

VERSION 2015.1  
JANUARY 19, 2015



# **Telerik** Test Studio

## TEST STUDIO QUICK-START GUIDE

STANDALONE & VISUAL STUDIO PLUG-IN

TELERIK A PROGRESS COMPANY

# TEST STUDIO QUICK-START GUIDE

---

## CONTENTS

Create your First Test .....	2
Standalone Web Test.....	2
Standalone WPF Test.....	4
Visual Studio Plug-In Web Test.....	7
Visual Studio Plug-In WPF Test.....	11
Recording Toolbar Overview.....	14
Add Quick Verification Steps.....	15
Create Advanced Verification Steps.....	17
Create a Data Driven Test.....	20
Custom Steps.....	25
Test Execution (Quick Execution) .....	27
Test Lists .....	29
Static Test List.....	29
Dynamic test List.....	31
Test Results.....	33
Elements Explorer .....	35
Change How an Element Is Found .....	38

---

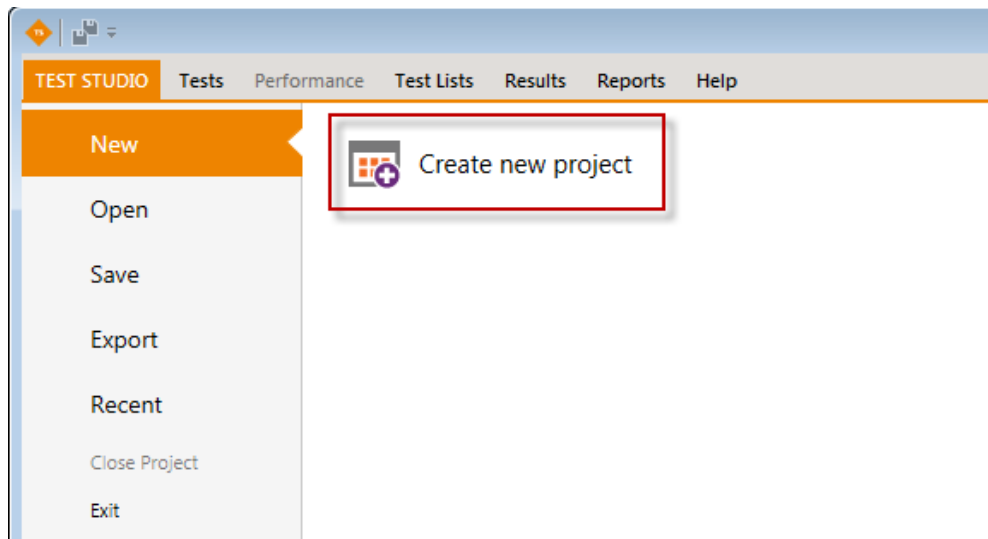
## CREATE YOUR FIRST TEST

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.

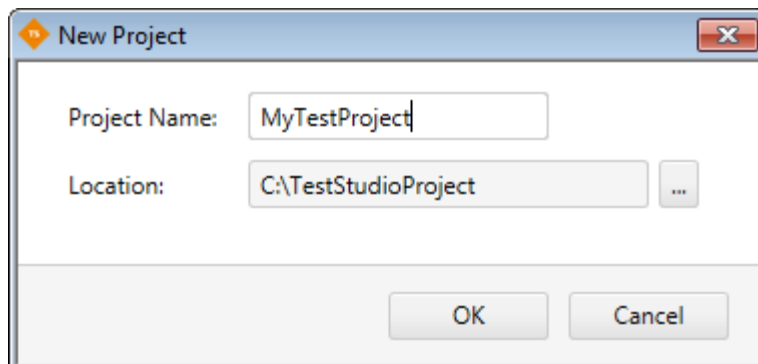
---

## STANDALONE WEB TEST

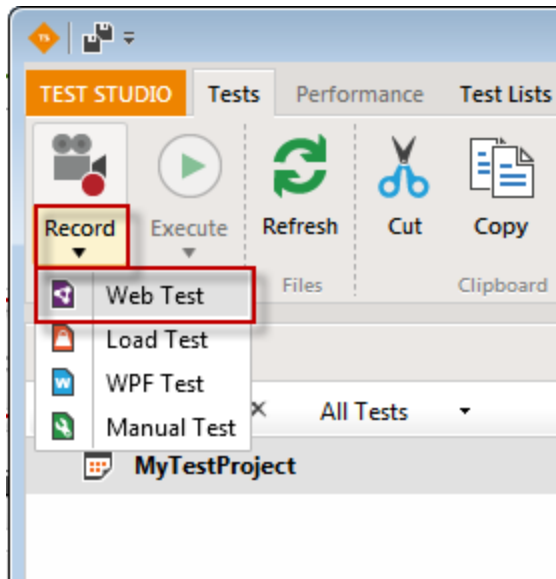
1. Launch Telerik Test Studio.
2. Click **New > Create new project**.



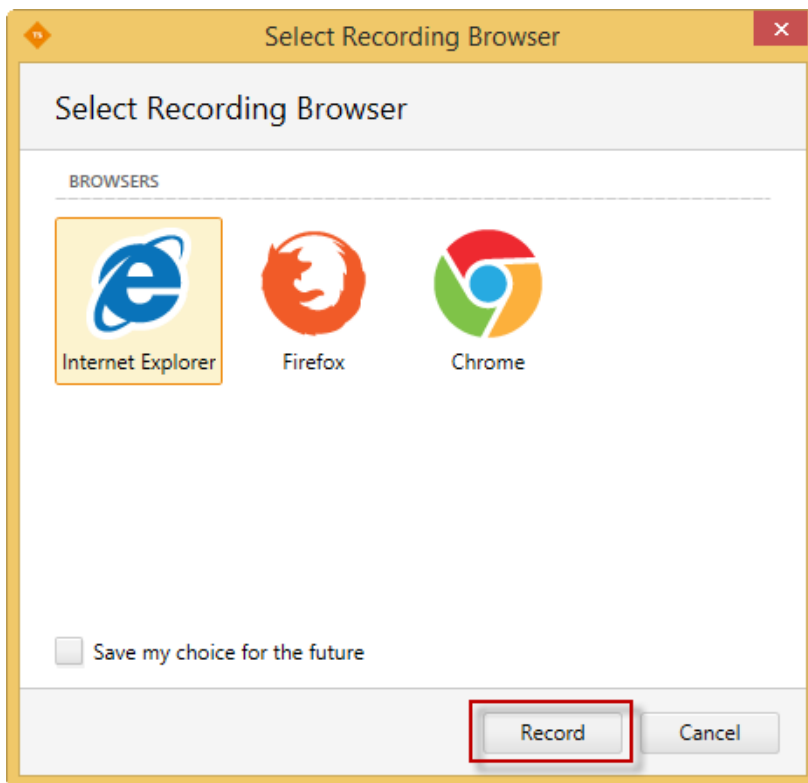
3. Provide Project Name and Location.



4. Click **Record > Web Test**.



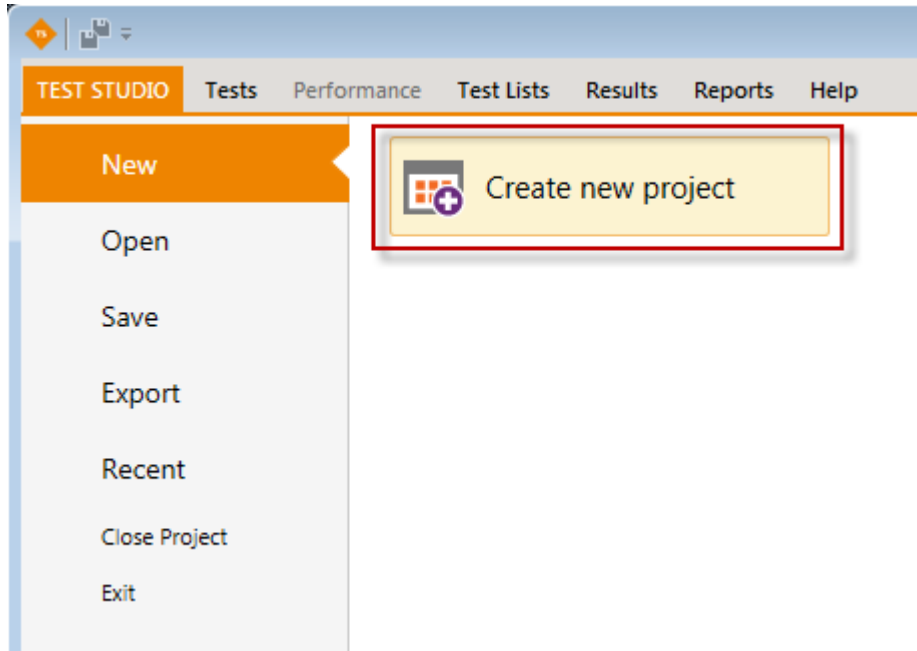
5. Select recording browser and click **Record**:



