The new, empty test opens automatically.

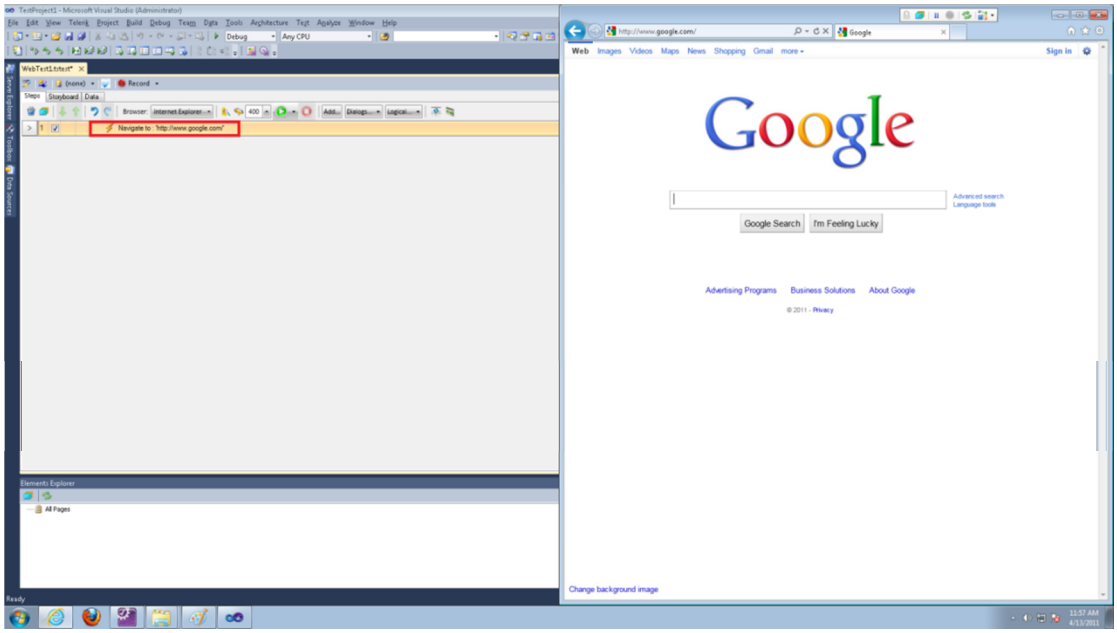
6. Click on the “Record” button (in red below) to begin recording test steps.



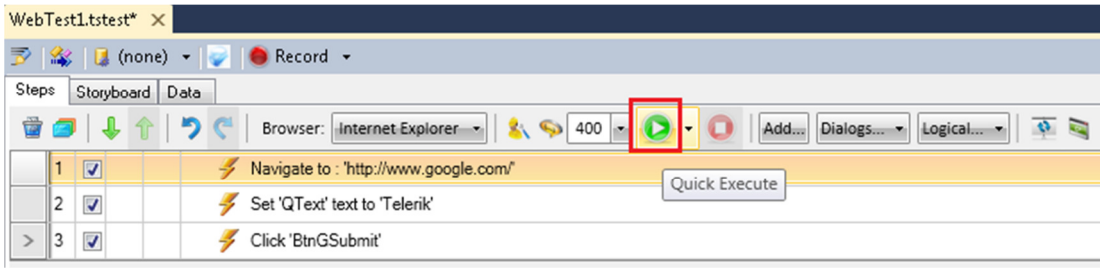
7. Internet Explorer will automatically open in record mode.



8. Type www.google.com in the URL box and press Enter.
9. Notice a recording step is added to the Steps tab.



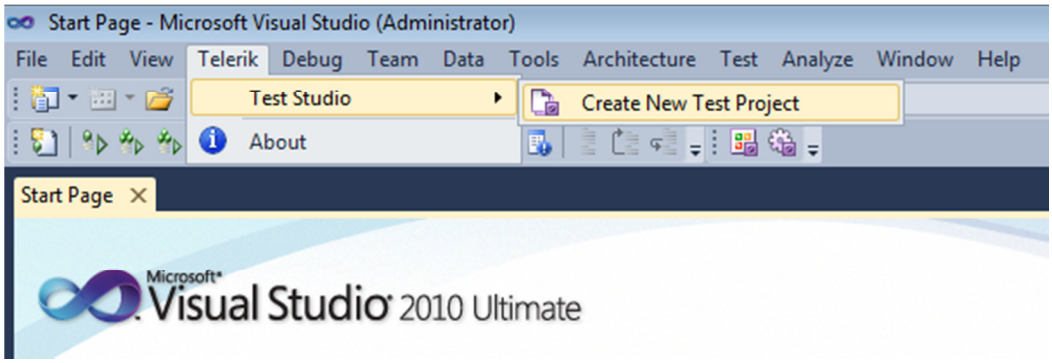
10. Type “Telerik” in the Google search box and click the search button.
11. Notice that two more steps are added.



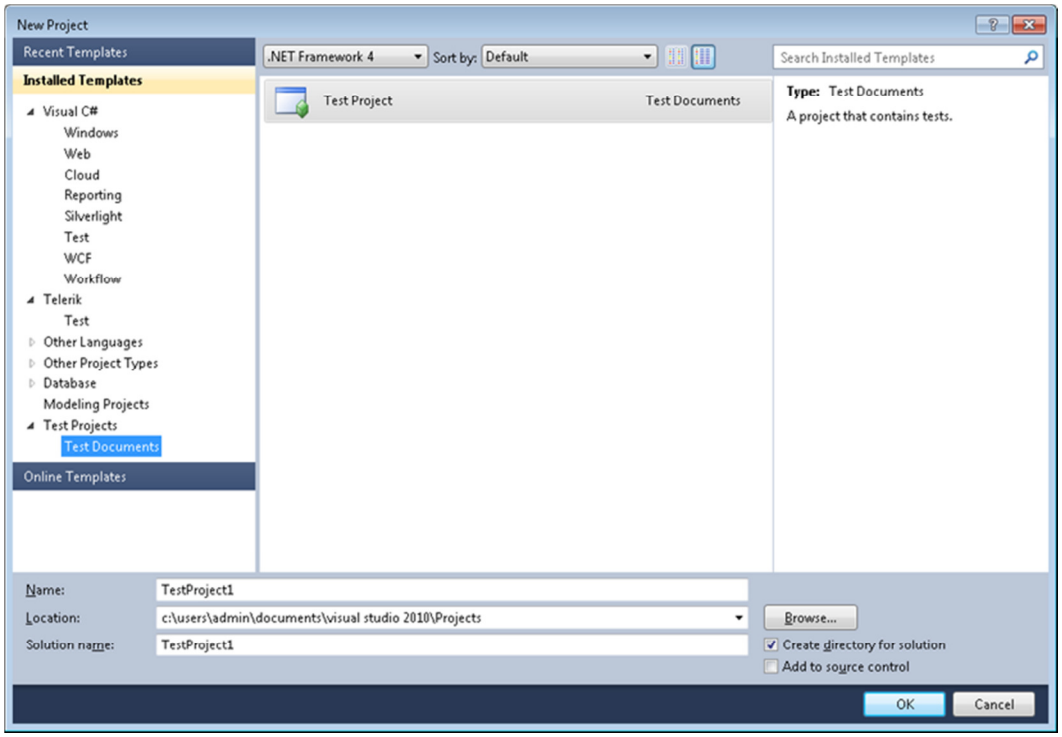
12. Close the recording browser window. That’s how easy it is to create your first Web Test! Save and build your project.

Visual Studio Plug-In WPF Test

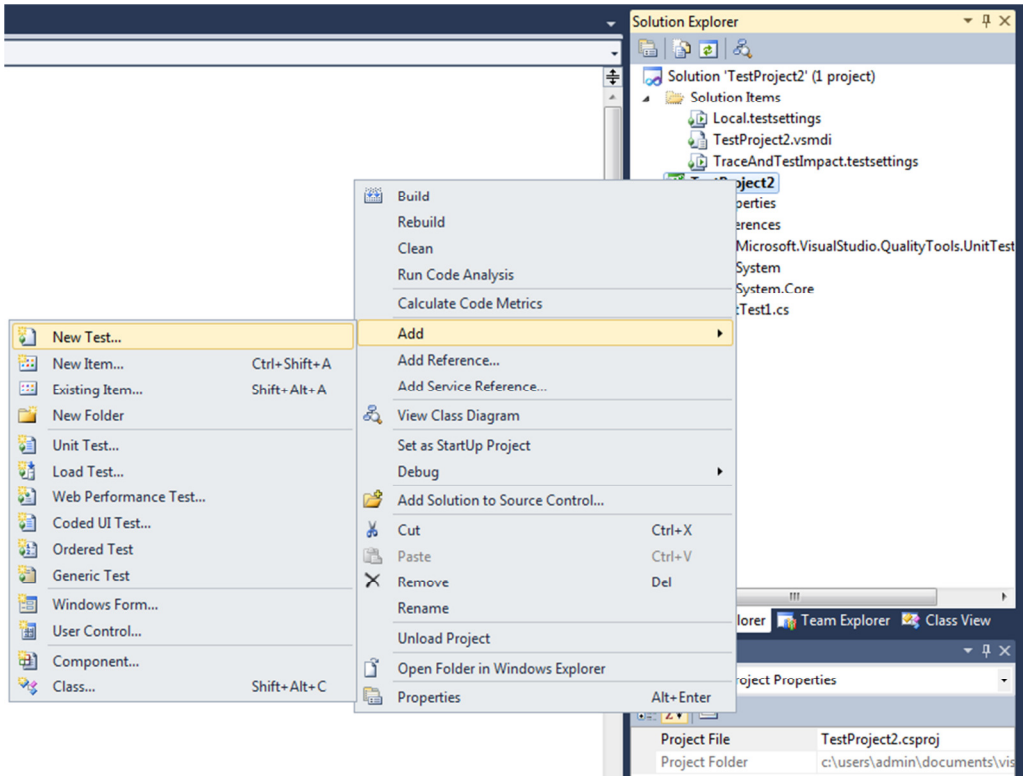
1. Within Visual Studio, click “Telerik”->”Test Studio”->”Create New Test Project.”



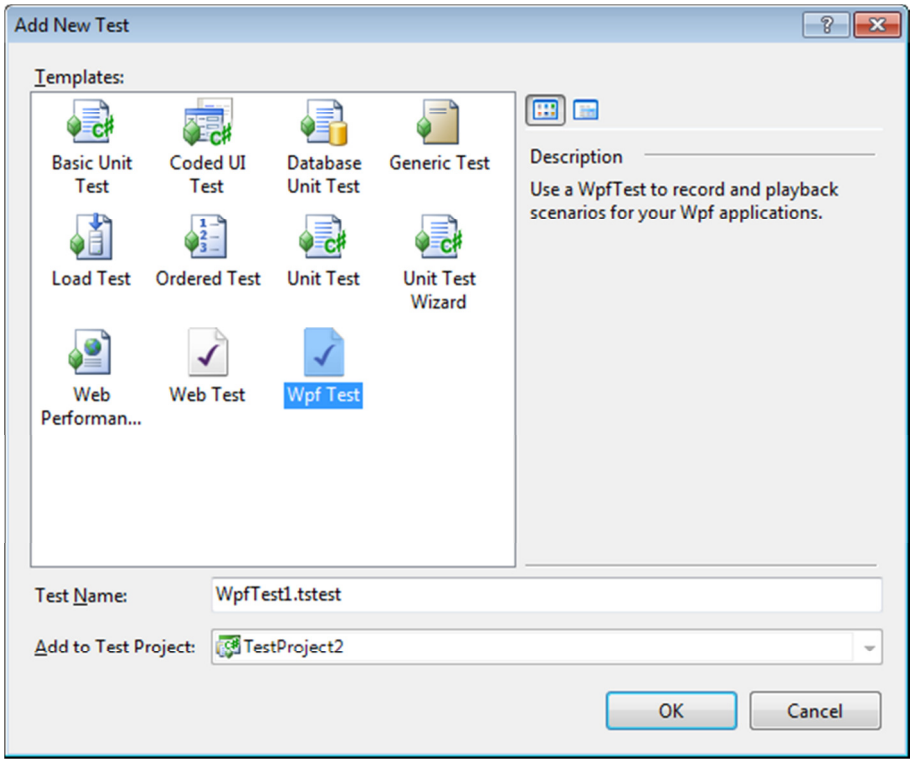
2. Choose “Test Projects” -> “Test Documents” -> “Test Project,” name the project, and click OK.



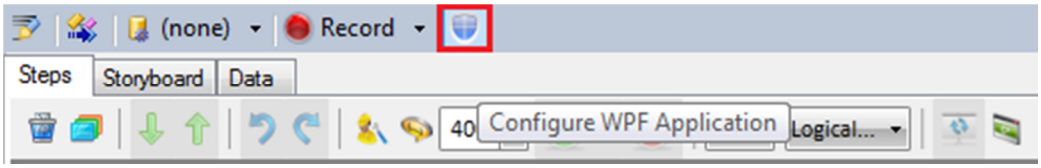
3. Right click on the project node in the solution explorer and select “Add -> New Test…”



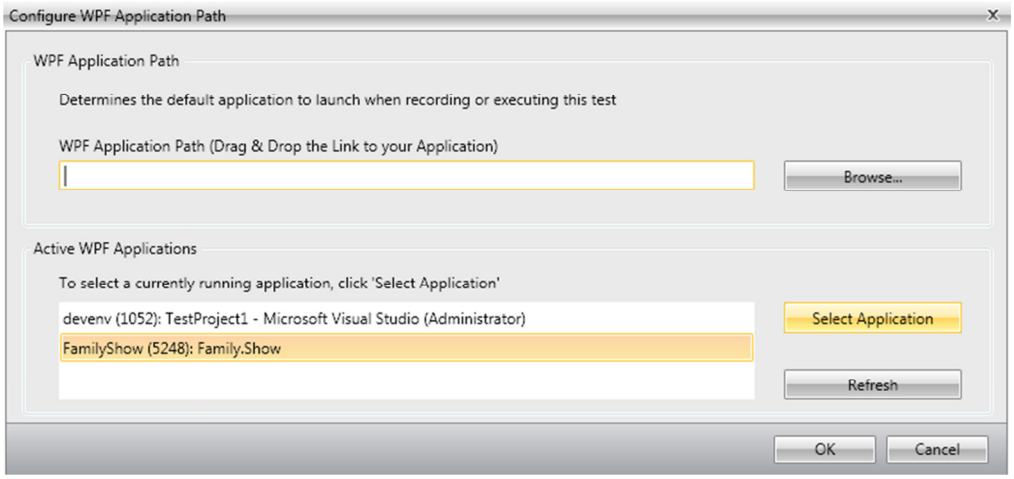
4. Click on the “WPF Test” template, name the test, and select OK.



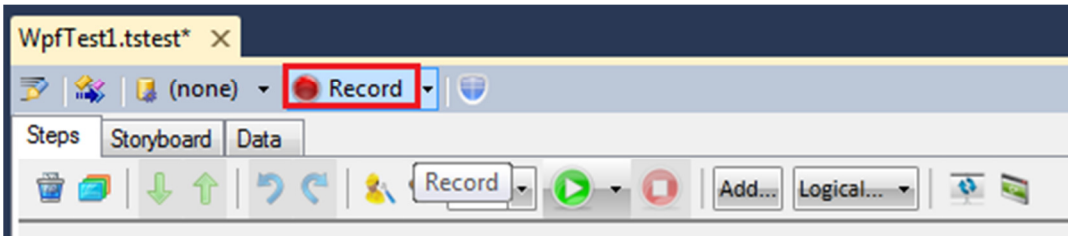
5. The new, empty test opens automatically. Click the “Configure WPF Application” icon in the toolbar.



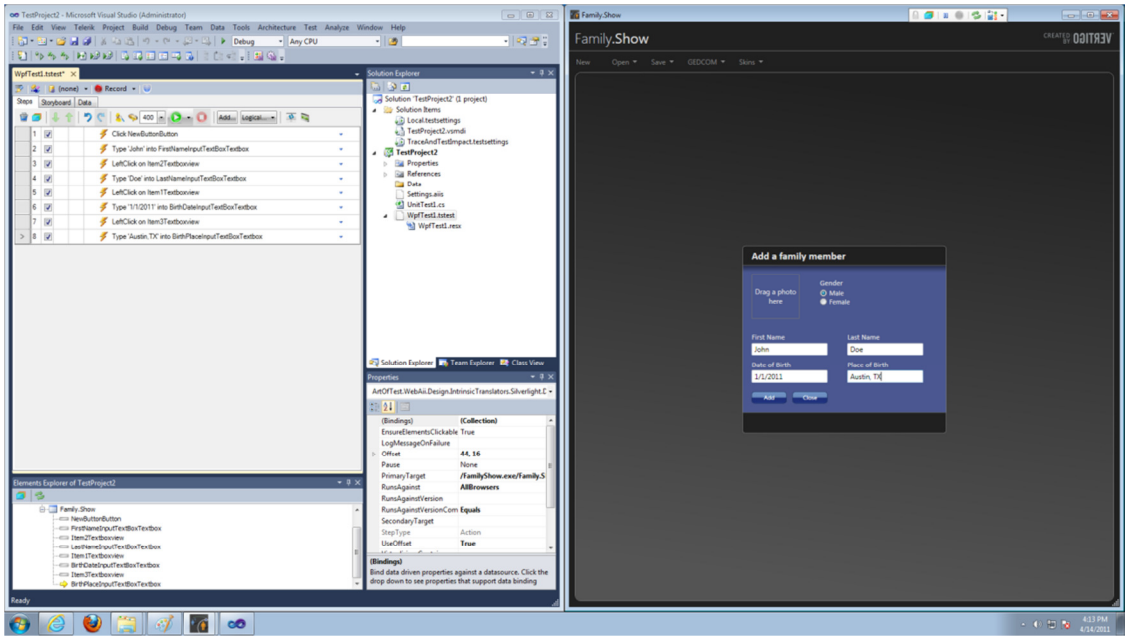
6. The “Configure WPF Application Path” window appears. There are two options to determine the default application to launch when recording or executing this test.
   a. WPF Application Path – drag and drop the shortcut into this text box, or click “Browse” and find it manually.
   b. Active WPF Applications – Telerik Test Studio detects all WPF apps currently running and lists them. Highlight the desired app and press “Select Application.” In this example FamilyShow is used, available from http://vertigo.com/Lab.aspx.



7. Hit “Record” to launch the app with the recording toolbar docked at the top.



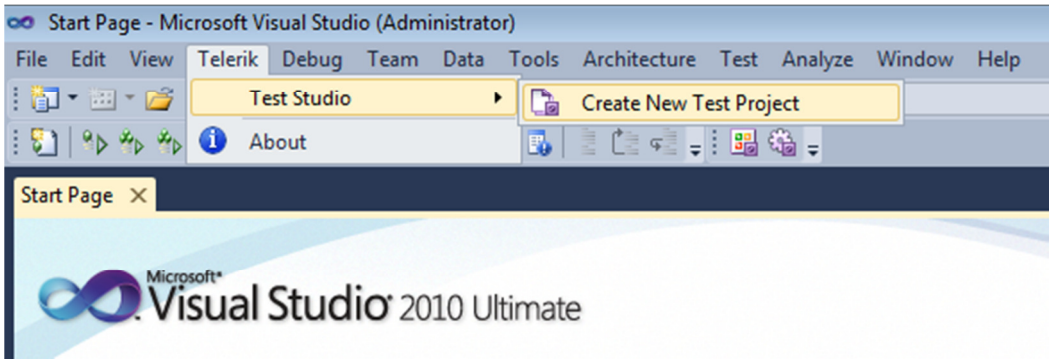
8. Notice that steps are added as actions are taken within the application.