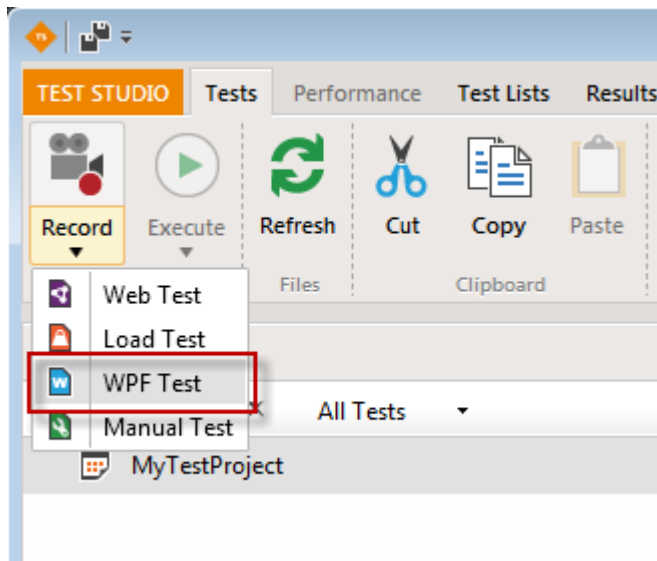
---

## STANDALONE WPF TEST

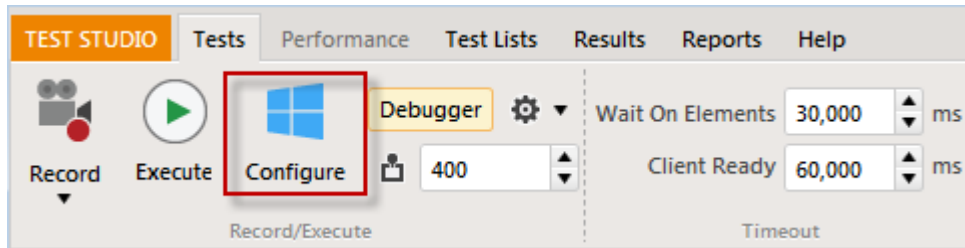
1. Launch Test Studio.
2. Click **New > Create new project**.



3. Click **New Test > WPF Test**.

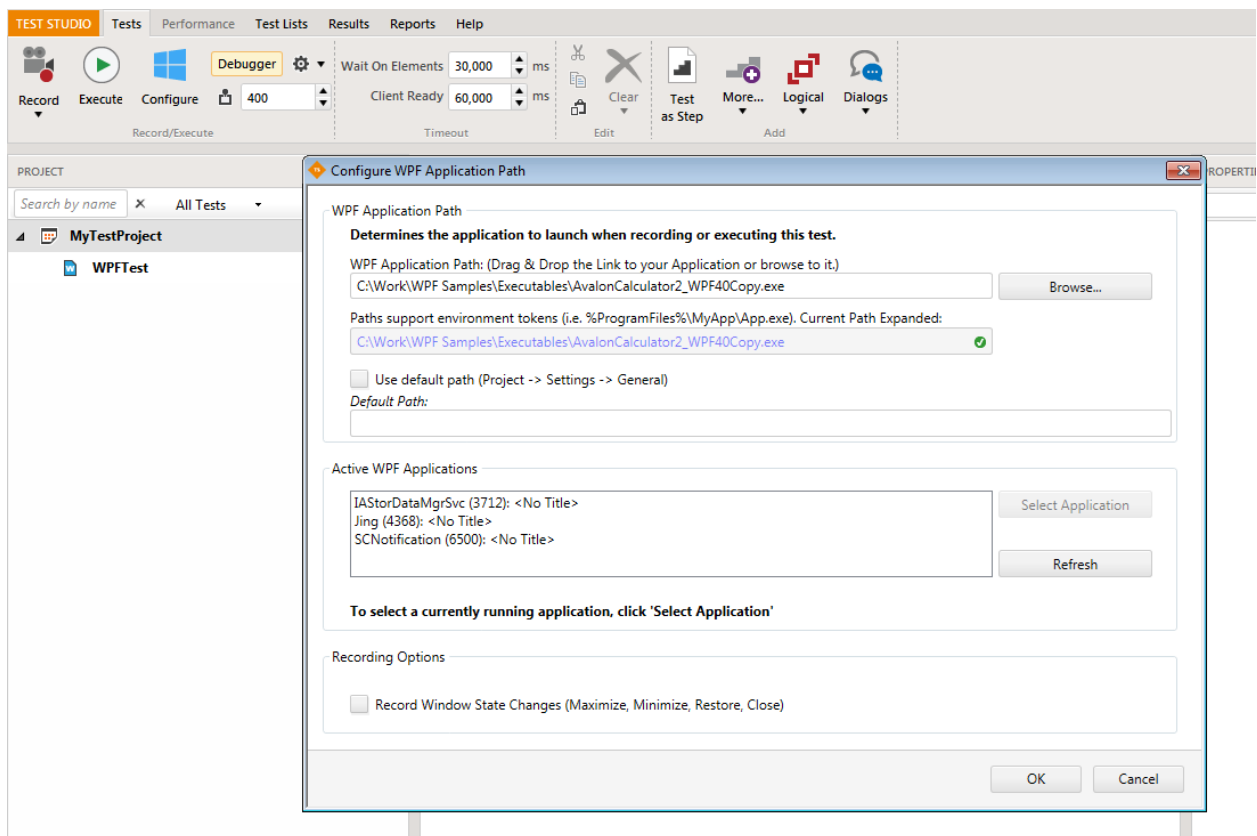


4. Click **Configure** button.

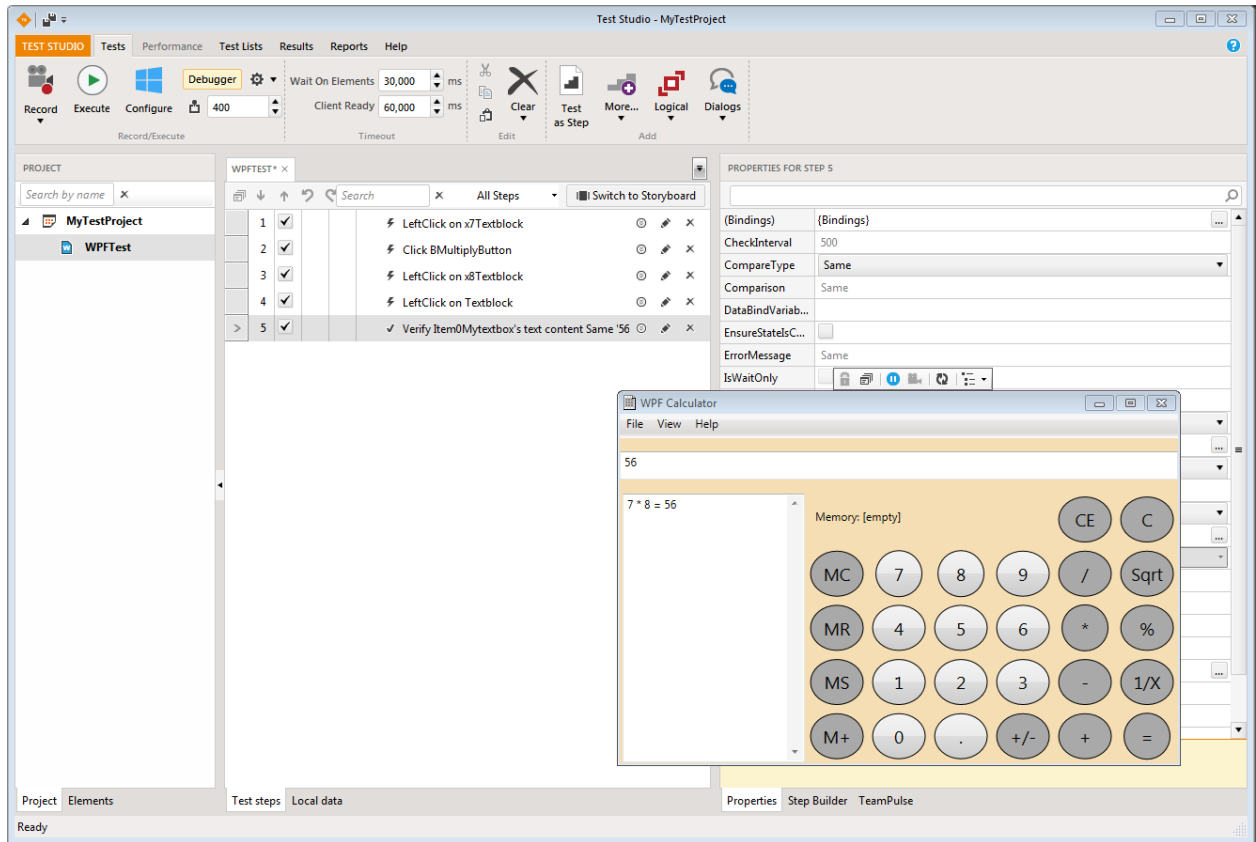


The **Configure WPF Application Path** window appears. There are two options to determine the default application to launch when recording and executing this test.

- **WPF Application Path** - drag and drop the shortcut icon into this text box, or click **Browse** and locate it manually.
- **Current Path Expanded** - read-only display of full path if environment variables are used.
- **Use default path** - whether to use the path set here or the default path set in **Project Settings > General**.
- **Active WPF Applications** - Telerik Test Studio detects all WPF apps currently running and lists them. Highlight the desired app and press **Select Application**.
- **Recording Options** - whether to record window state changes.



- Click **OK**. If you need to later modify the WPF application path, choose **Configure** from the toolbar.
- Hit **Record** to launch the app with the recording toolbar docked at the top.

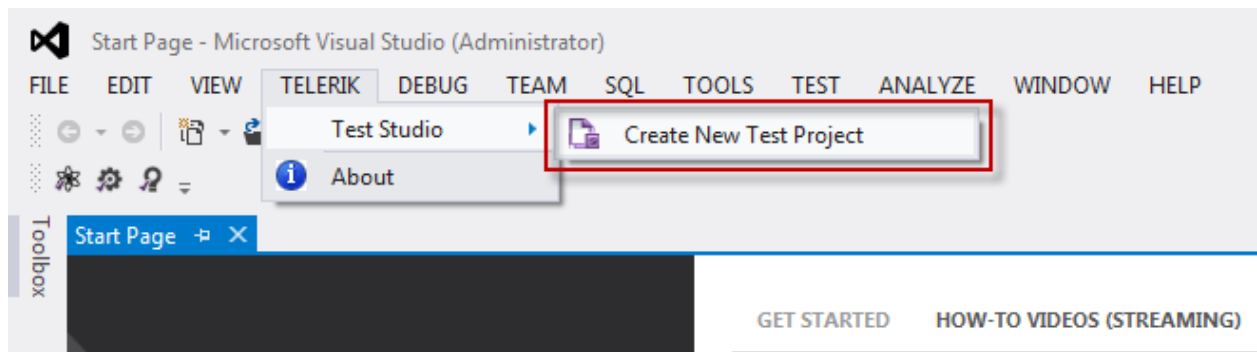


Notice that steps are added to the test as actions are taken within the application.

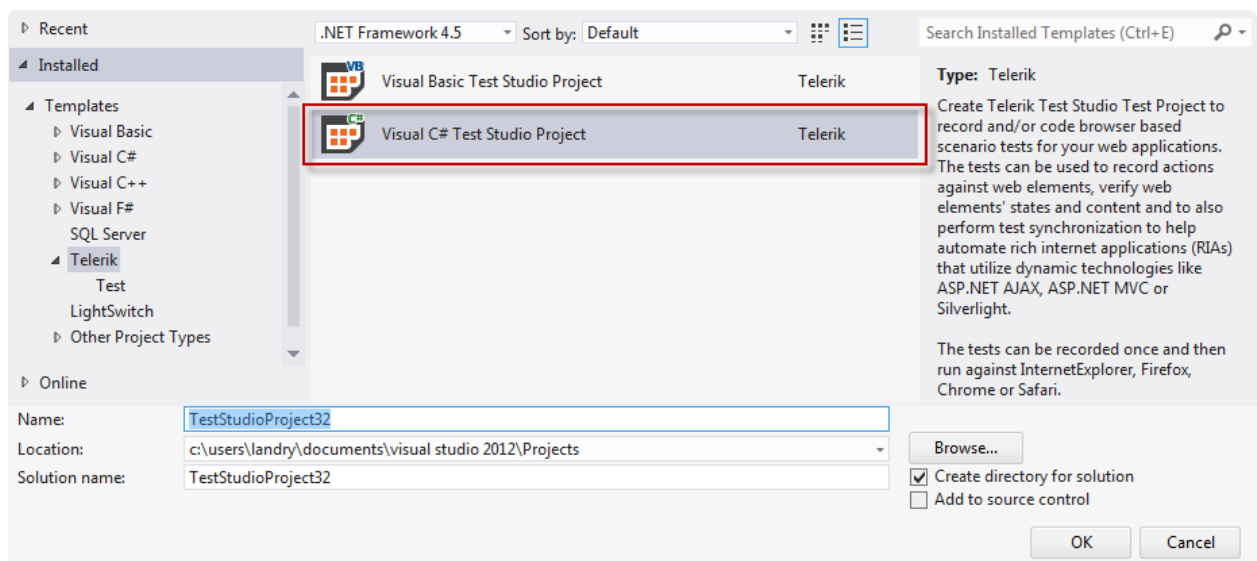
Close the application to stop recording. That's how easy it is to create your first WPF Test!