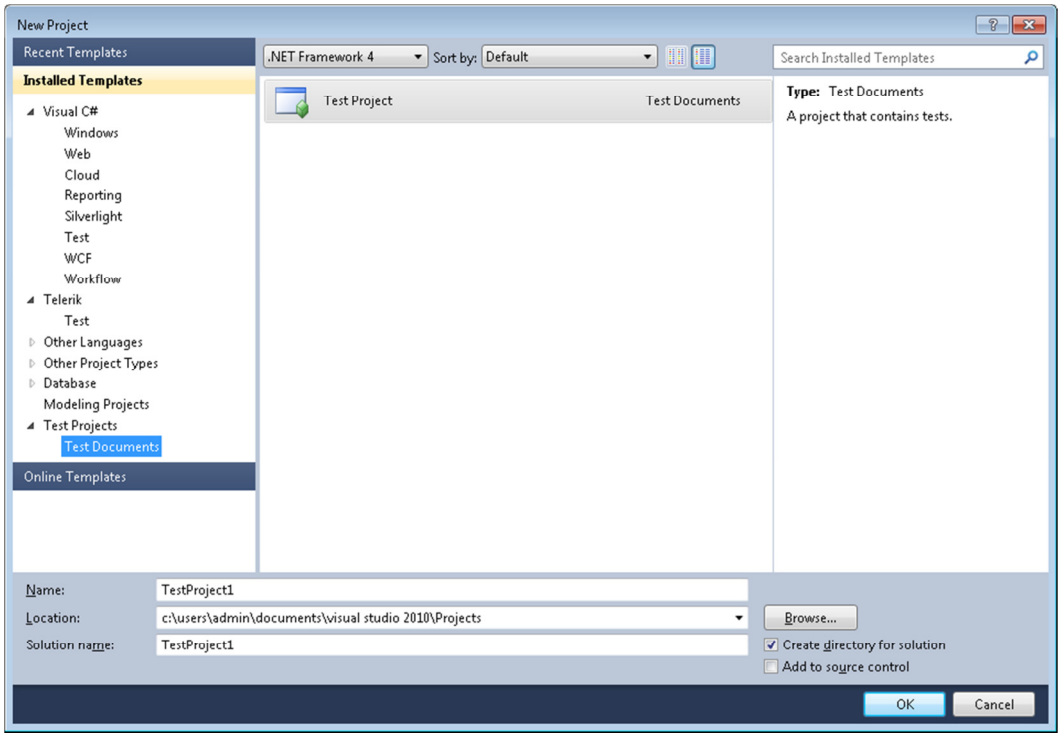
9. Close the application window. That's how easy it is to create your first WPF Test! Save and build your project.

Visual Studio Plug-In Silverlight Test

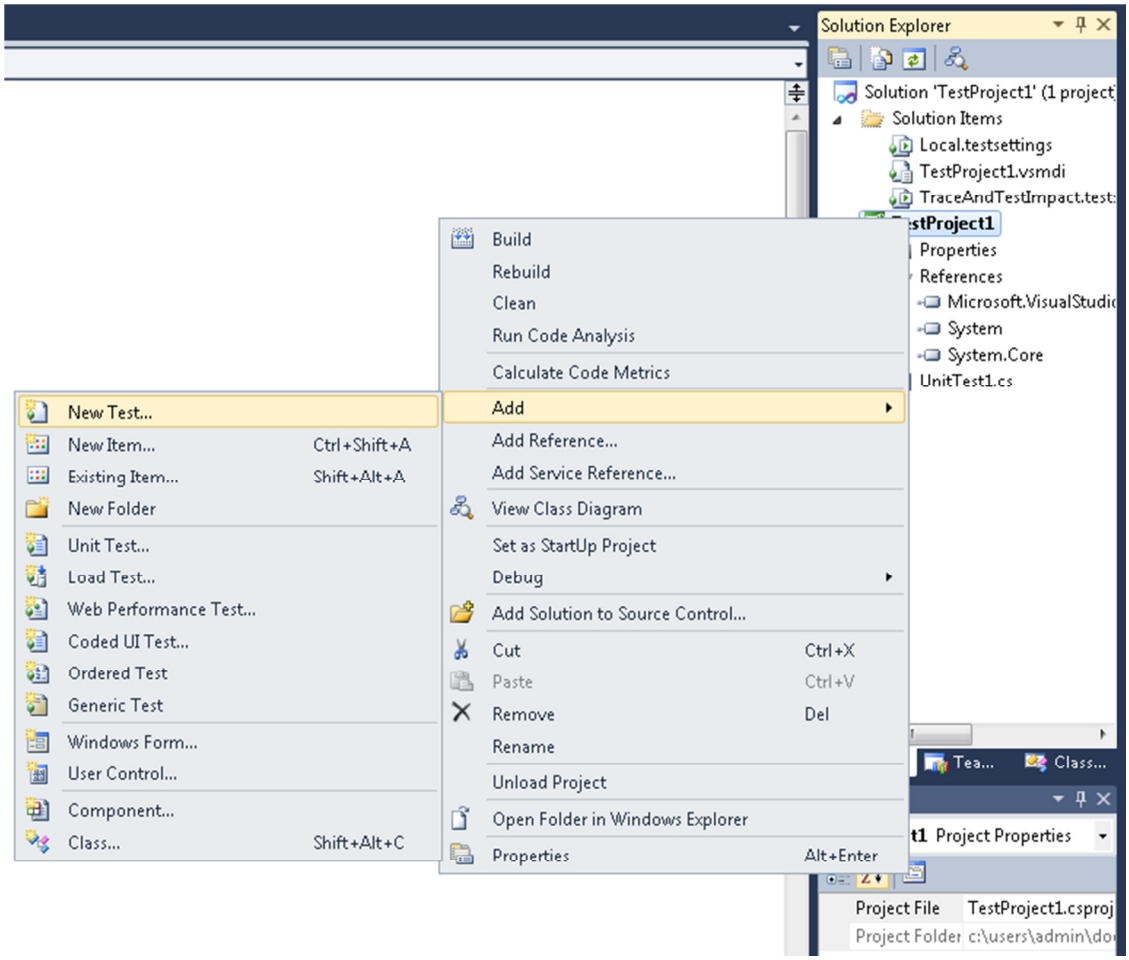
1. Within Visual Studio, click “Telerik”->”Test Studio”->”Create New Test Project.”



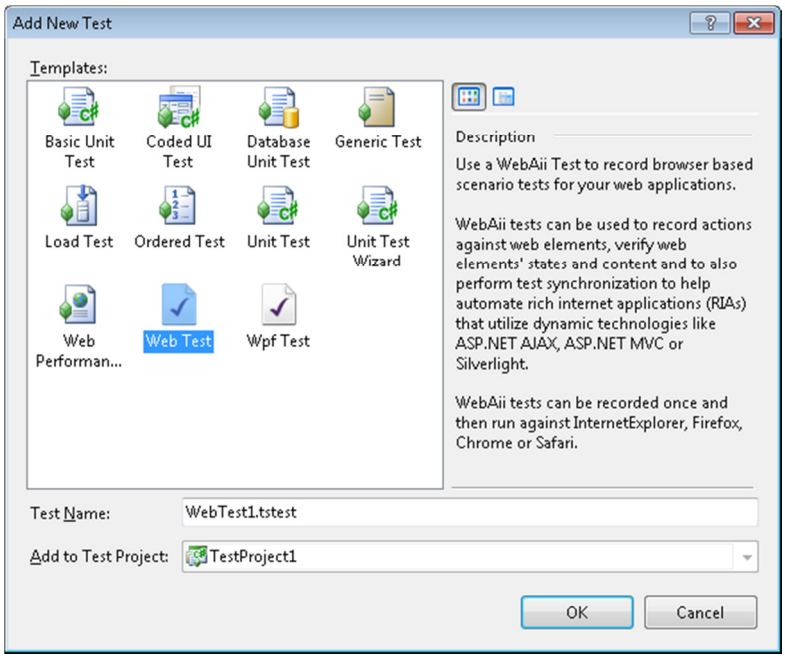
2. Choose “Test Projects” -> “Test Documents” -> “Test Project,” name the project, and click OK.



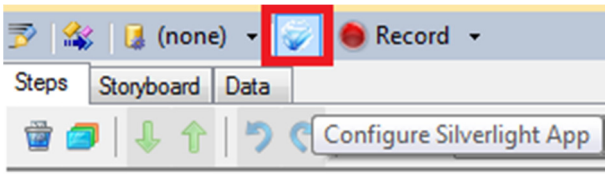
3. Right click on the project node in the solution explorer and select “Add -> New Test.”



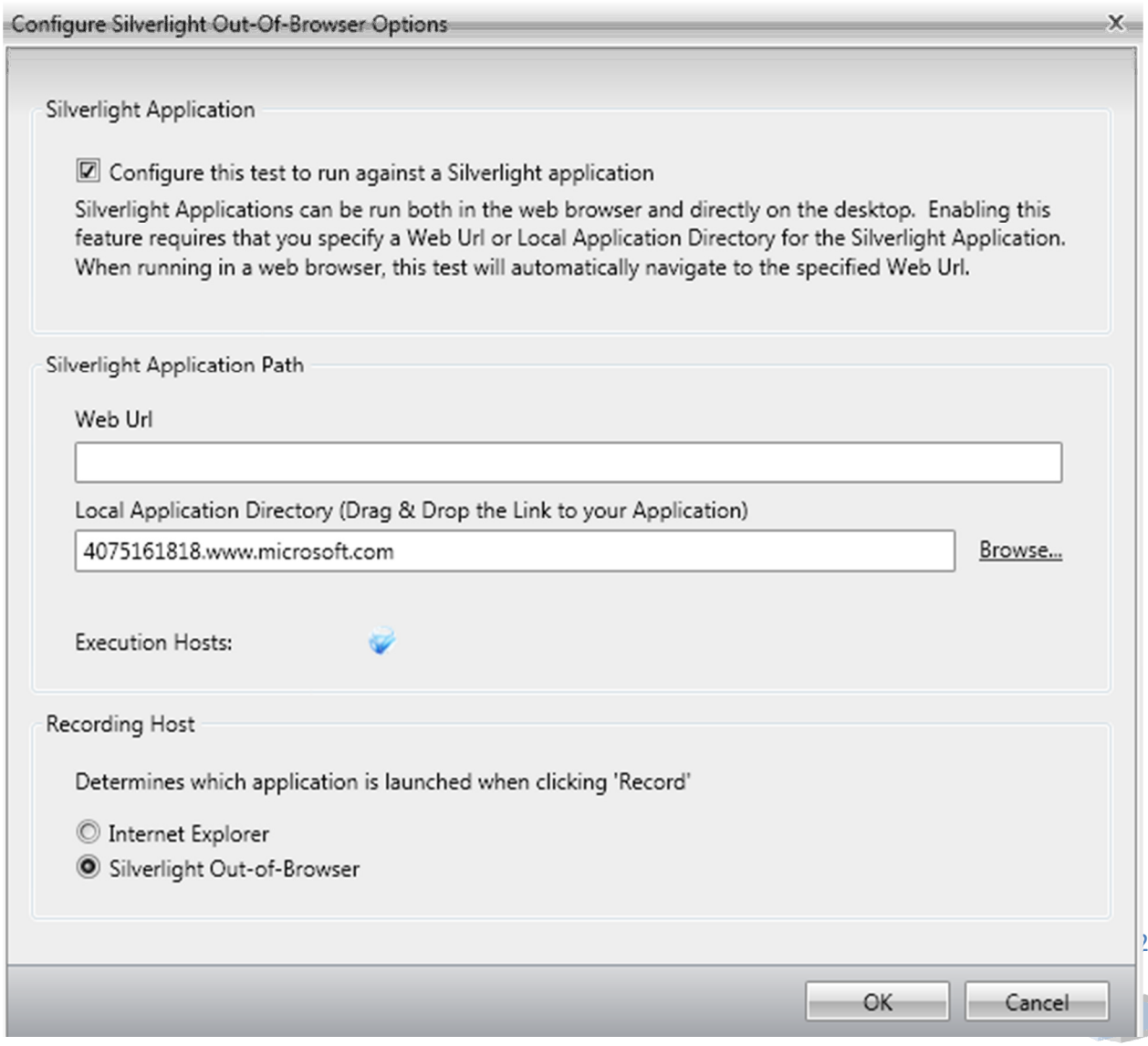
4. Click on the “Web Test” template, name the test, and select OK.



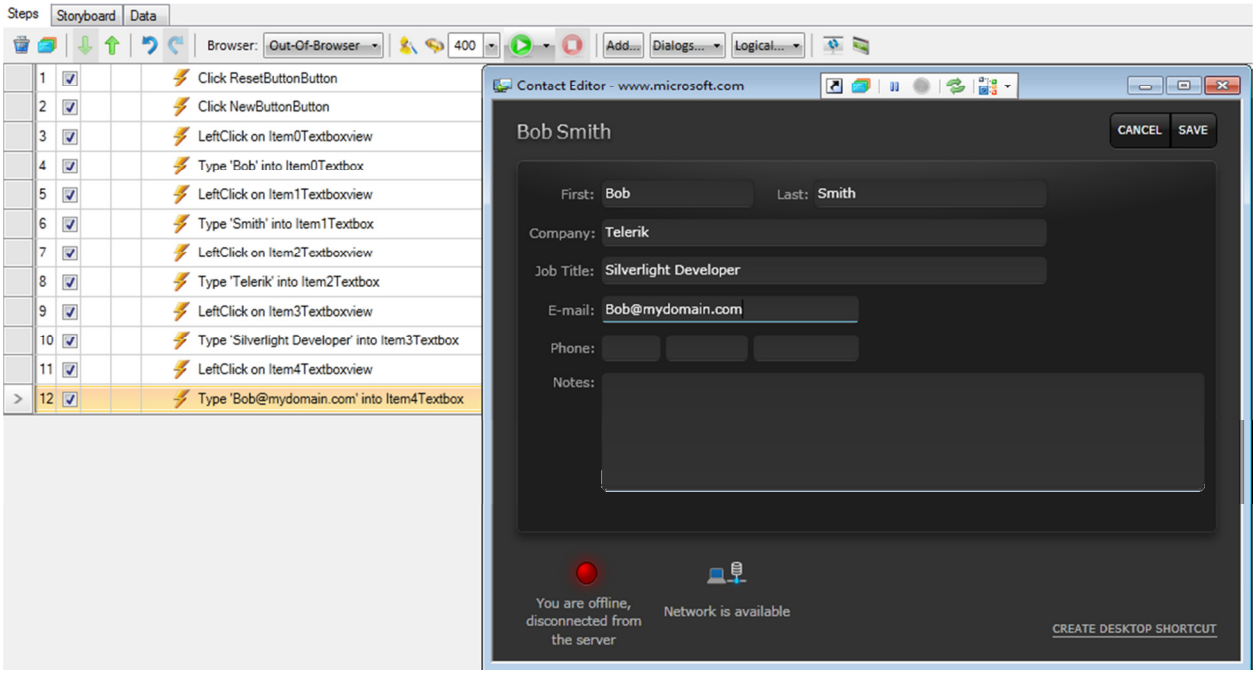
5. The new, empty test opens automatically. Click the “Configure Silverlight App” icon in the toolbar.



6. Check the “Configure this test to run against a Silverlight application” box in Out-Of-Browser options.
7. Enter either the Web URL, browse for the Local Application Directory, or drag and drop a link to your Silverlight application.
8. Select the “Recording Host.”
   a. Select “Internet Explorer” if you entered a Web URL.
   b. Select “Silverlight Out-of-Browser” if you entered a Local Application Directory.



9. Click “Record” to launch the browser or out-of-browser app with the recording toolbar docked at the top.
10. Notice that steps are added to the test as actions are taken within the browser/application.



11. Close the browser/application to stop recording. That’s how easy it is to create your first Silverlight Test!

Recording Toolbar Overview

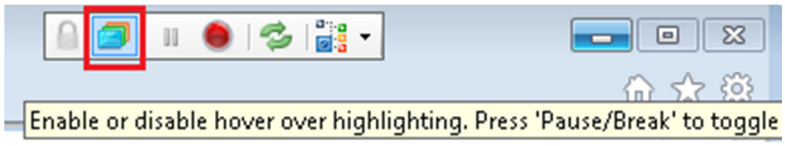
The recording toolbar attached to the IE browser or WPF app recording window contains (from left to right):

a. Go back to host – Makes Visual Studio active (locked in Standalone edition).
b. Enable or disable hover over highlighting. Press 'Pause/Break' to toggle.
c. Pause Recording. Press 'Print Screen' to toggle.
d. Start Recording. Press 'Print Screen' to toggle.
e. Reconfigure Recorder. (Restores communication between Test Studio and the recording browser window or WPF app if necessary)
f. Show the DOM Explorer.

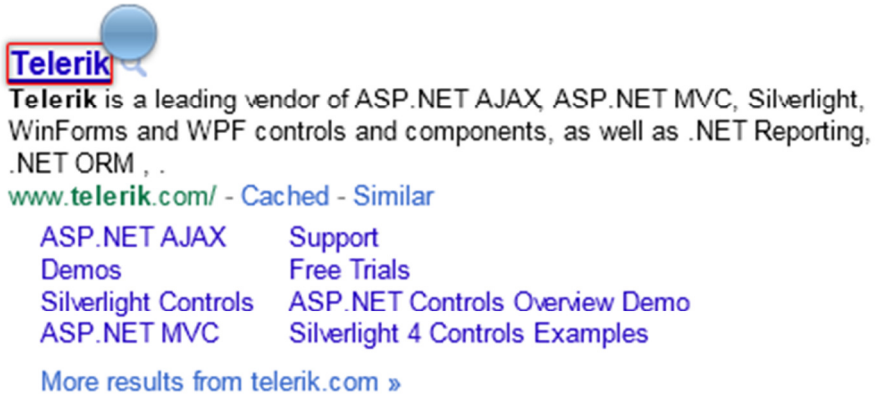
Add Quick Verification Steps

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

1. Enter record mode.
2. Enable hover over highlighting by clicking the icon (in red below) in the docked taskbar.

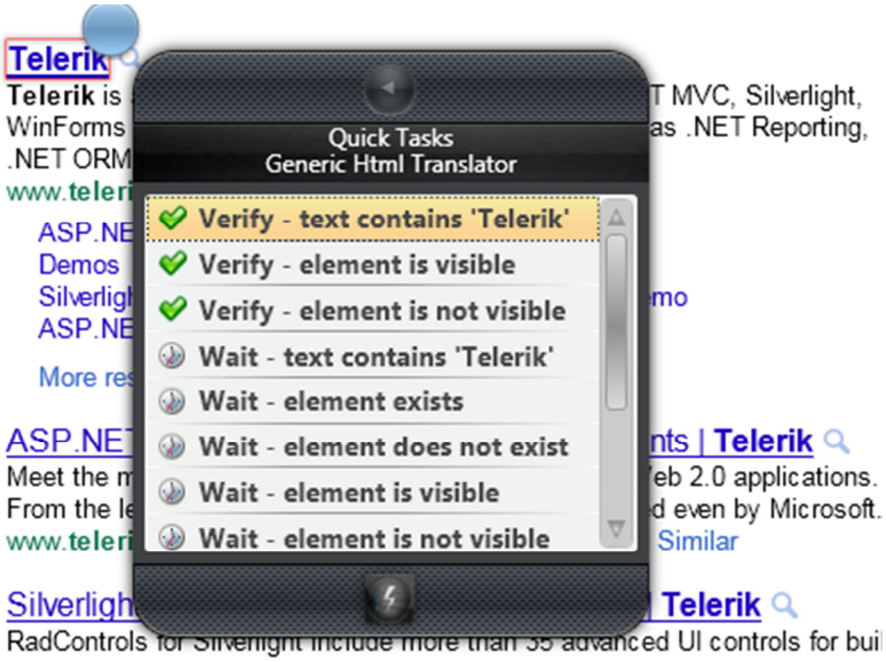


3. In the recording browser window, hover over the element against which to verify.



4. Wait for the blue circular nub to appear and click on it.
5. This opens the Element Menu.

6. Click the Quick Tasks button (in red above). This opens a list of available quick tasks. The type of element selected dictates what is listed.



7. Double clicking one of these items will add that Verify/Wait step to the test.

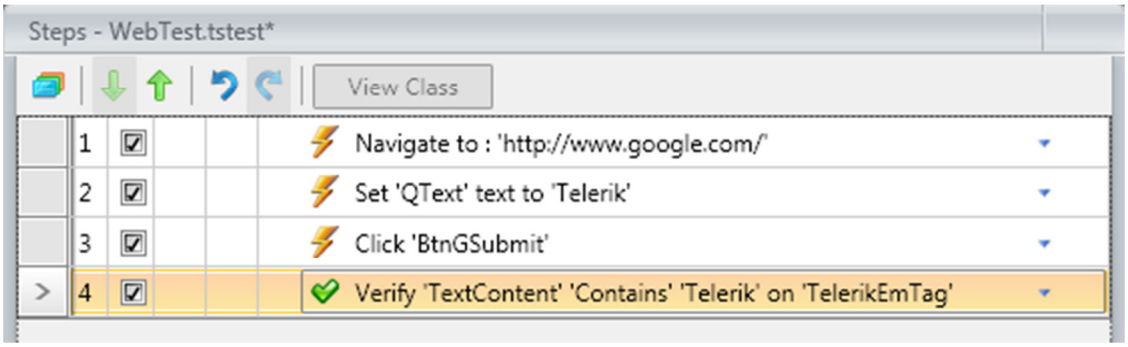


Figure 1 - Standalone version

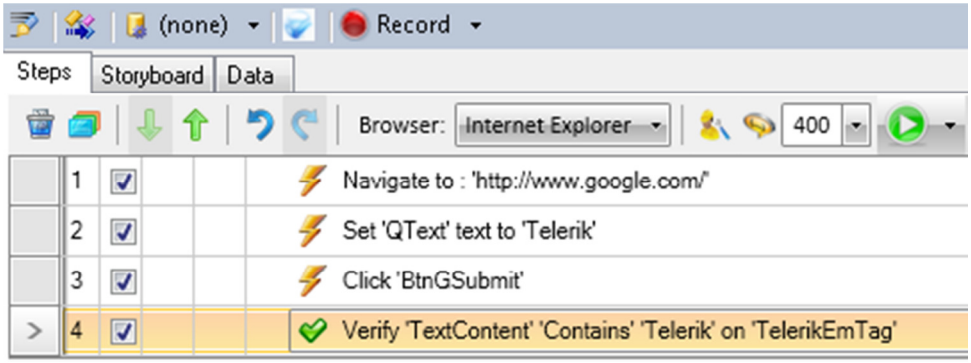


Figure 2 - VS plugin

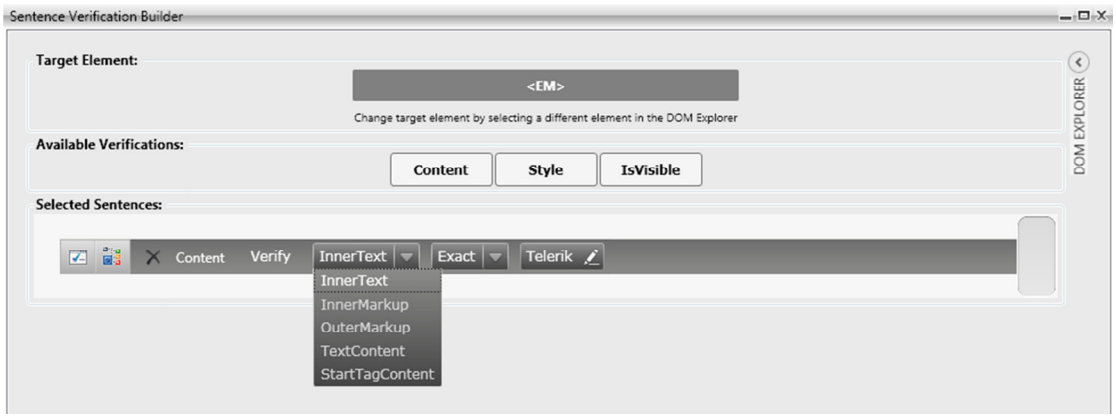
Create Advanced Test Verifications

Custom Verifications are created from the Sentence Verification Builder dialog.

1. Reopen the Element Menu and click “Build Verification.”

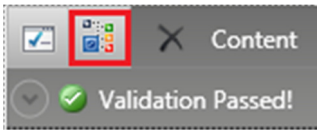
2. The verification builder offers sentence based verifications of elements. Start by selecting a type of Available Verifications.

3. When crafting verifications, content is dynamically built against the currently selected element. As selections are made, default values are populated according to values the element contains.
4. For example, choose “Content” as the verification type and three menu options appear. Click on the down arrow next to each option to see a list of possible values.



5. Validate the verification by clicking the “Verification” icon.

6. Locate the current element in the DOM tree by clicking the “Locate in DOM” icon.



7. Delete or start over by clicking the “Delete” icon.

8. Verifications can be crafted to verify many different values, styles, or attributes of an element. Craft multiple sentences by selecting a type and filling in the verification criteria.
   NOTE: Each sentence will add a separate verification step.
9. Once finished building the verification, select OK to add it as a step to the current test.

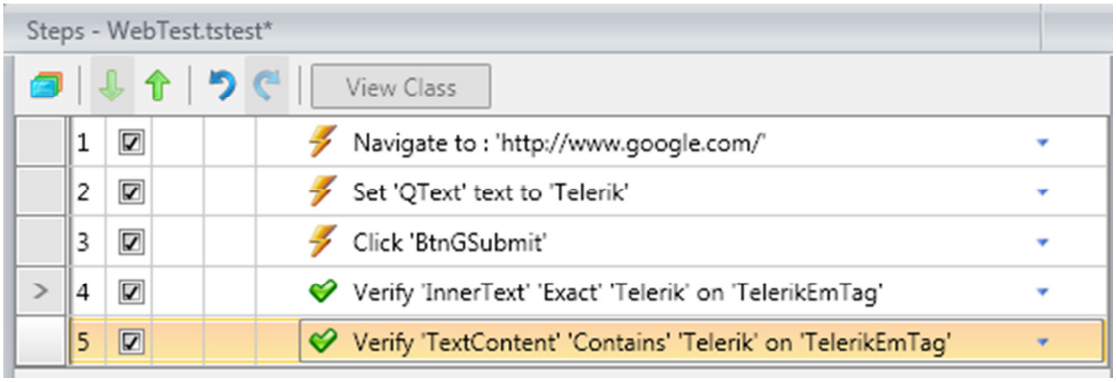


Figure 3 - Standalone version

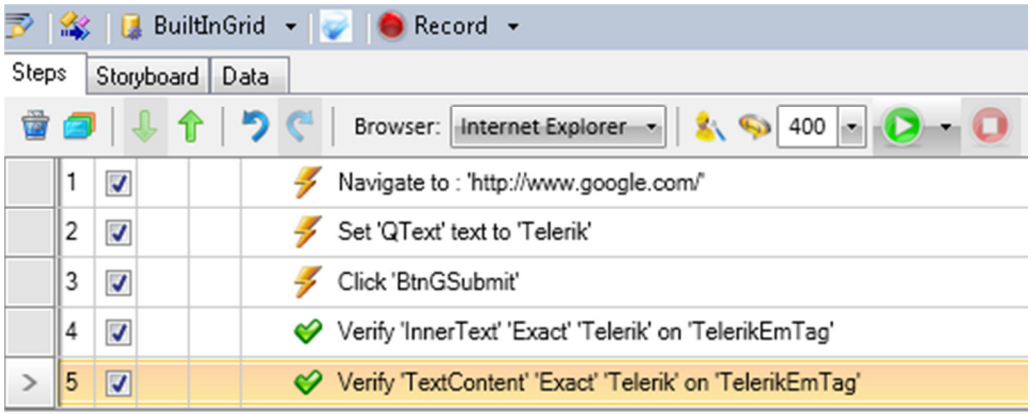
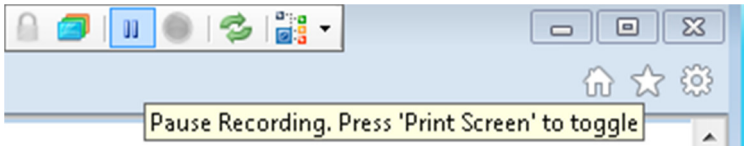


Figure 4 - VS plugin

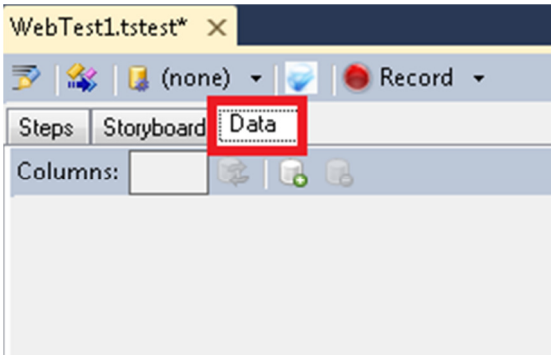
Create a Data Driven Test

Let's create a new, Data Driven test. We'll go through five iterations of the test, each with a different search text.

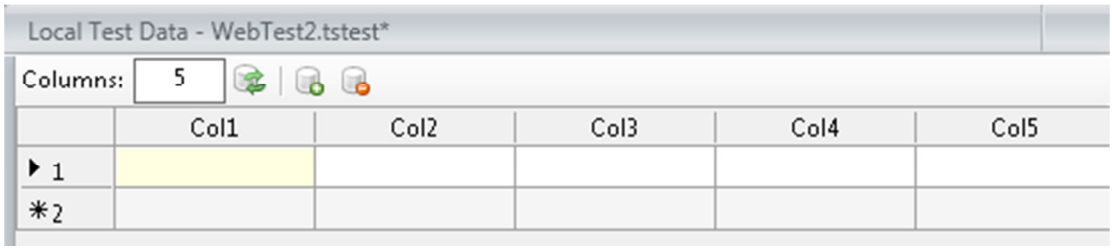
1. Create a new test as outlined above.
2. Select Web Test and alter Test Name if desired. Click OK.
3. Open the new test and click "Record."
4. Navigate to www.google.com, enter "Telerik" in the Google search box, and hit the Search button.
5. Click the "Pause" button in the docked IE taskbar.



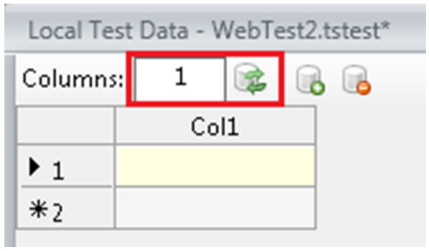
6. Click the "Local Data" button in the Test Views ribbon or the Data tab in Visual Studio.



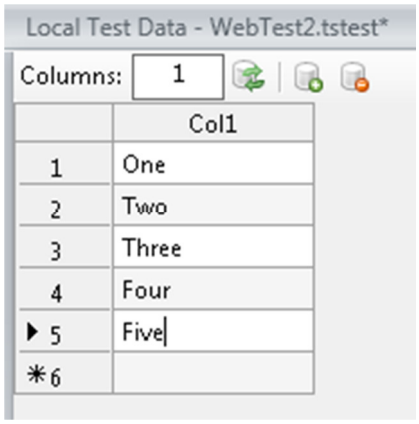
7. There are three buttons at the top of this pane. Only "Create a new data table" is enabled. Click it to add a new grid for data.



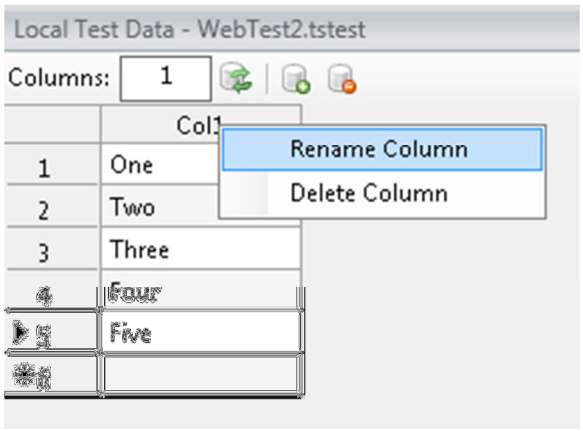
8. The default grid will have five columns. This example will execute five iterations of the test with different search text for each.
9. Change the columns text box to "1" and click "Update."



10. Enter any text into the first grid cell and hit Enter or Tab. The input will move to the second row.
11. Continue entering text for the remaining grid cells. New rows are added as you type.

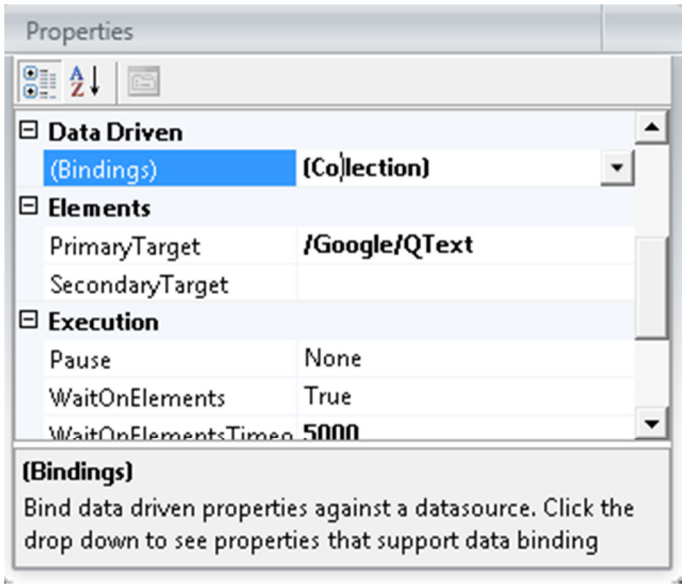


12. Right click “Col1” and choose “Rename Column.” In this example, “Numbers” is the new name.

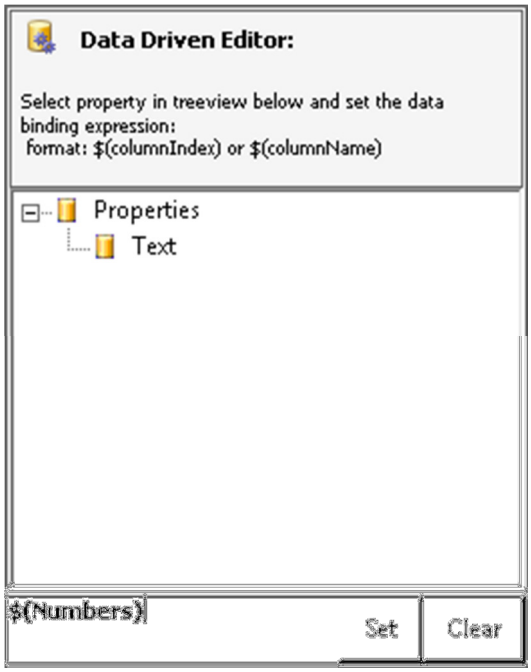


13. Save the test.
14. Data from an array can be used in recorded steps and code behind methods. To bind data from a data array to a recorded step, continue with the following steps. To use reference data from the data array in a code behind method, skip to step 24.
15. Press the F4 key to open the Properties pane.
16. Highlight the recorded step that sets the value of the Google search text box (test step 2 in this example).

17. The properties for this step will appear in the Properties pane.



18. Click the drop-down arrow for "(Bindings)."
19. Click on the "Text" node in the displayed tree.
20. Enter the name of the column you want to draw data from into the text box. In our example we want the first and only column, so we'll enter "$(Numbers)."



21. Click the Set button.
22. The data for the column named "Numbers" from the data array is now bound to the Text property for that step. Instead of entering "Telerik" into the search box, the data stored in the data array will be entered.
23. Save and execute the test. Note that the test will execute for each row in the data array.
24. To use reference data from the data array in a code behind method, follow these steps.

25. Locate the step that sets the value of the Google search text box (test step 2 in this example). Right click the step and select “Customize Step in Code.”
26. Choose Visual Basic or C# and click OK.
27. Use the “Data” property followed by the index of the column to reference data from the grid. For example:

```
// You can reference the column by index
Pages.Google.QText.Text = (string)Data[0];
// Or by name
Pages.Google.QText.Text = (string)Data["Numbers"];
```

28. Save and build the project.
29. Execute your test. Note that the test will be executed for each row in the data array.
30. If the test executes too quickly to validate visually, click the “Enable Annotation” button and set the delay in milliseconds before executing. These are located in the Quick Execution ribbon.

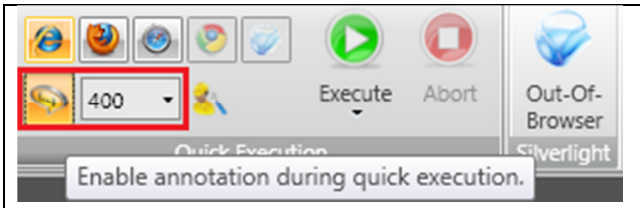


Figure 5 - Standalone version

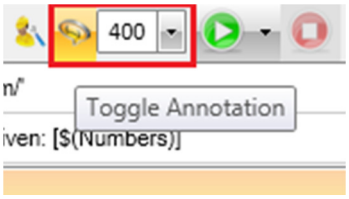


Figure 6 - VS plugin

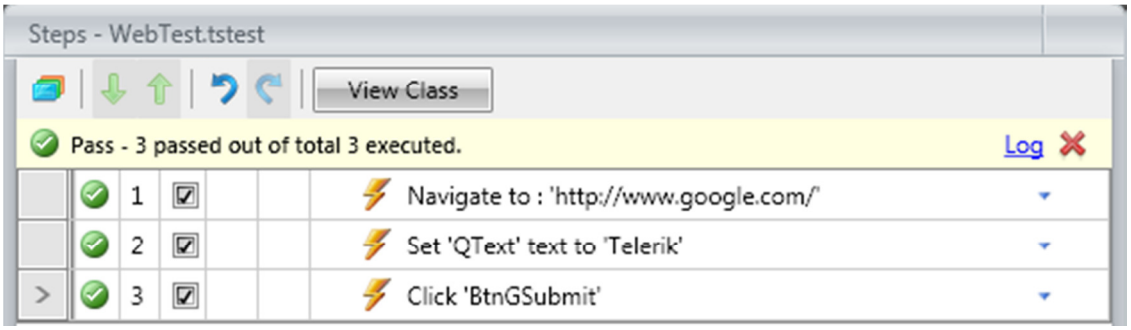
Test Execution (Standalone)

After recording your steps, it is time to execute your test. You have two options: Quick Execute or as part of a Test List.

The first method is Quick Execute.