## VISUAL STUDIO PLUG-IN WEB TEST

1. Launch Visual Studio.
2. Click **Telerik > Test Studio > Create New Test Project**.



3. Choose **Telerik > Test > VB or C# Test Studio Project**, name the project and click **OK**.

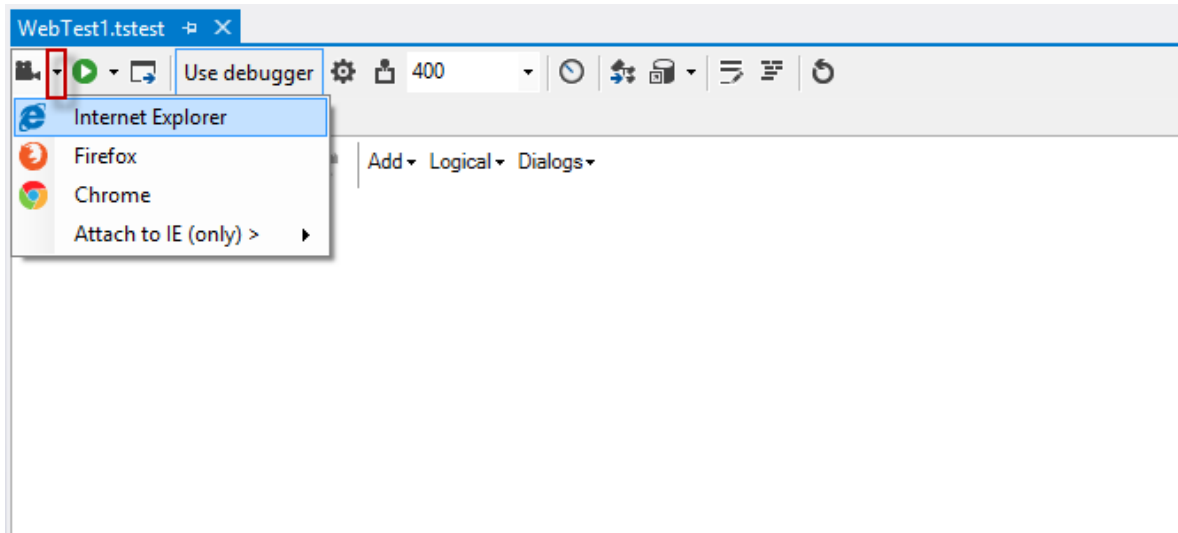


4. A Web Test is added to the project. Open the Web Test to view the Steps pane.

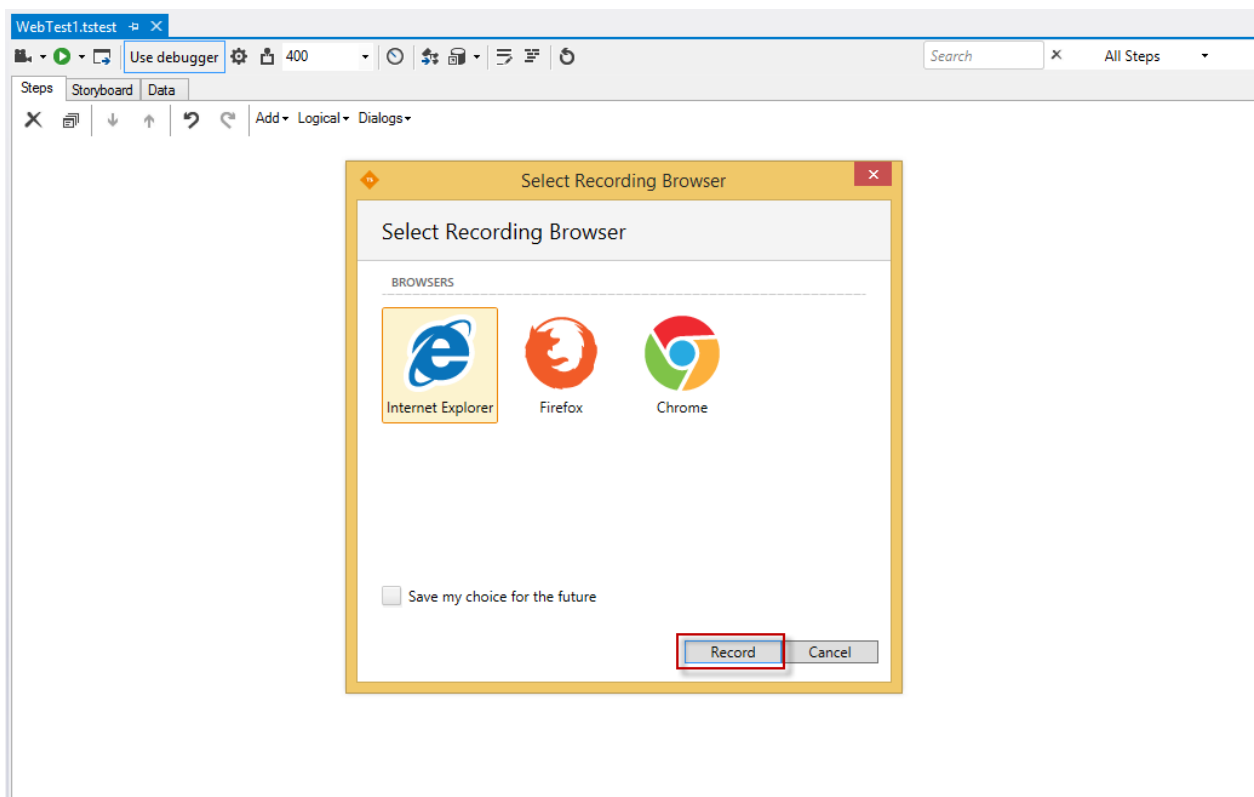


## BEGIN RECORDING IN THE VS PLUGIN

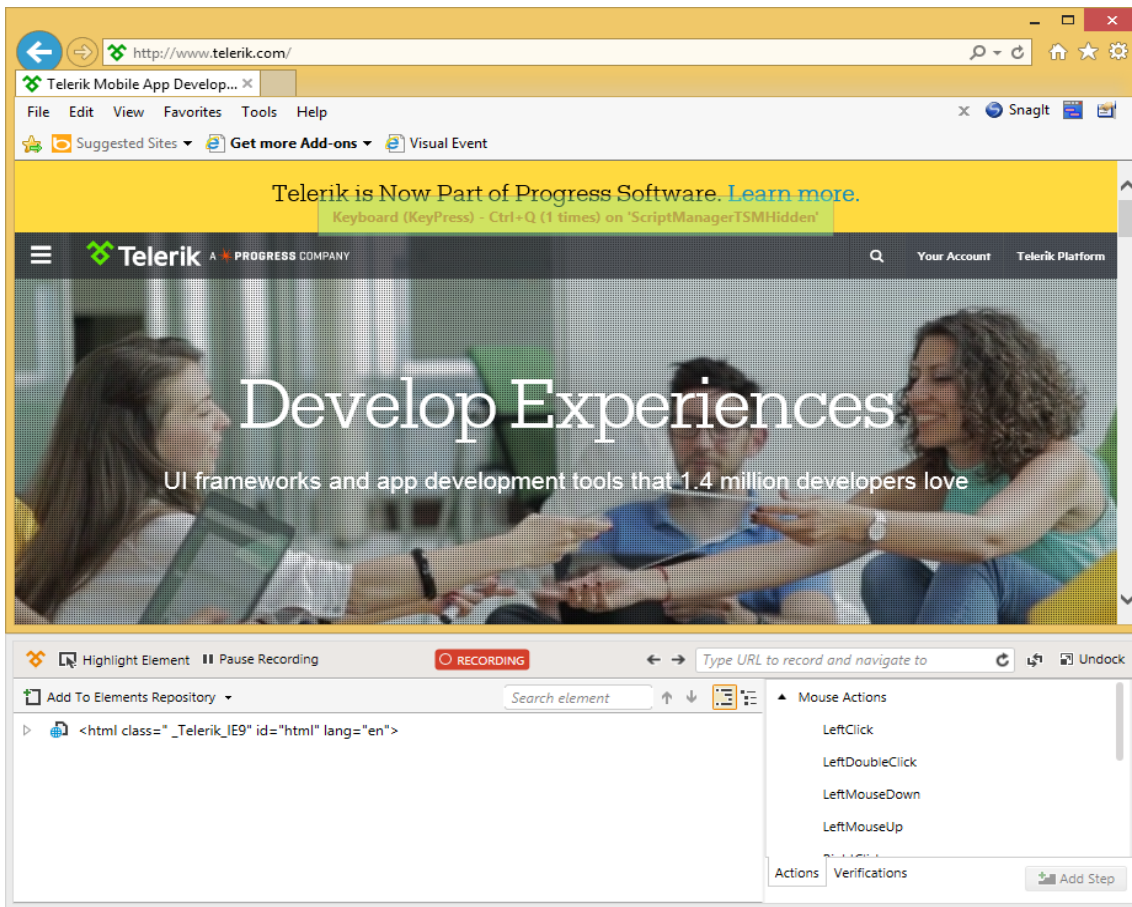
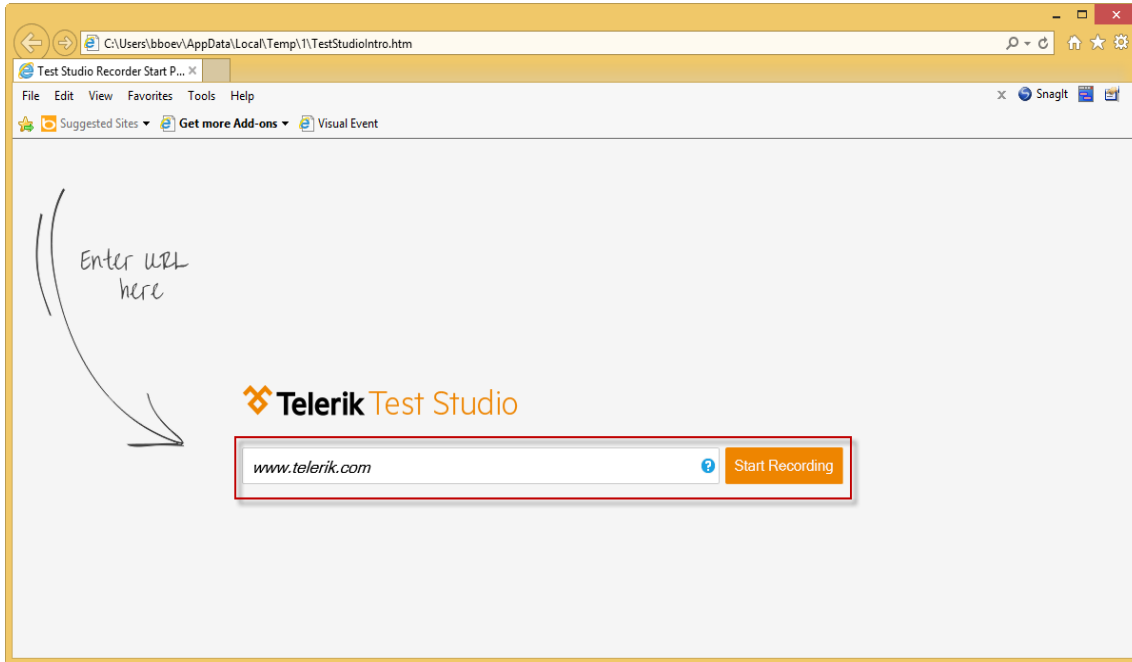
1. Select the recording browser. The recording will start automatically in the selected browser (2013 R2).