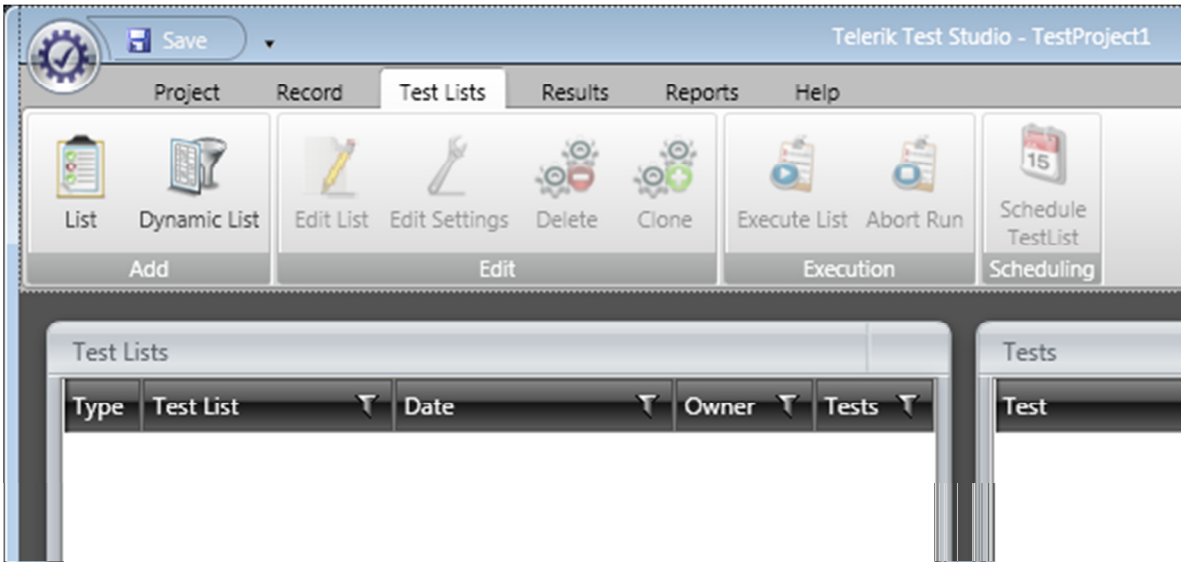
1. Click the Execute button.

2. Afterwards, test results are automatically displayed. Click “Log” for more information.

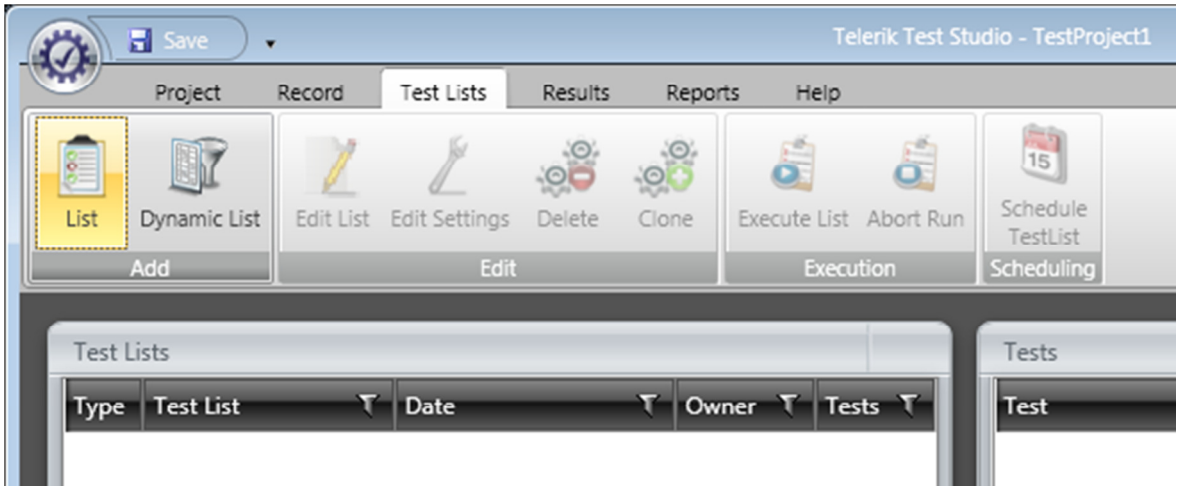


The second method is through a test list. Add the test to a list and then execute the test list.

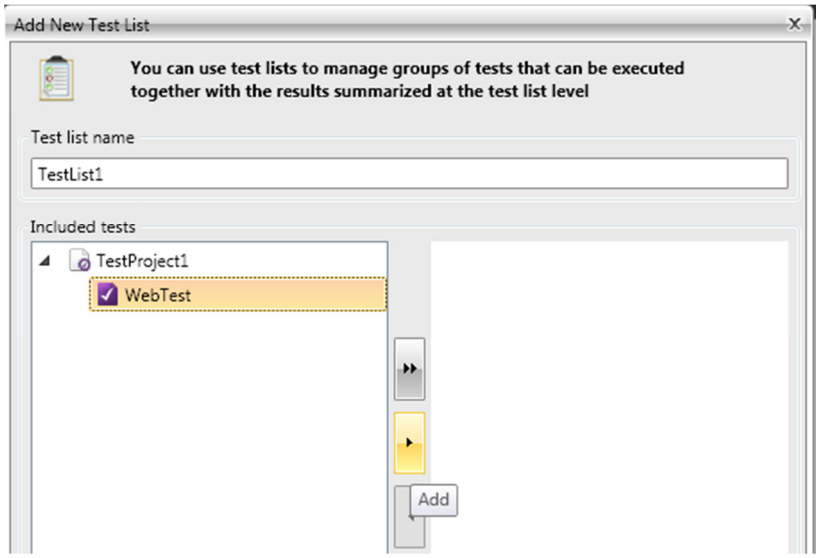
1. Click the Test Lists tab:



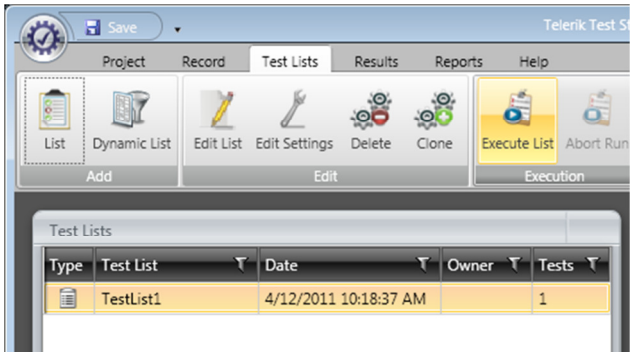
2. Click “List” in the Add ribbon.



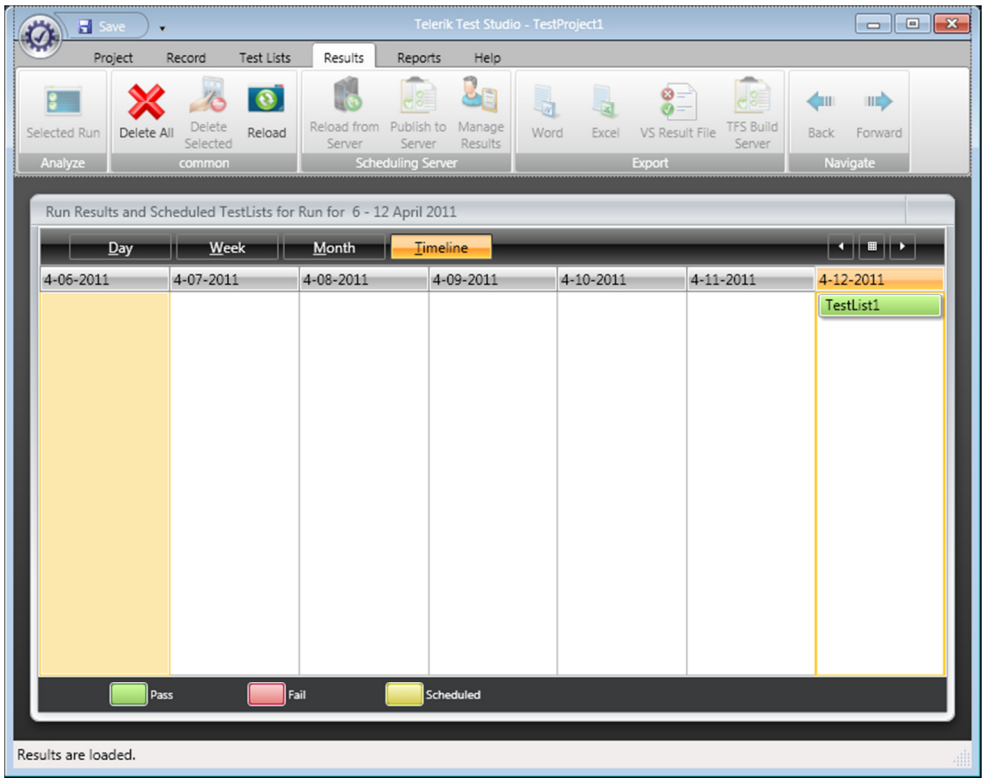
3. Give the test a name.
4. Add the test to the list on the right.



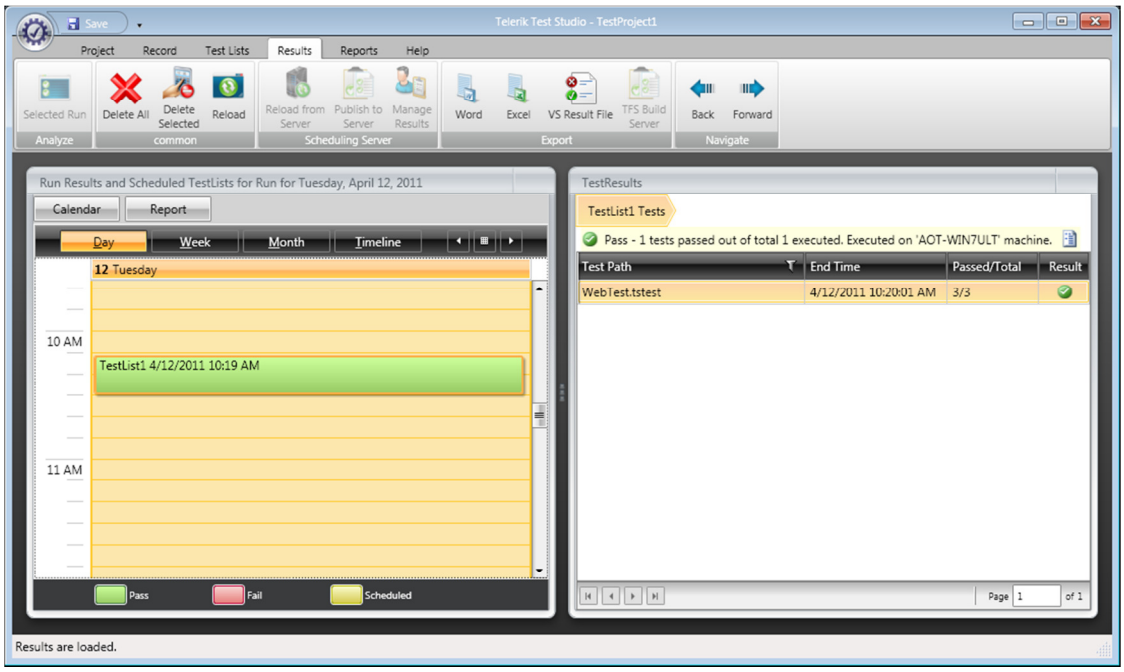
5. Click OK to save the new test list.
6. Click “Execute List” in the Execution ribbon.



7. A browser window or WPF app opens and the test executes. Upon completion, the browser window closes and the Results tab opens.



8. To view the test results, double click the test result entry in the Timeline view ("TestList1" in this example).

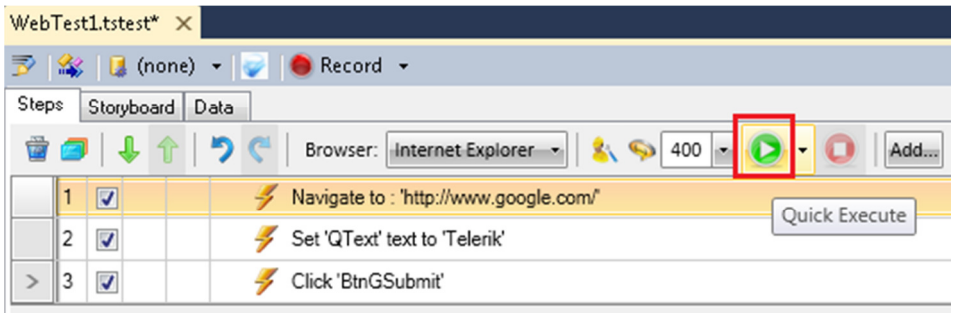


Test Execution (VS Plug-In)

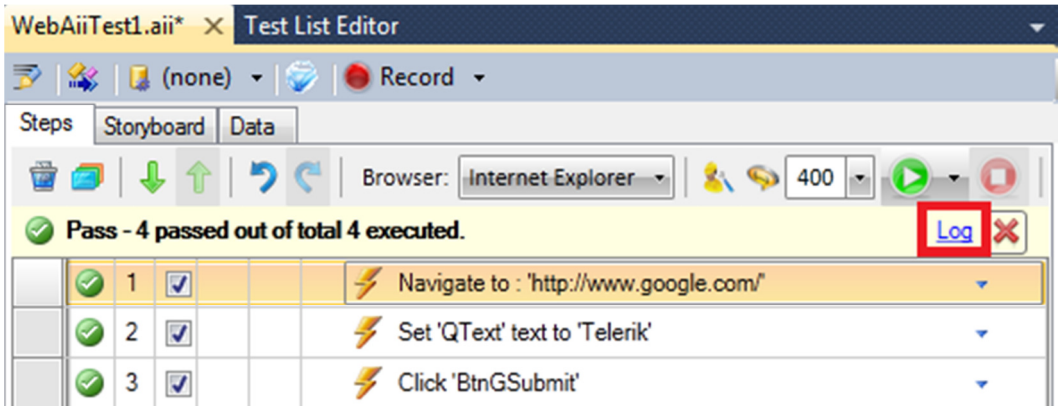
After recording your steps, it is time to execute your test. You have two options: Quick Execute or from Visual Studio Test View.

The first method is Quick Execute.

1. Click the Quick Execute button.

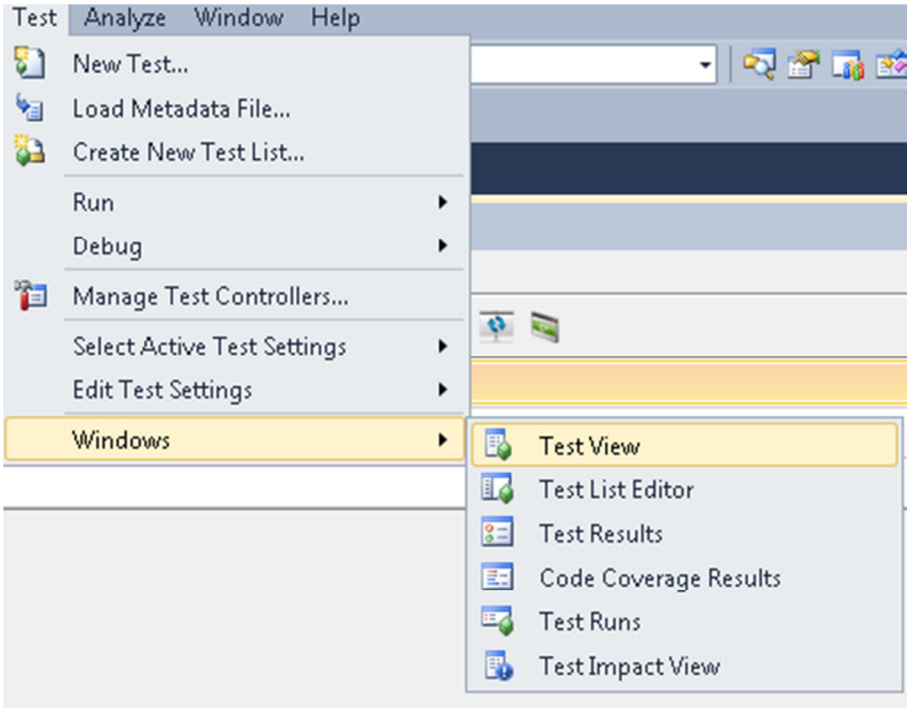


2. Afterwards, test results are automatically displayed. Click “Log” for more information.

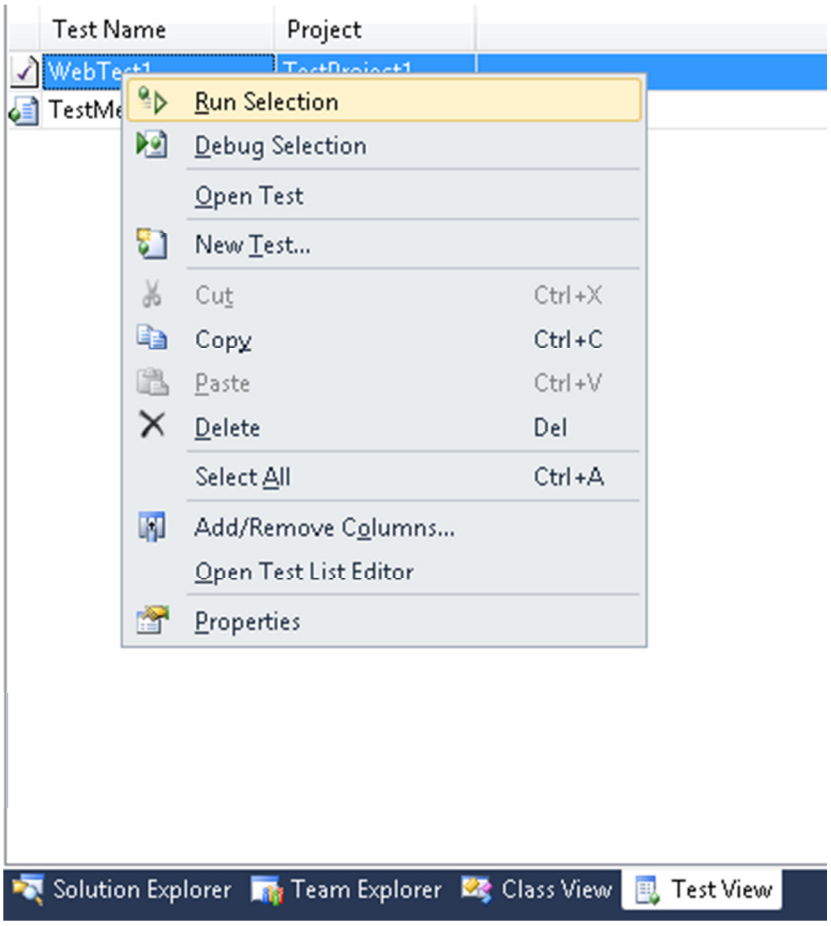


The second method is from the Visual Studio Test View, using the VS testing framework.

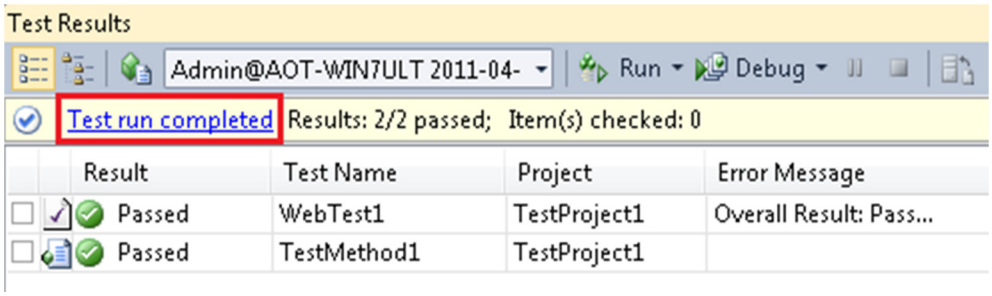
1. Ensure the Test View is visible. Click “Test”->”Windows”->”Test View.”



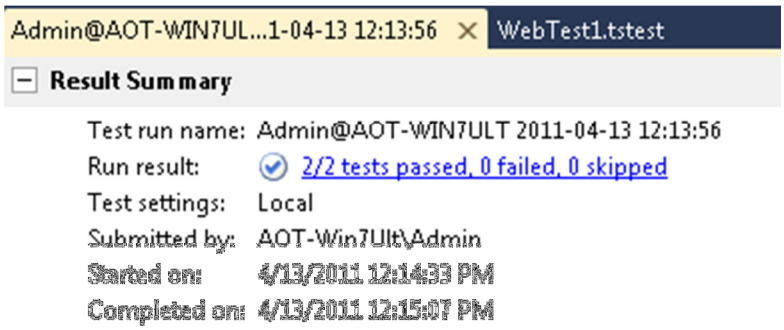
2. Right click on your test and click “Run Selection.”



3. A browser window or WPF app will open and the steps of your test will automatically execute.



4. After the test runs, click the link (in red above) in the "Test Results" tool window to view the execution log.



How to Resolve Test Step Failures

After running Quick Execution for a test, any failed steps will be marked with a red icon and the overall test will fail.

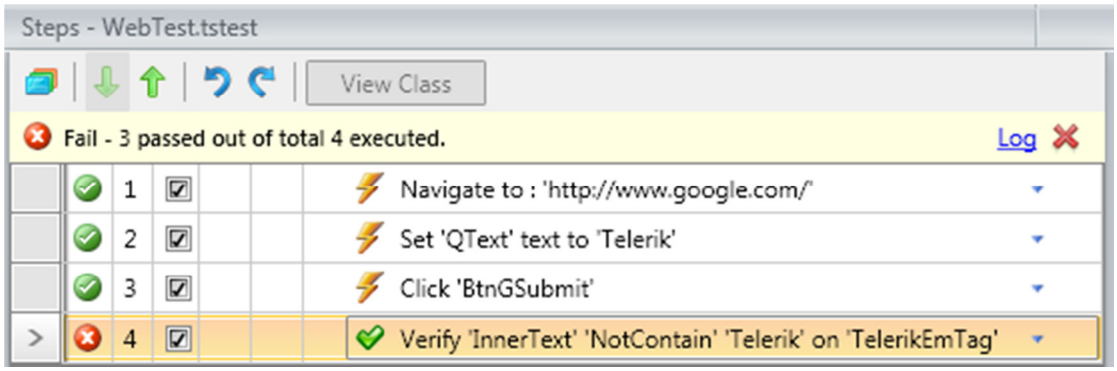


Figure 7 - Standalone version

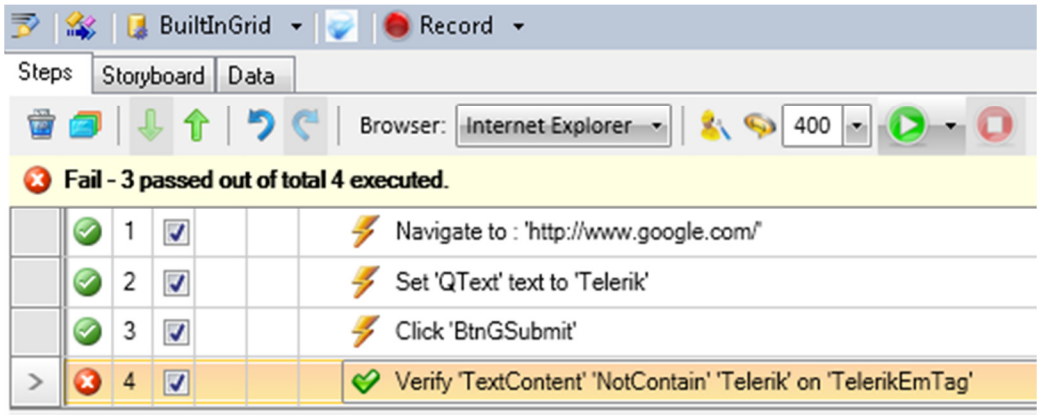
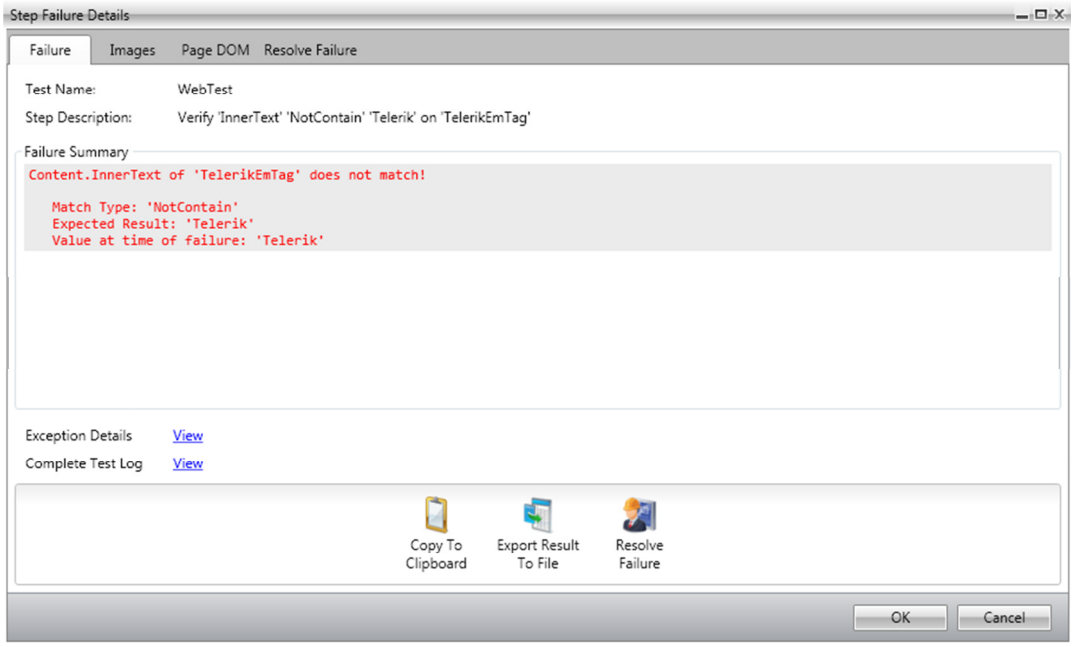


Figure 8 - VS plugin

1. Double click the X icon to launch the Step Failure Details UI.

2. The Step Failure Details UI gives a failure summary, exception details, and a complete test log.



3. The Failure tab gives a summary. The Images tab shows the browser image at the time of failure versus the expected image. The Page DOM tab shows the page's object model at the time of failure. And finally, the Resolve Failure tab allows you to change Selected Sentences and reverify.
4. Upon failure, decide if the test needs updating or the tested page, Web application, or WPF application has a defect. If the test needs updating, 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.

Elements Pane Overview

The Elements pane maintains a list of all Elements within the current project. It provides a one-stop shop to view elements and edit the way they are found during execution. The Elements menu bar has two buttons:

a. Enable/Disable Highlighting - this icon on the left controls highlighting of elements on the recording surface as they are selected in the tree view. Click this button to turn on element highlighting, click it again to turn it off.
b. Refresh - click this icon to refresh the display of the elements in Explorer. You seldom should have to do this because Test Studio normally refreshes the window properly.

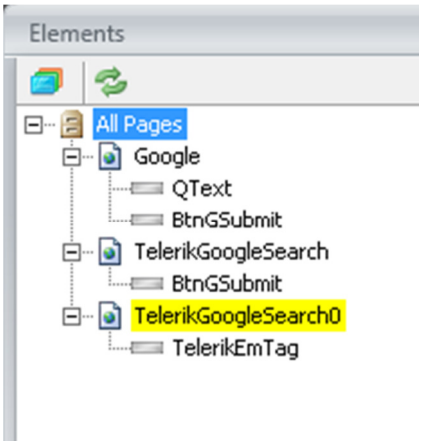


Figure 9 - HTML Tree View

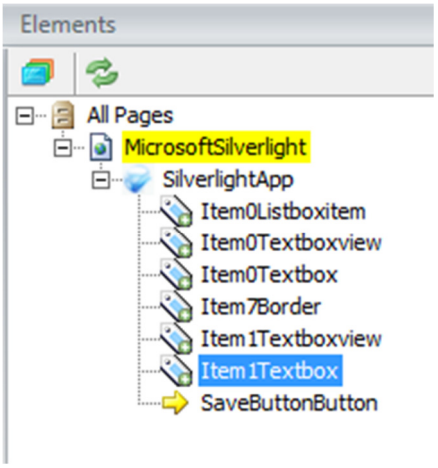


Figure 10 - Silverlight Tree View

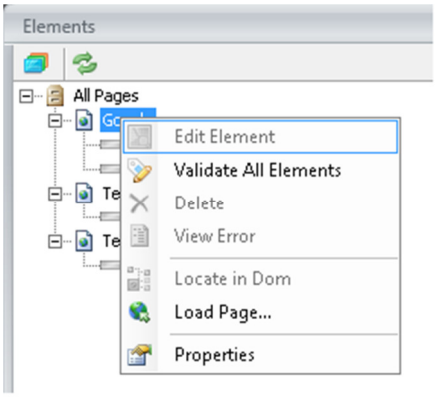
The HTML tree view is organized by Page -> Frame -> Test Regions -> Elements.

The Silverlight tree view is organized by Page->Frame->SilverlightApp->Elements.

The hierarchy is maintained according to where the element is 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.

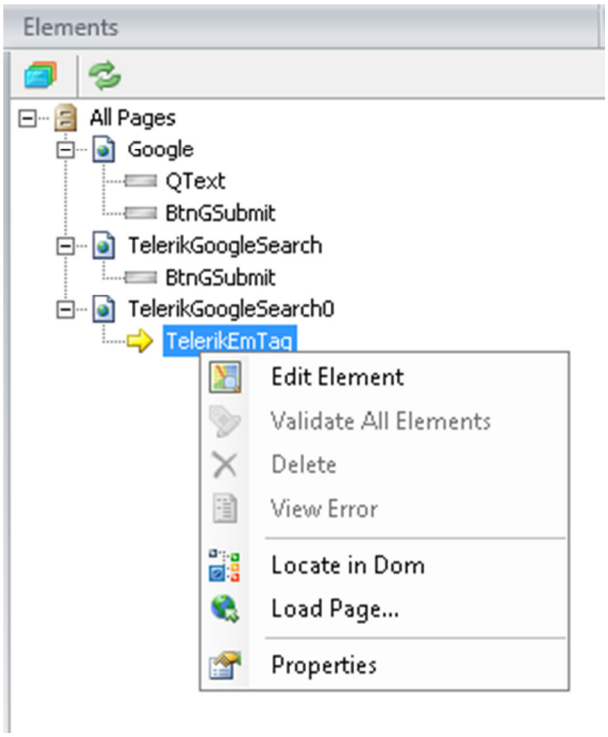
Right click each Page node to see a context menu with the following:

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



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

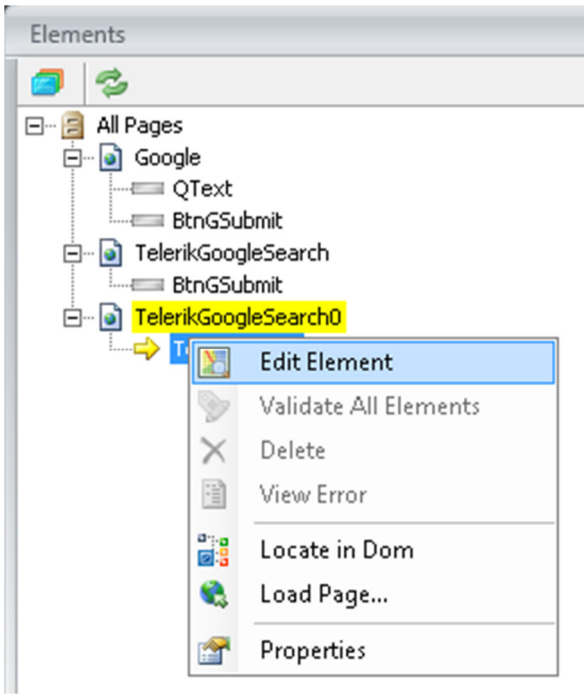
a. Edit Element – loads the Find Expression Builder.
b. Locate in DOM – highlights the element node in the DOM Explorer tool window.
c. Load Page – loads the URL to which the element belongs in the recording window.
d. Properties – makes the Properties pane active.



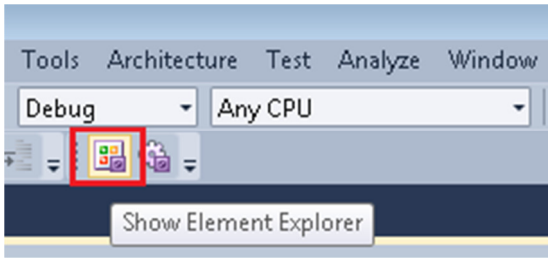
Change How an Element Is Found

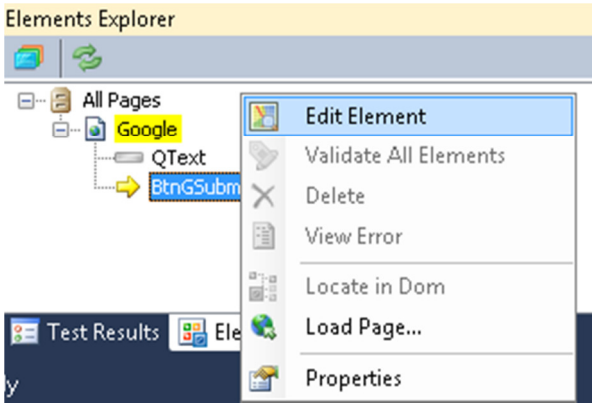
When a web page element has an action recorded against it, or you explicitly add an element to the "Elements" pane, a "Find Expression" is generated that tells the framework how to find that specific element on the web page.

1. To change how an element is found, right click on the element in the Explorer and select "Edit Element."

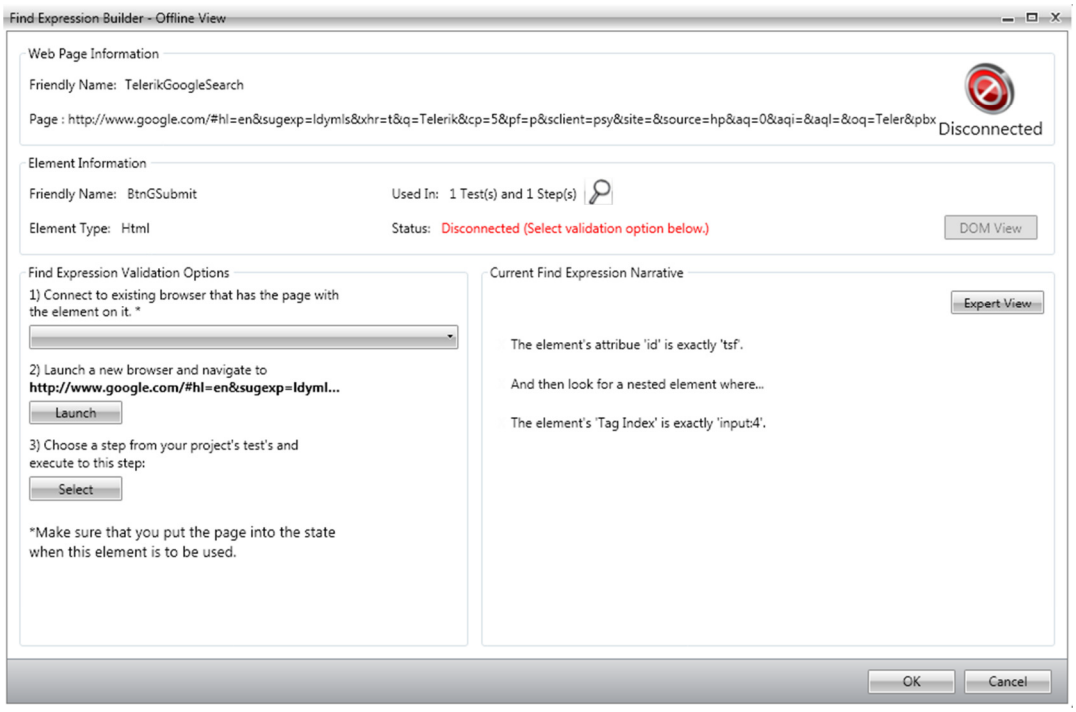


In the VS Plug-In, click the "Show Element Explorer" icon in the toolbar (in red below) and locate the Explorer at the bottom of the screen.





2. The "Find Expression Builder" launches and displays how the element is currently found.



3. The “Find Expression Builder” consists of the following:
    a. Web Page Information – Friendly Name and Page URL.
    b. Element Information – Friendly Name, the number of tests and steps used in, Element type, Status, and DOM View.
    c. Find Expression Validation Options – three ways to connect to the browser and validate the find expression.
        i. Connect to existing browser that has the page with the element on it. Click the drop-down to see open instances of IE.
        ii. Launch a new browser and navigate to Web Page URL. Use this is IE is not open.
        iii. Choose a step from the projects’ tests and execute to this step.
    d. Current Find Expression Narrative – represents the find expression with a simple statement. Edit by hovering over the applicable words or phrases and click when the white pen appears. Delete a statement by clicking the “X” at the beginning of it.
        i. Element Attribute – the property or attribute of the element to examine. HTML element examples include TextContent, NodeIndexPath, TagName, TagIndex, and XPath. HTML attribute examples include name, id, and visibility. XAML element examples include TextContent, XamlTag, Name, and TagIndex. XAML attribute examples include Foreground and Content.
        ii. Change the compare type via the drop-down menu (e.g. Exact, Contains, StartsWith, etc.).
        iii. Value to look for – can be a string, number, index path (tag or XAML index), or a RegEx expression.
        iv. And/Then – Use multiple criteria for locating the correct element. “And” means the current line and the next line must be to true. “Then” equals a chained find

expression. Specify how to find element ‘A,’ and then underneath element ‘A’ find element ‘B’. This is useful for pages in which multiple custom complex controls exist in different panels. Use a chained find expression to locate the correct panel, and then find the correct element (such as a button) within that panel.