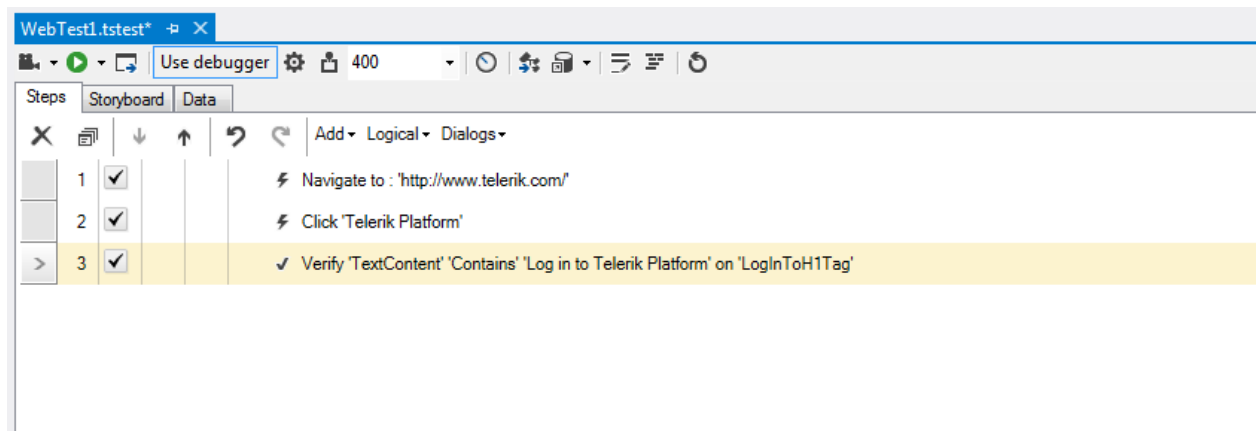
Or click the **Record** button to begin recording test steps in the already selected default browser.



2. Enter a URL and press Enter/Start Recording button. A recording step is added to the Steps pane and the recorder is attached to the browser.



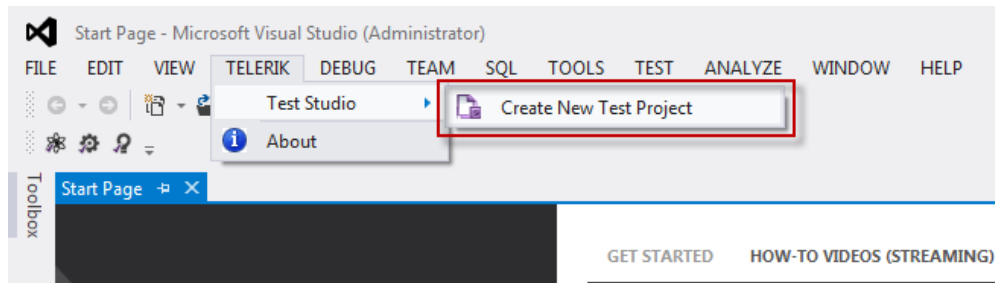
3. Perform actions on the page. More steps are added to the Steps pane.



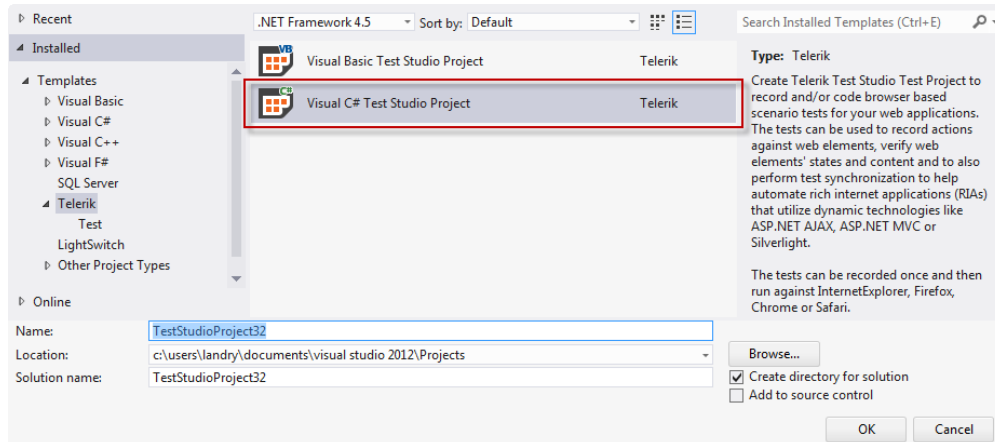
4. Close the recording browser window. That's how easy it is to create your first Web Test! Save and build your project.

## VISUAL STUDIO PLUG-IN WPF TEST

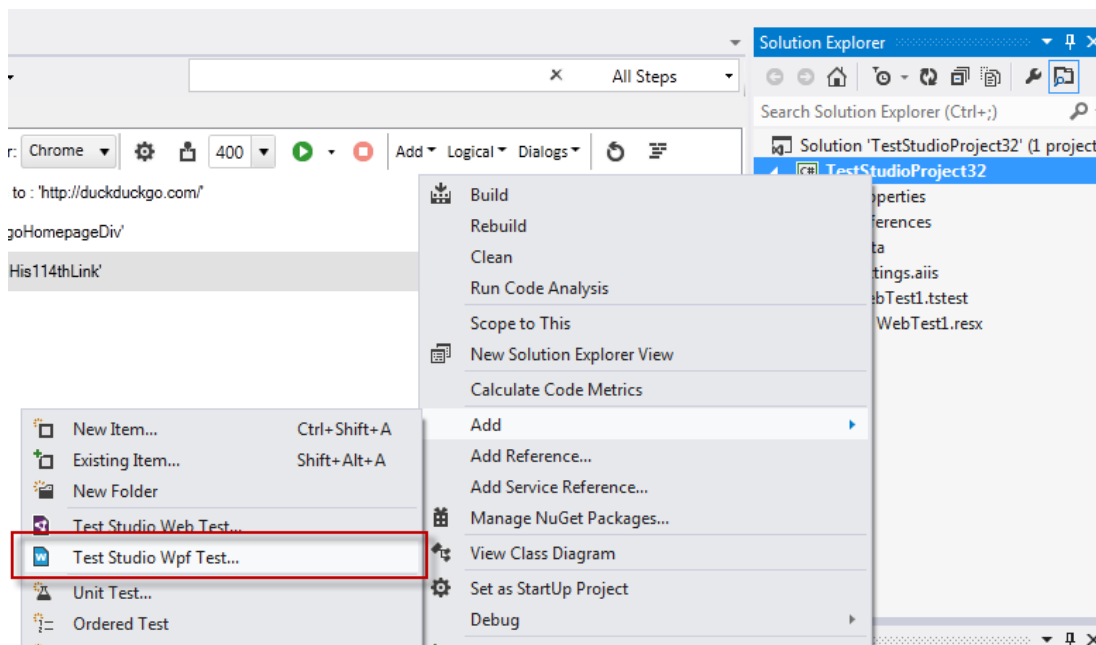
1. Launch Visual Studio.
2. Click **Telerik > Test Studio > Create New Test Project**.