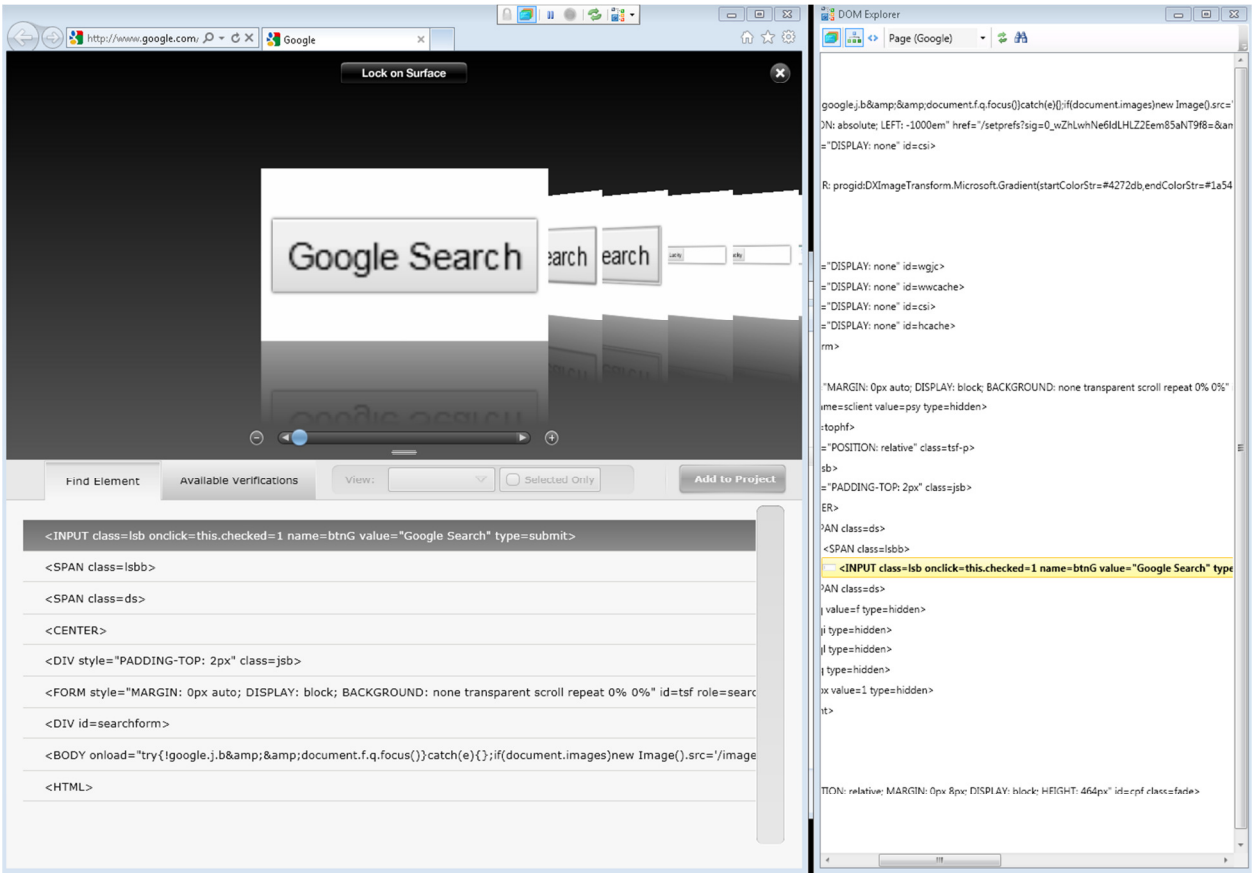
v. Switch to “Expert View” to validate.

4. After specifying the method and criteria used to find the element, click “Expert View” and then “Validate.” This validates that the 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.

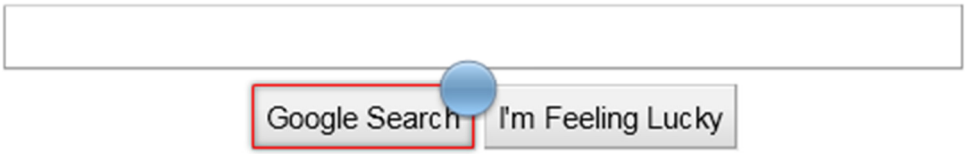
Use the 3D Viewer

The 3D viewer shows a hierarchical view of the elements contained in your application. It starts at a selected element, and then traverses the DOM tree up to the DOM root. It is used to identify/lock on elements and quickly build verifications.



There are two ways to open the 3D viewer:

1. Use the pop-out menu.
    a. In the recording browser window, enable hover over highlighting and place the mouse cursor over an element to examine in the 3D viewer.
    b. Wait for the blue circular nub to appear.

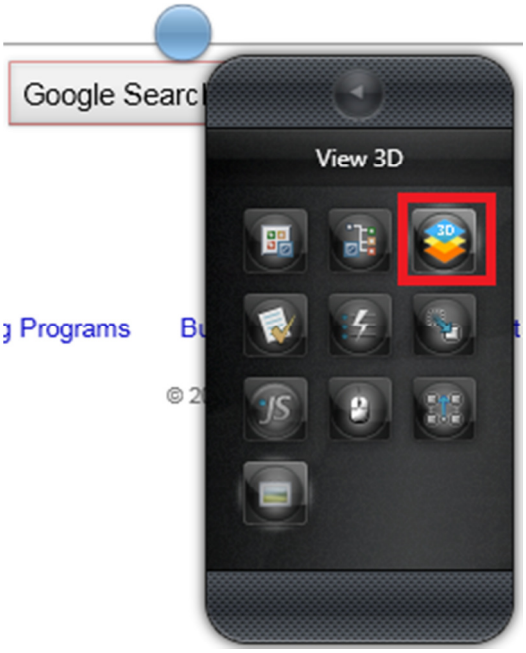


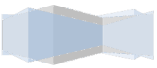
    c. Drag this nub. It changes into an image of the highlighted element and a pop out panel appears on the left side of the screen with three buttons in it. Drop the image of the element onto the middle button (3D viewer). The 3D viewer will open and the dragged

element is selected.

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

2. Use the Element Menu.
   a. Perform steps A & B above.
   b. Click the blue nub. The Element Menu opens.
   c. Click the View 3D button.

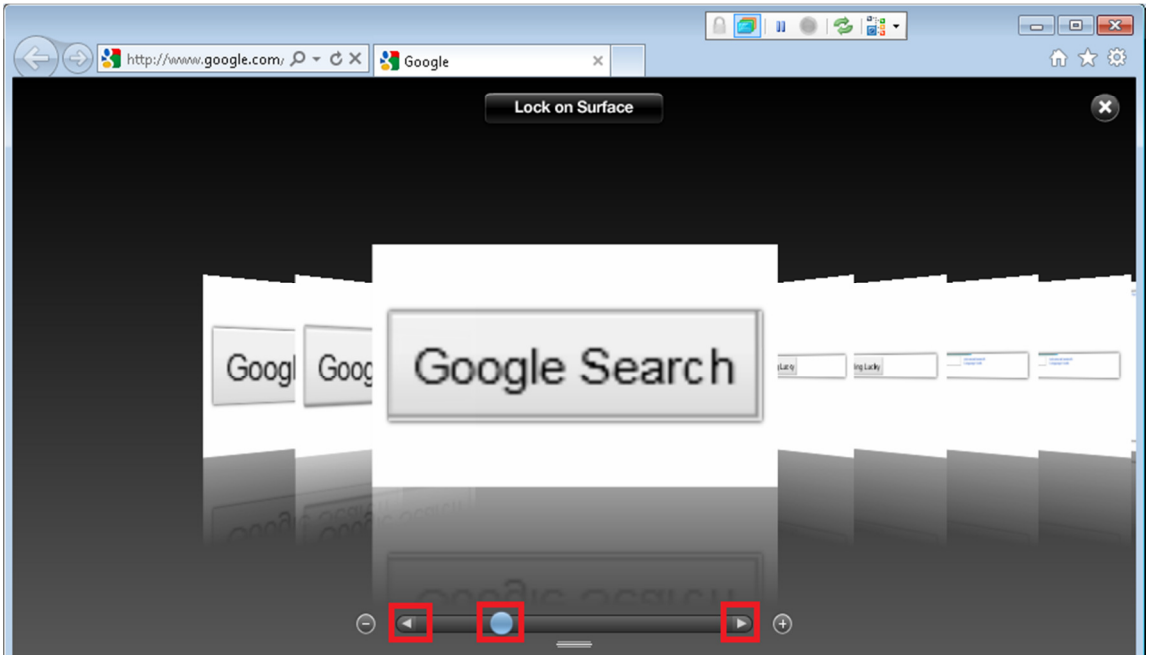




d. The 3D viewer opens and the specified element is selected in the viewer.

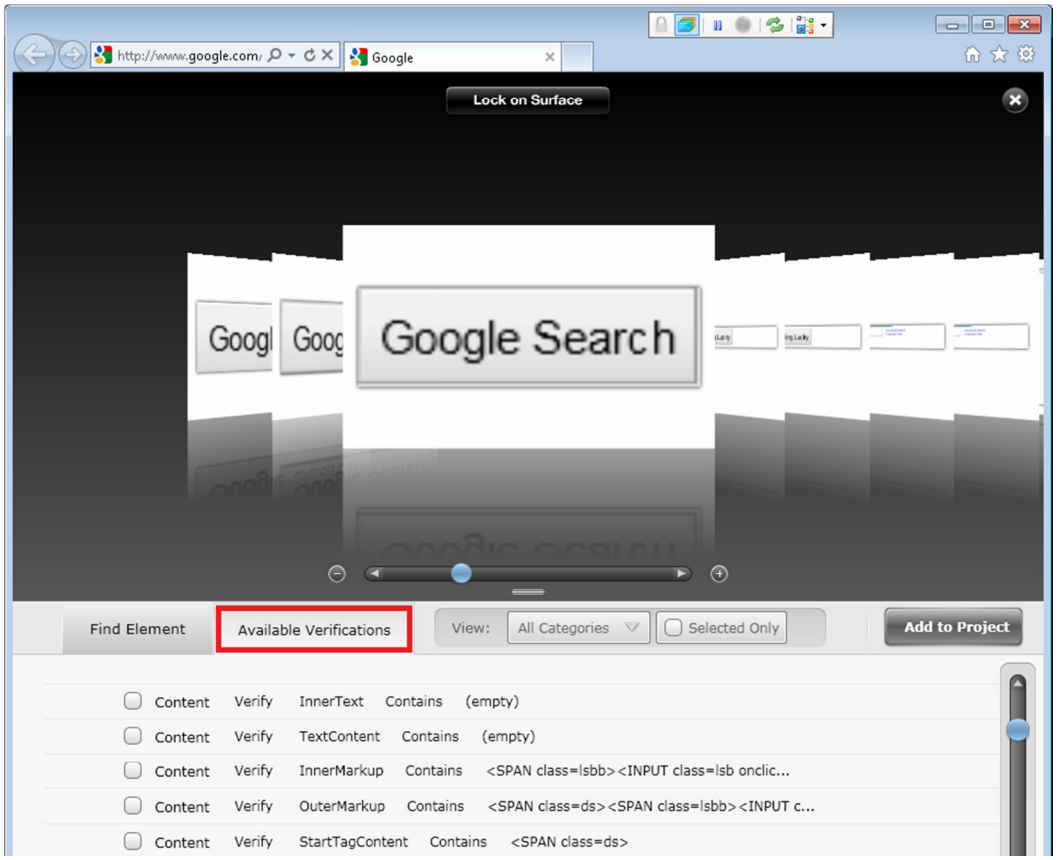
Once the 3D viewer is open, perform the following:

3. 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).
4. Increase/decrease magnification of the screenshot by clicking the - or + symbol (shown above).
5. Click the “Lock on Surface” button to:
    a. Close the 3D viewer and return to the IE recording window.
    b. Highlight that element.
    c. Open the Element Menu.

6. Click the “Available Verifications” tab to display a list of prebuilt verifications.



7. The View group boxes control which of the available verifications to display. Change the displayed category with the Categories drop-down. Limit the view to only selected verifications by ticking the “Selected Only” checkbox.
8. Quickly add verifications by ticking the checkbox(es) for the desired verification(s) and then clicking “Add to Project.”
9. When a Verification is highlighted, two buttons appear next to it. The first button (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.

10. 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.”

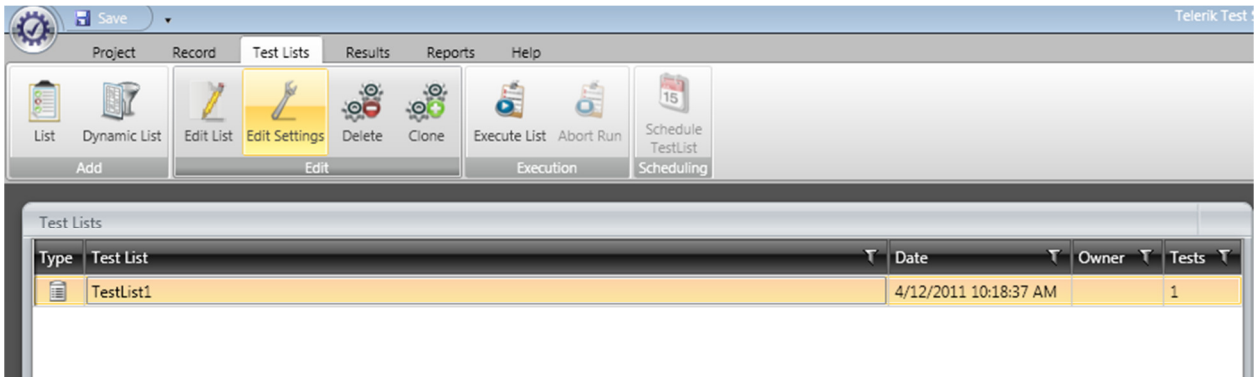
Perform Common Automation Tasks

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

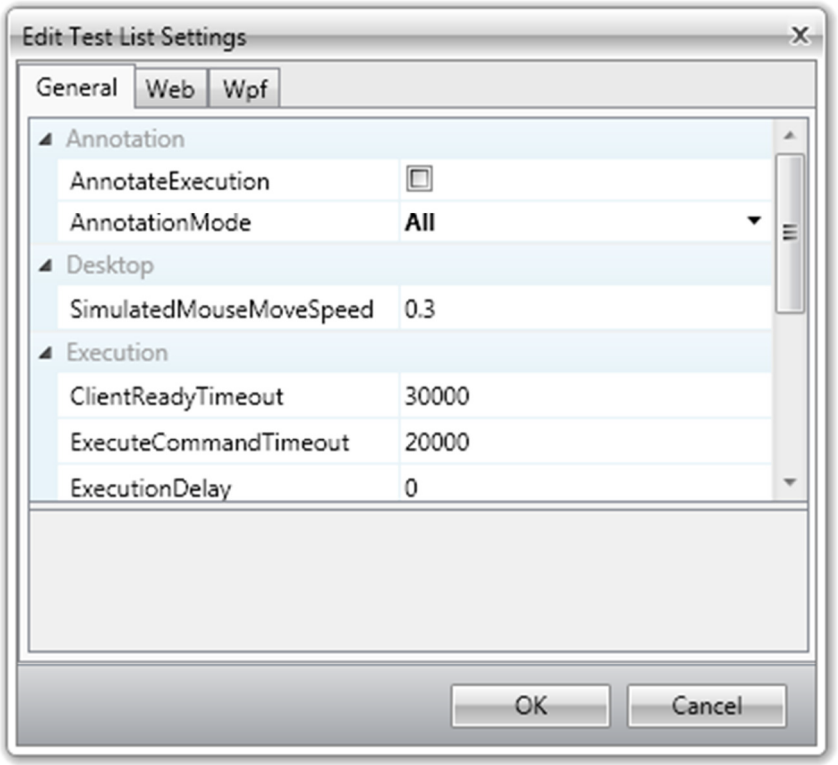
Telerik Test Run Configuration (Standalone)

To configure Telerik Framework settings for execution:

1. Click the Test Lists tab.
2. Highlight a test list to configure by clicking on it in the Test Lists pane on the left.
3. Click “Edit Settings” in the Edit ribbon.



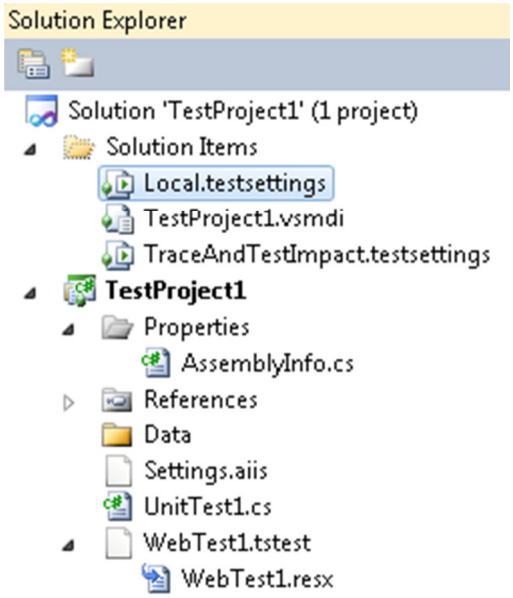
4. This opens a dialog box that shows all of the possible configuration settings for that test list. Change the settings as needed for your test list.



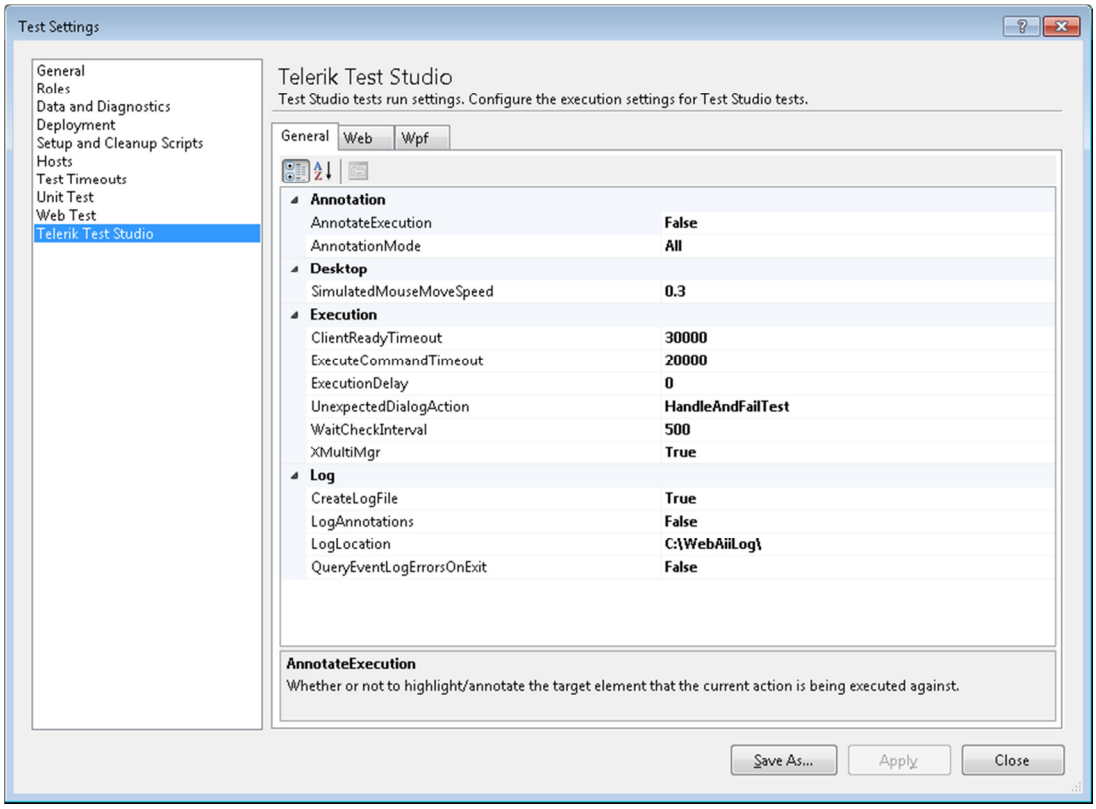
Telerik Test Run Configuration (VS Plug-In)

To configure Telerik Framework settings for execution:

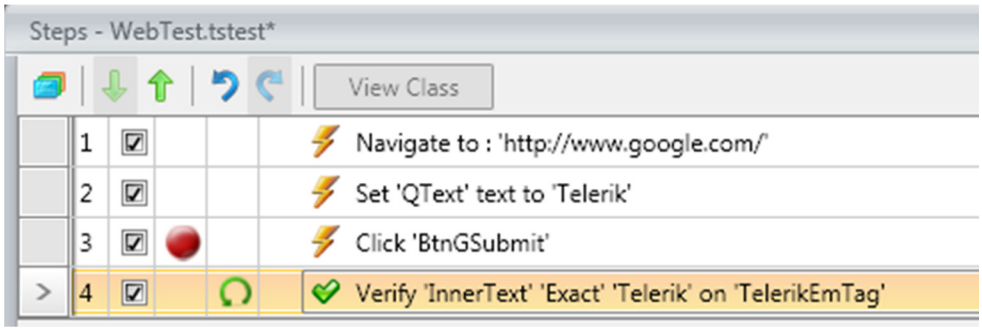
1. Double click “Local.testsettings” under “Solution Items” in the Solution Explorer.