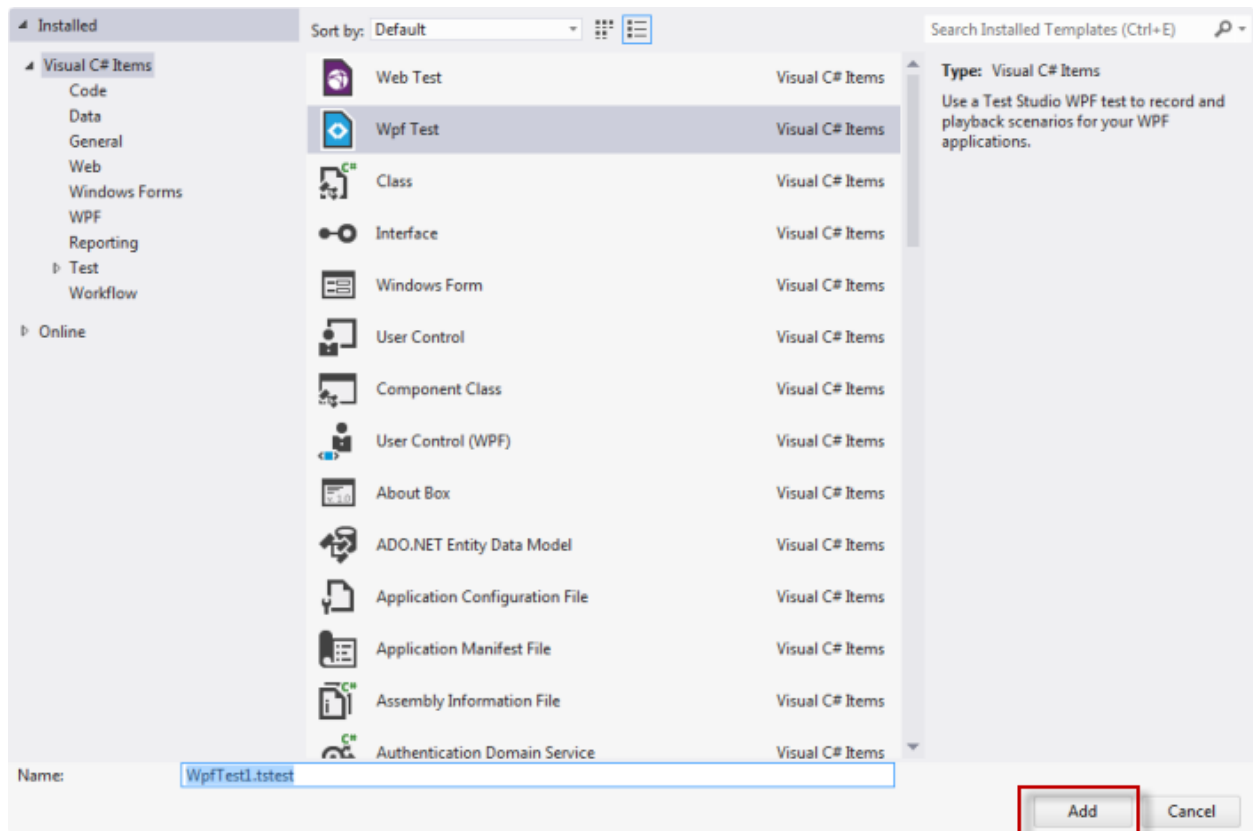
3. Choose **Telerik > Test > VB or C# Test Studio Project**, name the project and click **OK**.



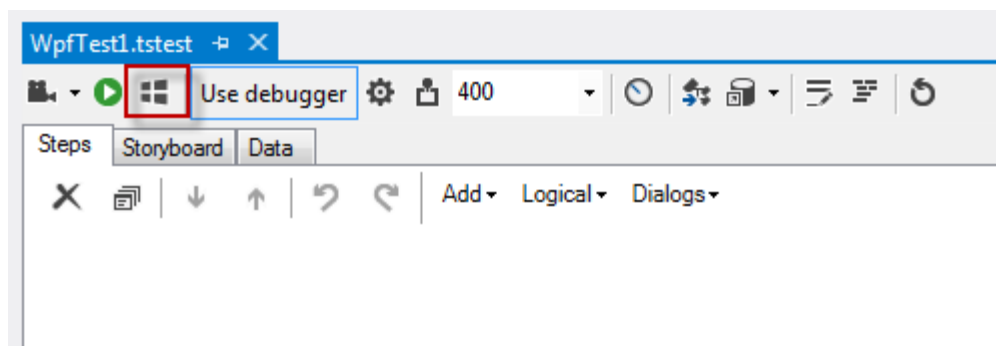
4. Right click on the project node in the solution explorer and select **Add > New WPF Test**.



5. Click **Add**.



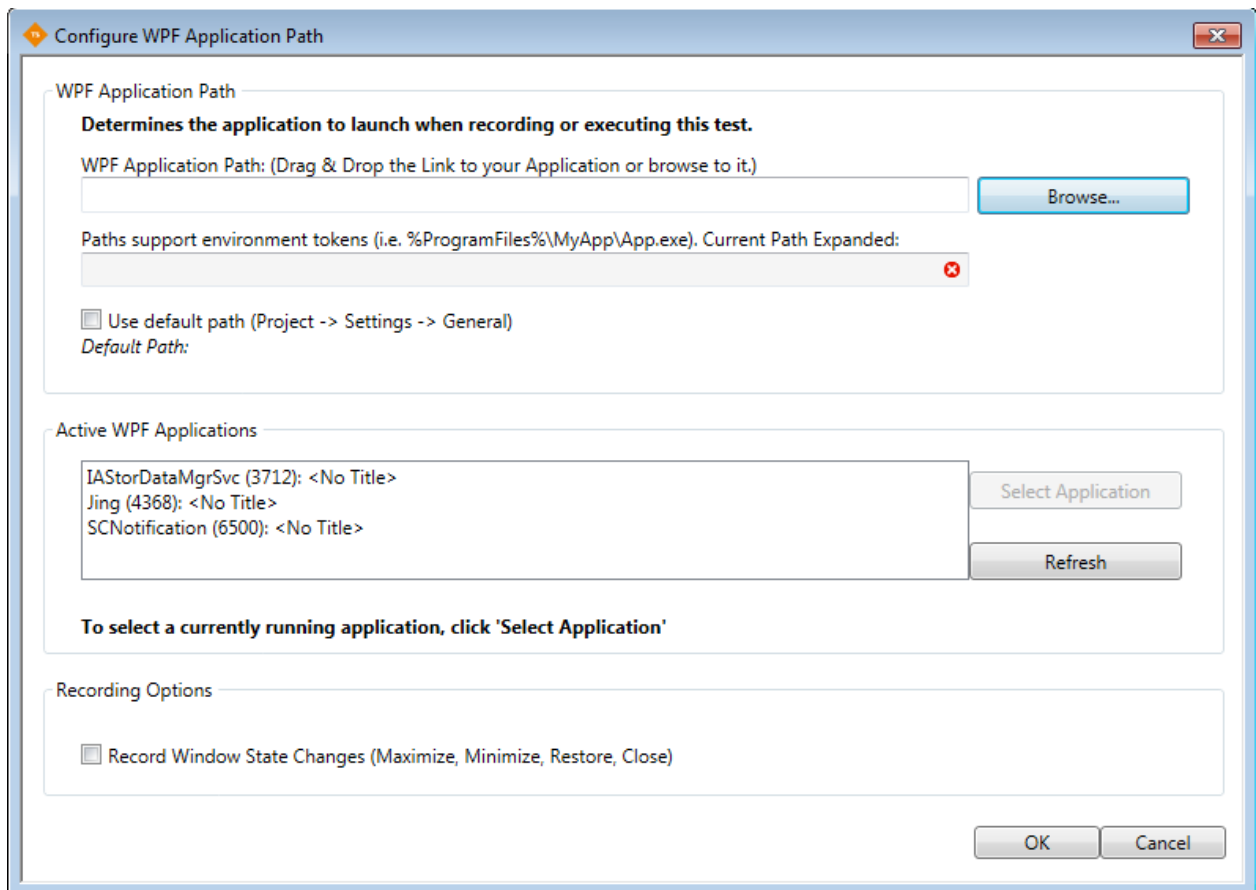
6. Open the new test and click the **Configure WPF Application Path** icon in the toolbar.