2. On the left side, click “Telerik Test Studio.”
3. This gives you a page with three tabs to configure the run time configuration settings for all test executions within this project.



Steps Pane Overview (Standalone)



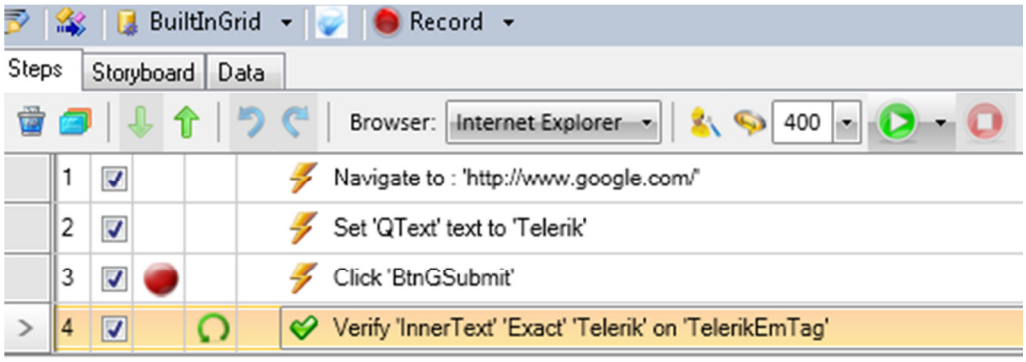
The Steps pane provides the list of steps in the currently selected test. Each step has:

a. Order of the Step.
b. Enabled checkbox – whether the step will run during execution.
c. Breakpoint indicator – whether the test will pause at that step and show Debug Options.
d. Continue on failure indicator – whether the test will stop if that step detects a failure.
e. Type Icon – represents the type of step: Action, Verification, Coded, etc.
f. Step description.
g. Delete Button – deletes the step from the test.

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

a. Toggle Highlighting – when enabled, the element of the selected step is highlighted in the recording surface.
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/changed to its previous state and location.
e. Redo – reapplies the edit that was undone with the Undo button.
f. Class View –switches the view between the code for the currently selected coded step and the entire code behind file (if the test has a code behind file).

Steps Pane Overview (VS Plug-In)



The Steps pane provides the list of steps in the currently selected test. Each step has:

a. Order of the Step.
b. Enabled checkbox – whether the step will run during execution.
c. Breakpoint indicator – whether the test will pause at that step and show Debug Options.
d. Continue on failure indicator – whether the test will stop if that step detects a failure.
e. Type Icon – represents the type of step: Action, Verification, Coded, etc.
f. Step description.
g. Delete Button – deletes the step from the test.

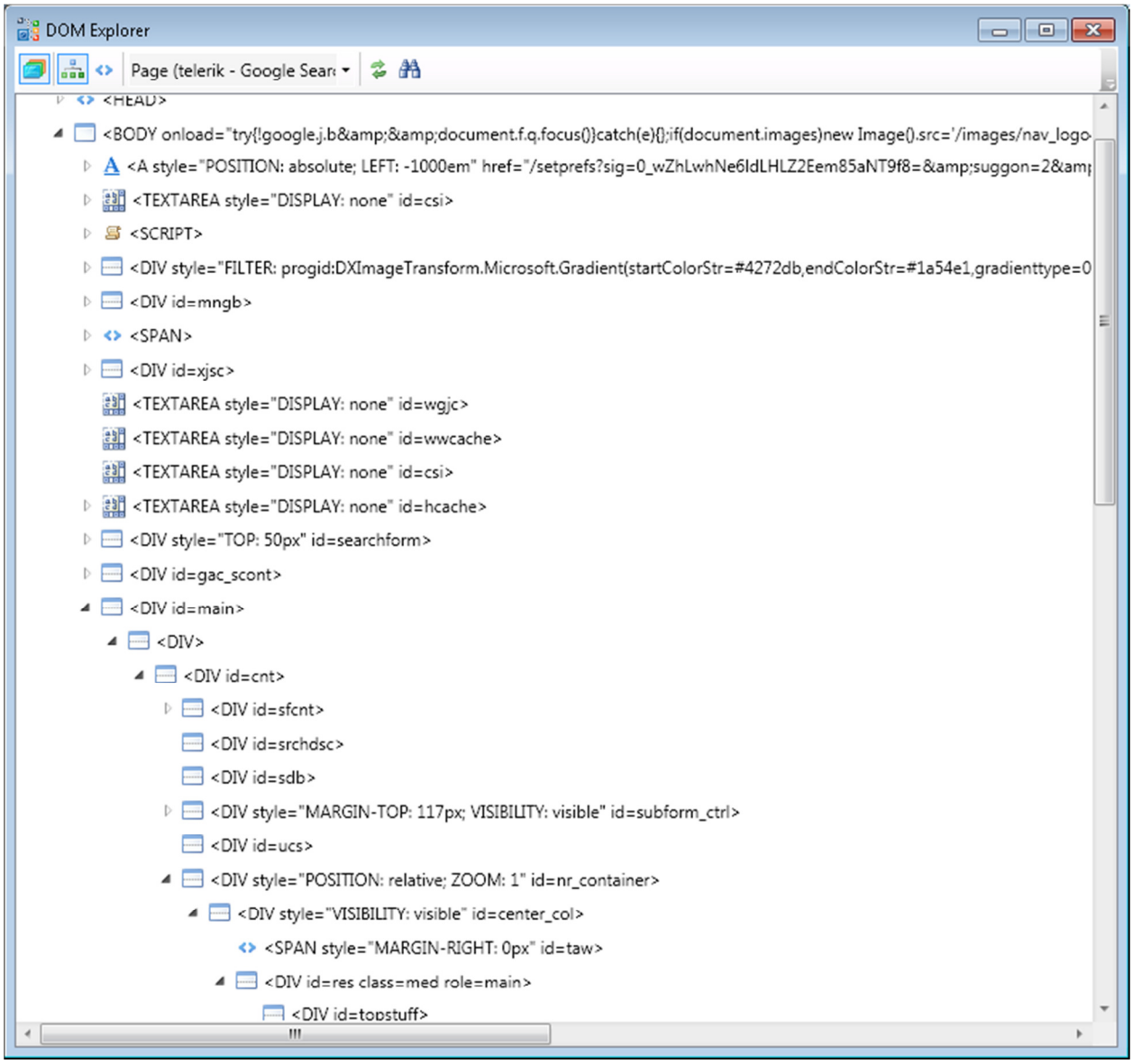
The menu bar for the Steps tab contains (from left to right):

a. Clear All Steps – deletes all steps from test.
b. Toggle Highlighting – when enabled, the element of the selected step is highlighted in the recording surface.
c. Move Step Down – moves selected step down one in the sequence.
d. Move Step Up – moves selected step up one in the sequence.
e. Undo – restores a step you just deleted/changed to its previous state and location.
f. Redo – reapplies the edit that was undone with the Undo button.
g. Browser Drop-Down – selects which browser to use for Quick Execution.
h. Show Debugger Options.
i. Enable Test Annotation – enables test annotation during Quick Execution. To change the execution across all tests, change the setting in the test configuration as previously explained.
j. Execution Delay – sets the time (in milliseconds) to delay between each step.
k. Quick Execute – immediately executes the current test in the selected browser.
l. Stop Execution – halts a running test.
m. Add – This drop-down adds special steps that cannot be recorded. They 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 and stores it in a file.
    iii. Custom Annotation – inserts personalized annotations into the test.

iv. Test as Step – 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, resumes running the rest of the steps of this test.
v. Delay Execution – causes the test to pause for the specified number of milliseconds.
vi. Clear Cookies – clears browser cookies.
vii. Wait for URL – waits for a specific URL to appear in the navigation bar of the browser. This is useful for page redirection or form posting where there can be a processing delay by the web server.
viii. Inspection Point – pauses the test and opens a window with the current DOM. The test resumes when the DOM window is closed.
ix. Comment – adds a comment to the test.
x. Manual Step – adds a manual step to the test.

n. Dialog – add specific dialog handlers as test steps. The dialogs that can be handled are Alert, Confirm, Logon, File Upload, File Download, Logon, and a special Generic dialog handler, which simply closes most dialogs.
o. Logical – inserts conditional statements as test steps. These include if…else, while…loop, and loop.
p. Recapture Storyboard – recaptures screen shots of each test step if the web site or application changes.
q. Preview Code – displays code with the ability to copy to clipboard.

DOM Explorer Tool Window Overview



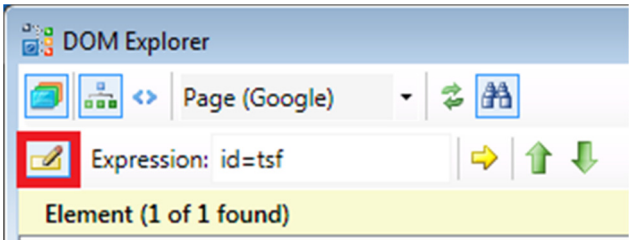
The DOM Explorer tool window displays the Document Object Model for the currently loaded page in the IE recording window or WPF app. The nodes are a hierarchical representation of all the HTML/XAML elements. Each node in the tree is listed by:

<[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 is highlighted in the IE recording window or WPF app by having a red rectangle drawn around it.
b. Hierarchal View – displays the DOM as a hierarchal tree in the order displayed.
c. TagName – displays the DOM sorted by tag names.

d. Move to main element drop-down – lists the root element, 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 – rebuilds the tree from the current page.
f. Search – enables another toolbar below the first. This second toolbar searches the DOM tree for elements within it.
    i. Standard search – does simple string search and match.
    ii. Search Using Find Expressions – click the pencil icon (in red below) to change the search method. Use the following syntax, referenced from the Expert View in the Find Expression builder: element attribute=string (see below for example).

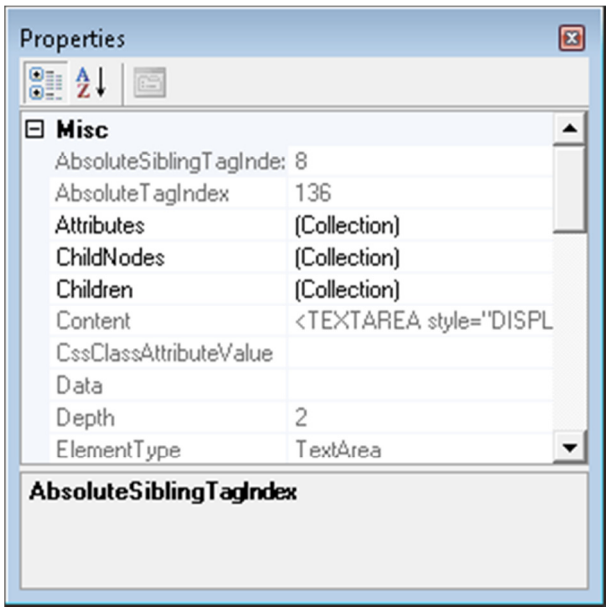


g. Perform Search – evaluates the find expression and highlights the first matching element in the DOM Explorer.
h. Move to Previous Result
i. Move to Next Result

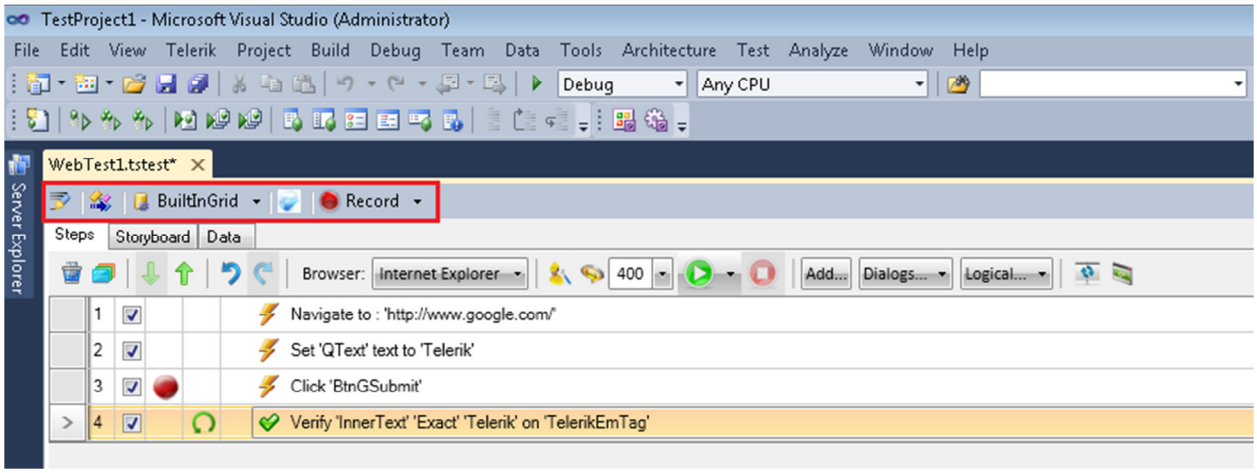
Right click a node to bring up a context menu containing these options:

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 – adds 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. Data is available to paste into another application or report.

e. Properties – opens a properties window showing detailed information about the selected element.



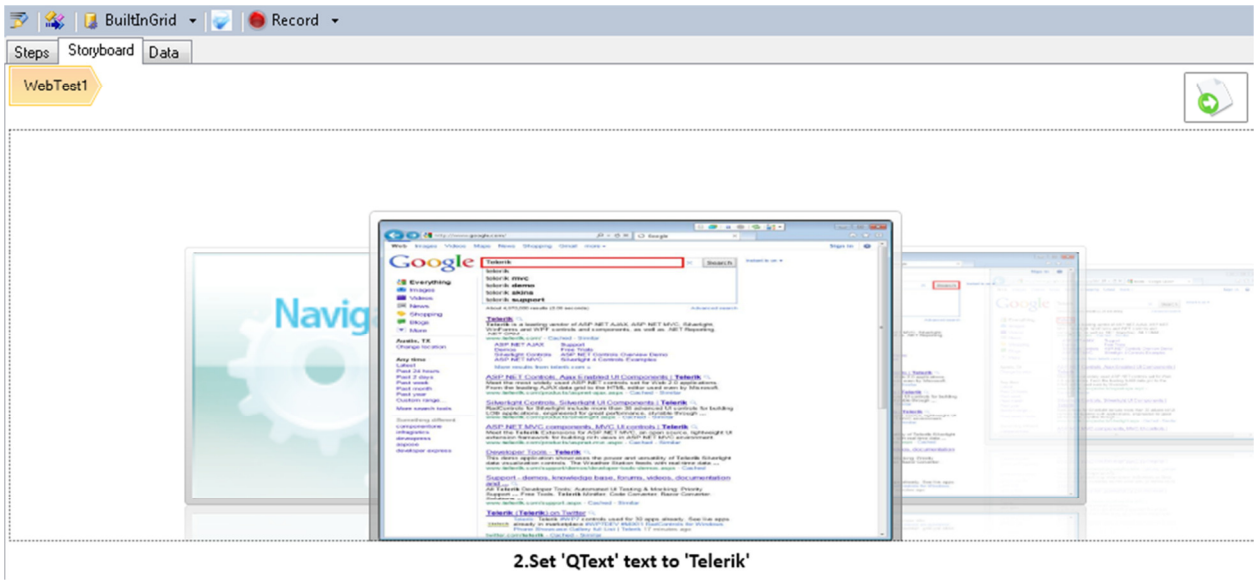
VS Plugin Telerik Test Tab Overview



Each Telerik test loaded in Visual Studio has its own document tab that allows for step visualization and test manipulation. The test tab has a menu bar with the following icons:

a. Add 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 all steps to code.
c. External data sources – a drop-down menu used to bind the test to an external data source for data driven testing. Use Excel, CSV, XML, or external database files as data sources.
d. Configure Silverlight App – loads Silverlight Out-of-Browser options.
e. Record – opens the Recorder window and directly enables recording for the test.

Note: With the Storyboard tab selected, some recorded steps will not have an image associated with them. This is due to the nature of the action. For example, the “Navigate To” action has no image.



2.Set 'QText' text to 'Telerik'

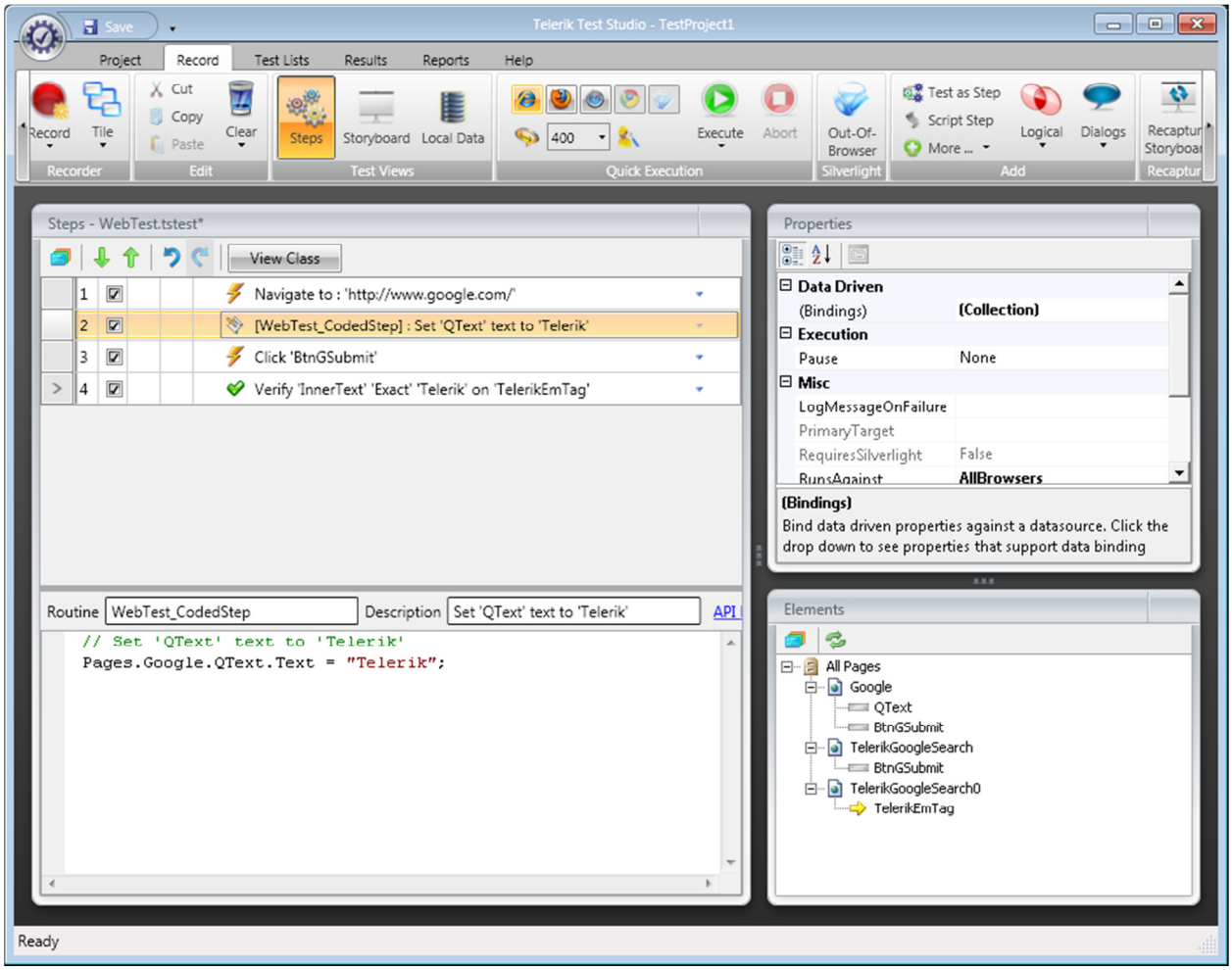
Click on any of the step images in the Storyboard tab to bring that image forward and move to that step on the Steps tab. Each recorded image has the element that it was recorded against highlighted within the image.

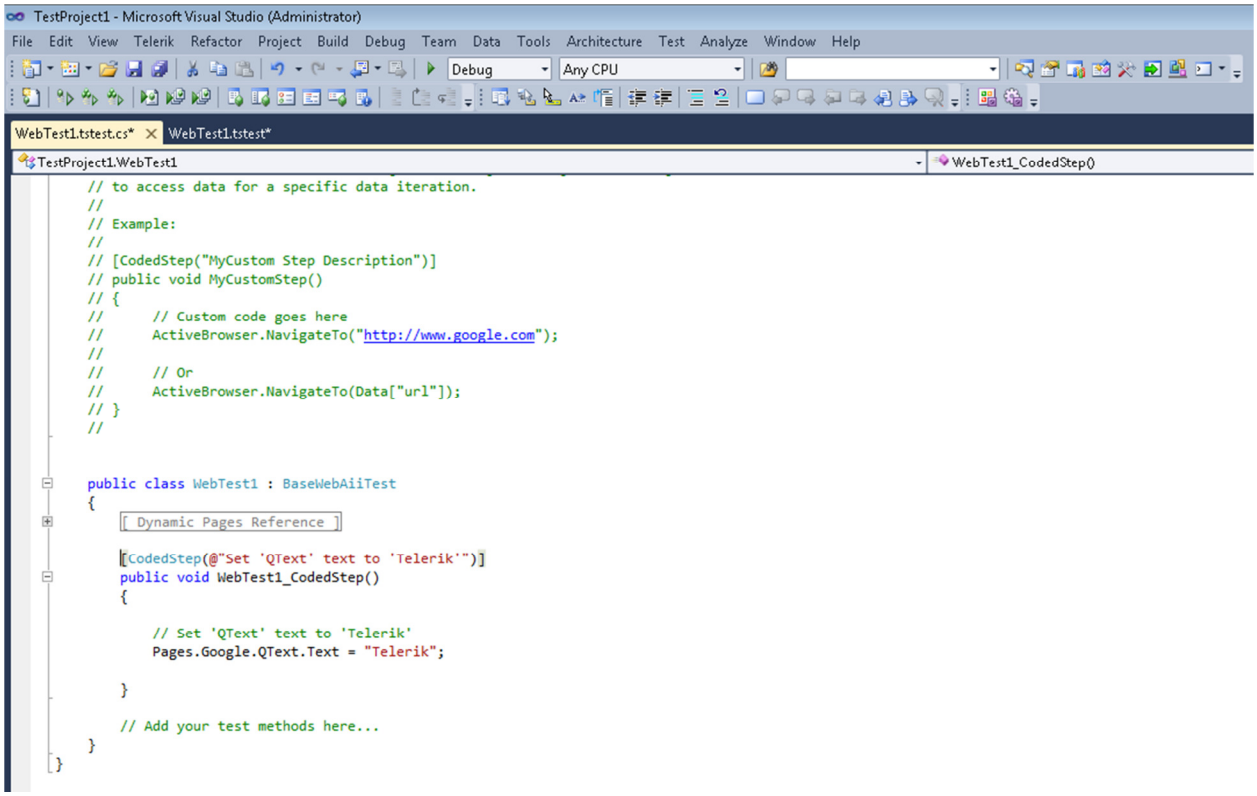
The test tab also contains a “Data” tab. It allows for the creation of a simple, Excel-like data array to be used by the test steps. See the “Creating a Data Driven Test” section for more details and an example.

Customize Test Using C# or VB.NET Code

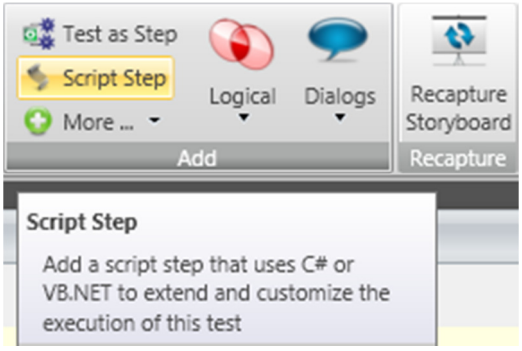
Telerik Test Studio supports code behind files. This allows you to write code and have it executed as a test step. It is for scenarios that require a test step that is more complex than what can be composed with the Verification Builder or the actions provided by the Element Menu.

Create a Test with a Custom Coded Step

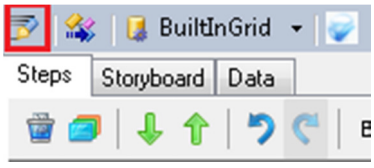




1. Create a test as outlined above.
2. Record a few steps as outlined above.
3. There are two ways of creating a code behind file for your test.
    a. Standalone – click the “Script Step” button in the Add ribbon.



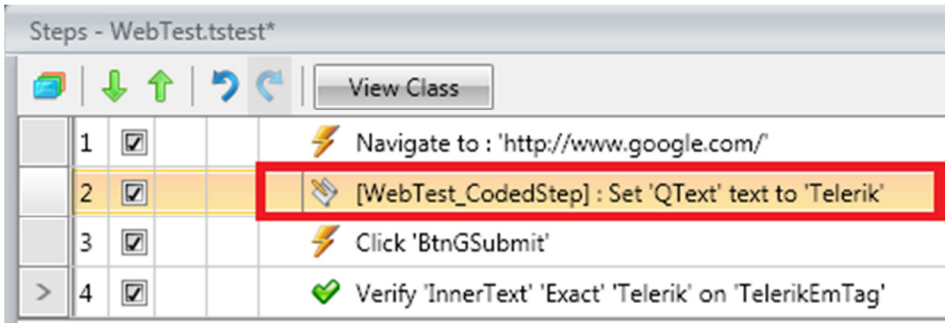
VS Plug-In – click the “Add code behind file” button in the test toolbar.

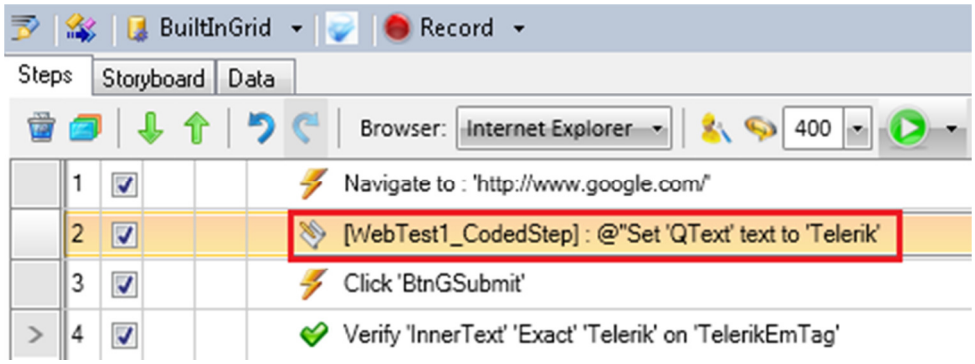


- OR -

    b. Right click on any step and select “Customize Step in Code.”
4. Step 3a creates a code behind file with an empty test method. Step 3b creates a code behind file with the selected step converted into a coded method matching the selected step name.

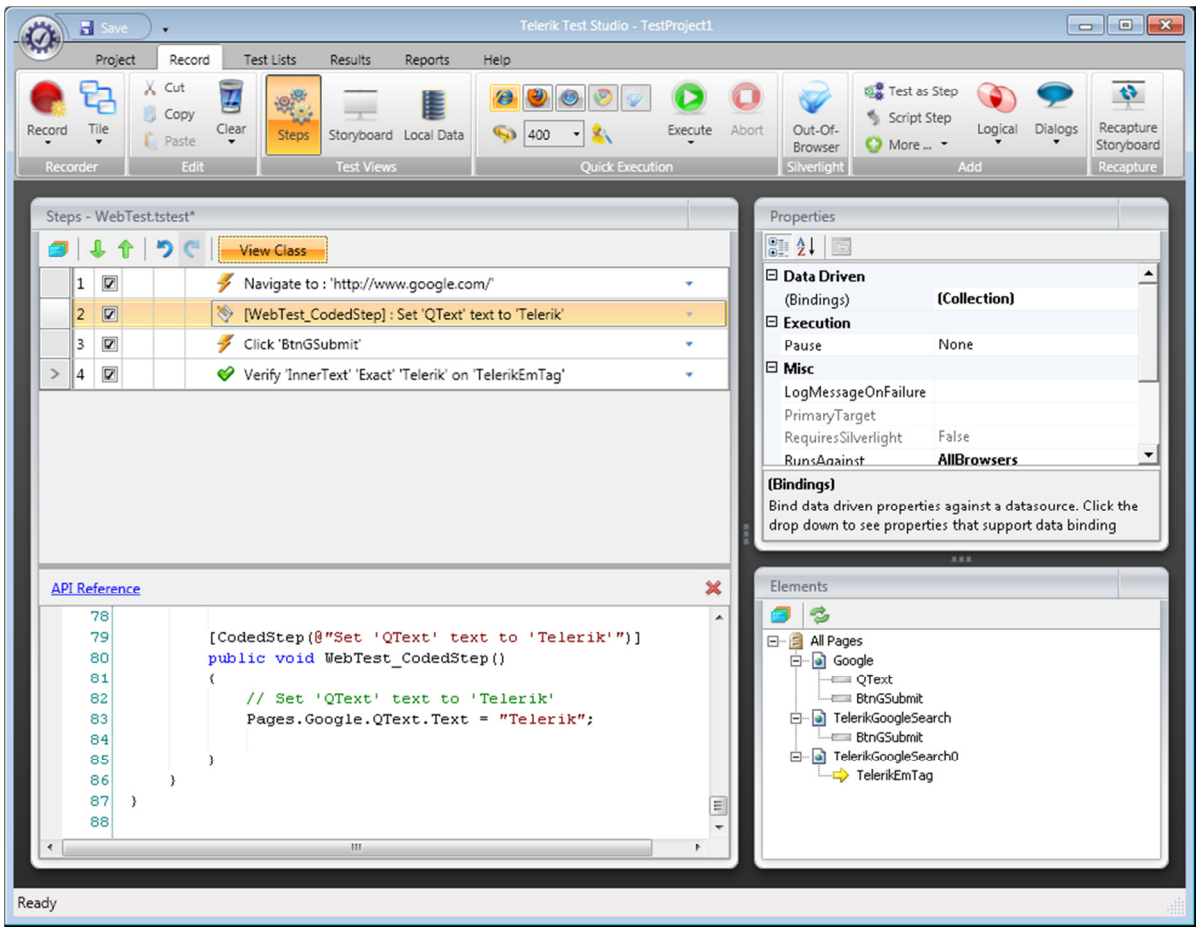
5. Note that a step with generated code is read-only in the Steps tab. The description can only be changed in the methods “CodedStep” attribute that appears in the code behind file. A coded step is indicated with a code icon.

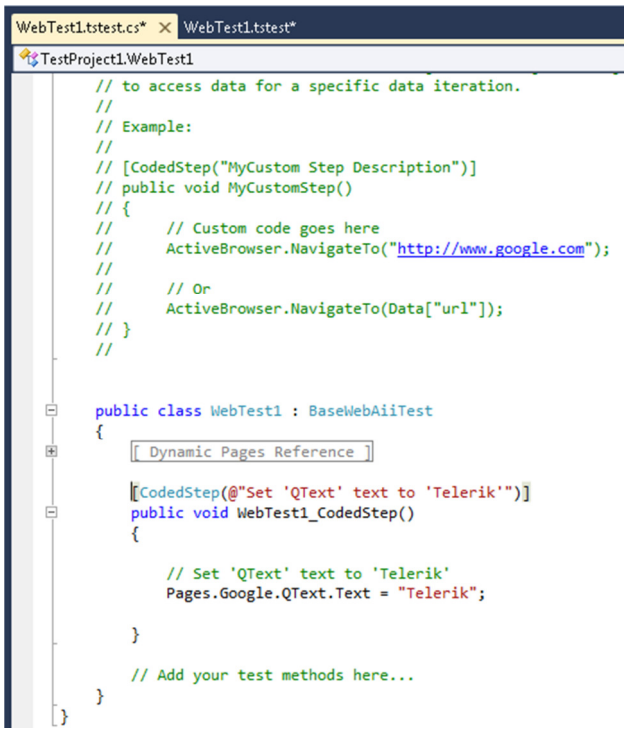




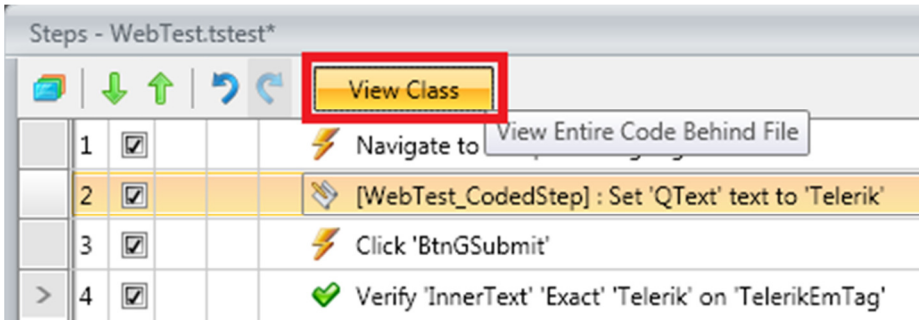
6. Once a step is converted to code, it cannot be converted back to a regular step.

Create a Test That Uses a Code Behind File

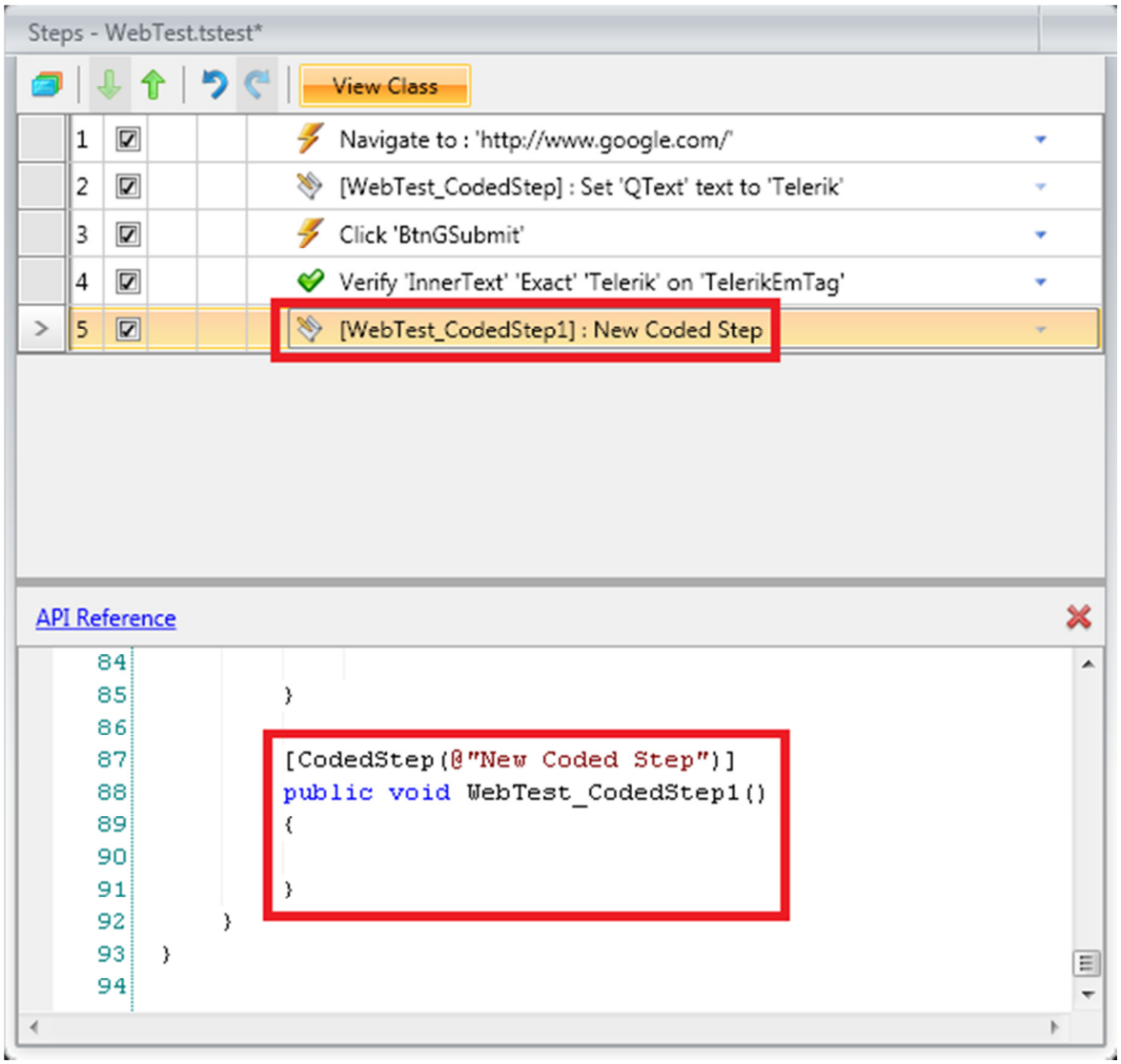




1. Create a test and add a code behind file using Step 3a outlined above.
2. To see the entire code behind file, click the “View Class” button below the test title.



3. 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.
4. Notice a code method skeleton is created. Change the description and/or the method name to suite your needs.



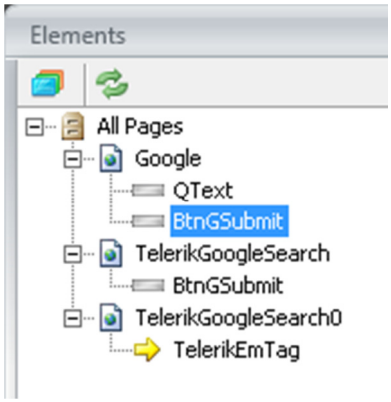
5. Do not remove the “CodedStep” attribute. This is how Telerik Test Studio recognizes custom coded steps versus other methods it should ignore.
6. In order to write custom steps in code with the VS Plug-In, add a method in the code behind file that takes no parameters and has ”void” as its return type.
7. Now add code to the method. The method must be decorated with the “CodedStep” attribute:

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

8. Add the above method or one similar to the code behind and save.
9. The name of the method becomes the step name. The Coded Step attribute description is listed as the description in the steps tab.
10. Save and build the project.
11. Execute the test.
12. The code behind file has access to all Telerik run-time objects, like ActiveBrowser. This is an identical coding experience to the Telerik run-time automation framework.

How to Reference an Element in a Code Behind File

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



2. For example, in the image above, the "BtnGSubmit" element is referenced this way:

```
Pages.Google.BtnGSubmit
```

3. It returns the strongly typed Telerik framework "HtmlInputSubmit" type.