7. The **Configure WPF Application Path** window appears. There are two options to determine the default application to launch when recording and executing this test.

- **WPF Application Path** - drag and drop the shortcut icon into this text box, or click **Browse** and locate it manually.

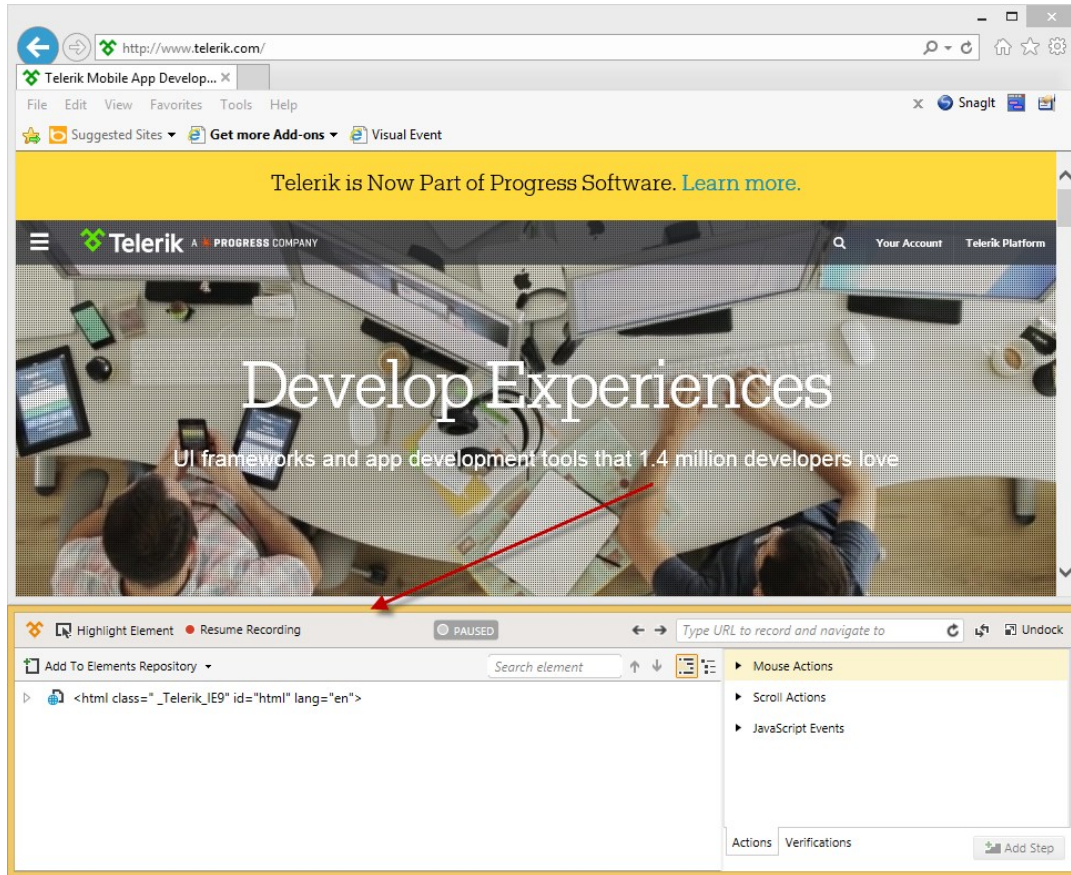
- **Current Path Expanded** - read-only display of full path if environment variables are used.
  - **Use default path** - whether to use the path set here or the default path set in **Project Settings > General**.
  - **Active WPF Applications** - Telerik Test Studio detects all WPF apps currently running and lists them. Highlight the desired app and press **Select Application**.
  - **Recording Options** - whether to record window state changes.
8. Hit **Record** to launch the app with the recording toolbar docked at the top.
  9. Notice that steps are added to the test as actions are taken within the application.











10. Close the application to stop recording.

## RECORDING TOOLBAR

After entering a URL and pressing Enter/**Start Recording** button the recording toolbar is attached at the **bottom** of the browser:



Here are its features:

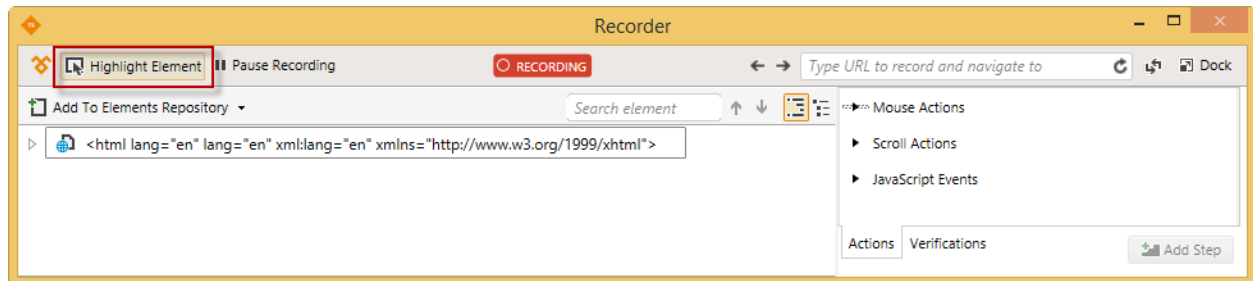
- |  |  |
|--|--|
|  Highlight Element                  | <b>Enable/Disable Hover Over Highlighting</b> (Press " <b>Pause/Break</b> " to toggle)   |
|  Pause Recording                    | <b>Pause Recording</b> (Press " <b>Print Screen</b> " to toggle)   |
|  Resume Recording                   | <b>Start Recording</b> (Press " <b>Print Screen</b> " to toggle)   |
|  Reconnect Recorder                 | <b>Reconnect Recorder</b> (Restores communication between Test Studio and the recording browser window or WPF app, if necessary) |
|  Type URL to record and navigate to | <b>Navigation panel</b> (Type URI to record and navigate to / Refresh Browser))  |
|  Undock / Dock                      | <b>Undock/Dock</b> (Undock or Dock again the recorder)   |
|  Add To Elements Repository         | <b>Add To Elements Repository</b> (Add a single or multiple elements to the <a href="#">Elements Explorer</a> )                  |
|                                     | <b>Elements Tree/Tag View</b> (Whether to display the elements as tree view or tag view )  |



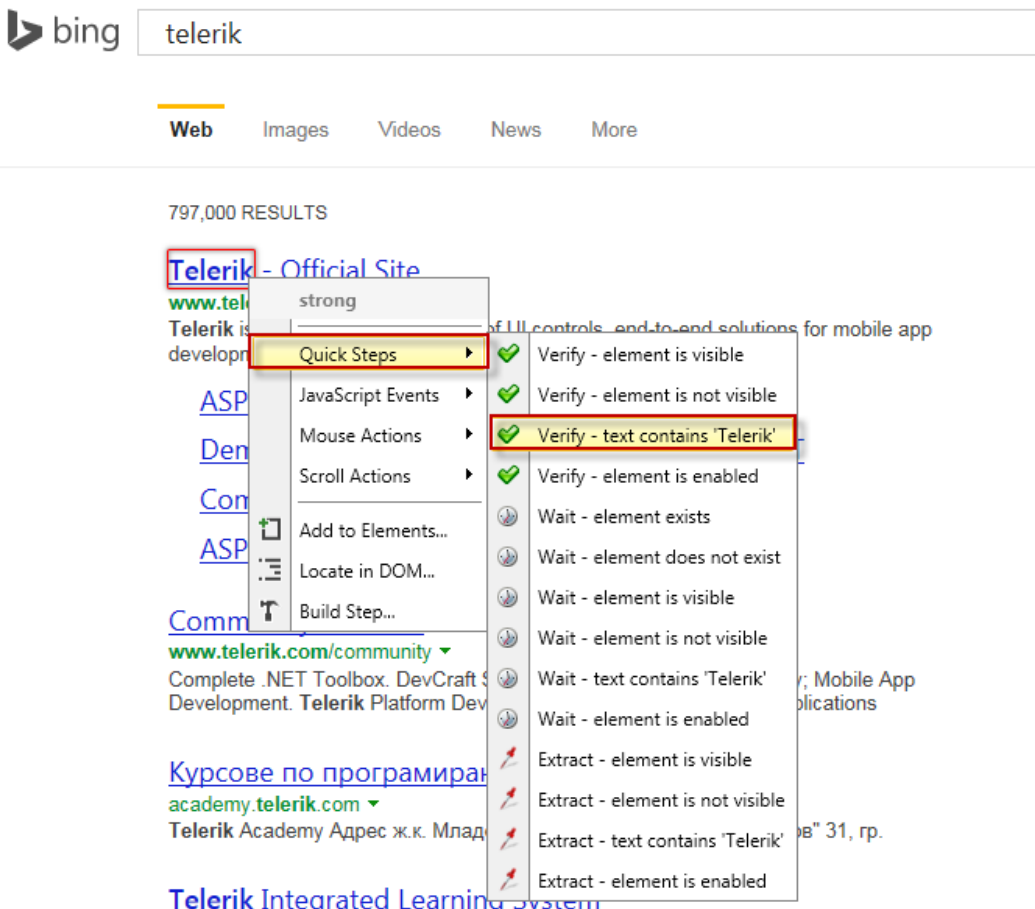
## ADD QUICK VERIFICATION STEPS

The easiest and fastest way to add a verification to your test is through the Quick Tasks menu.

1. Create a Web Test and click Record.
2. Navigate to <http://bing.com>
3. Enter Telerik in the search box and click the Search button.
4. Enable hover over highlighting by clicking Highlight Element in the Test Studio Recorder.

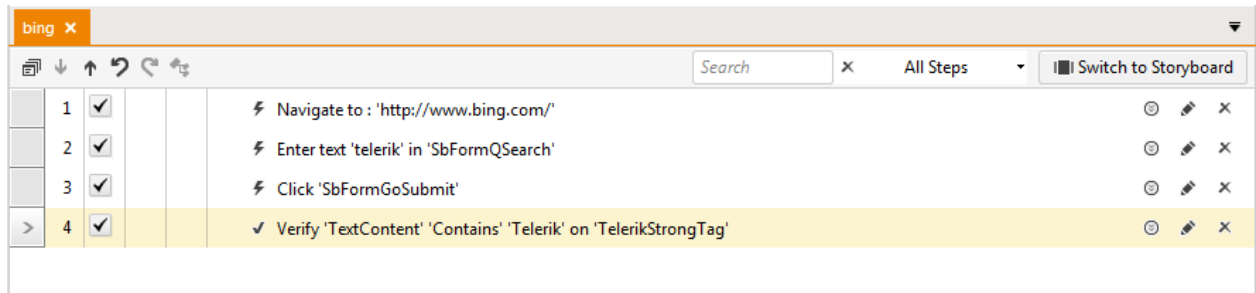


5. In the recording browser window, hover over the element against which to verify. Choose **Quick Steps > Verify - text contains** entry.





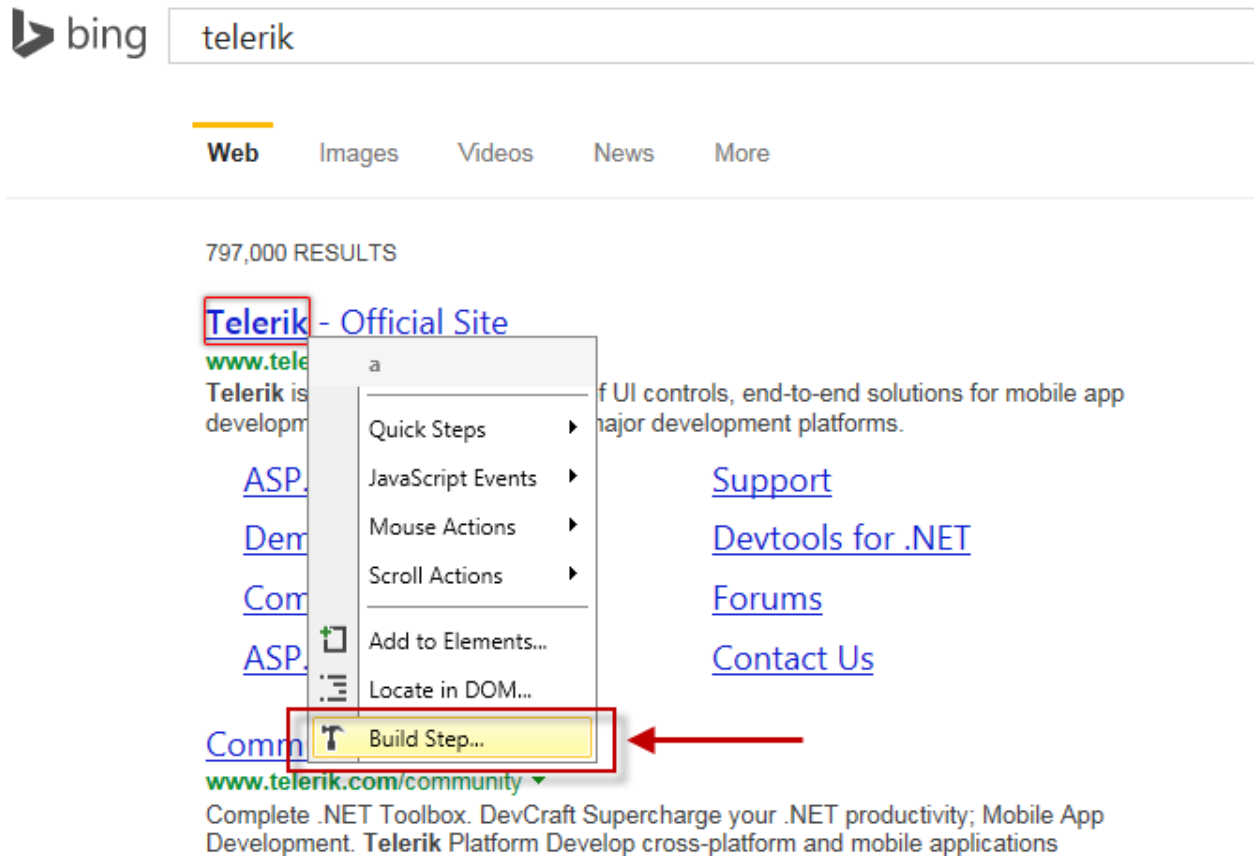
6. The Verify step is added to the test.



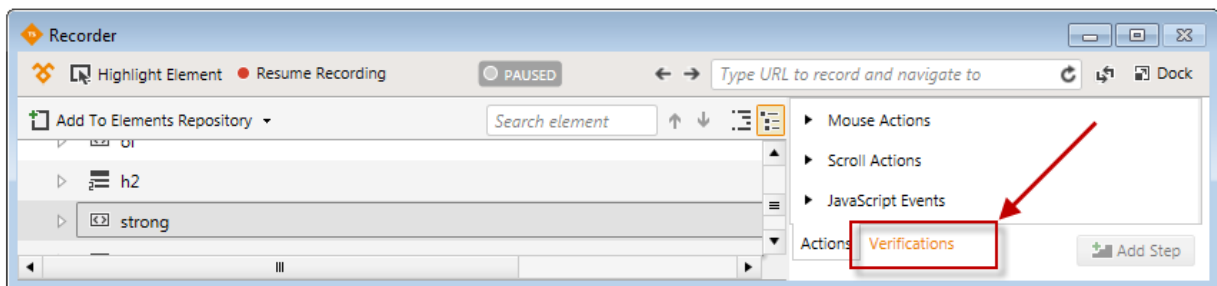
## CREATE ADVANCED VERIFICATION STEPS

An advanced verification allows you to interactively build verification rules and validate them against a live web document or WPF application.

In order to create custom verification you need to highlight the element and choose Build Step... from the menu:

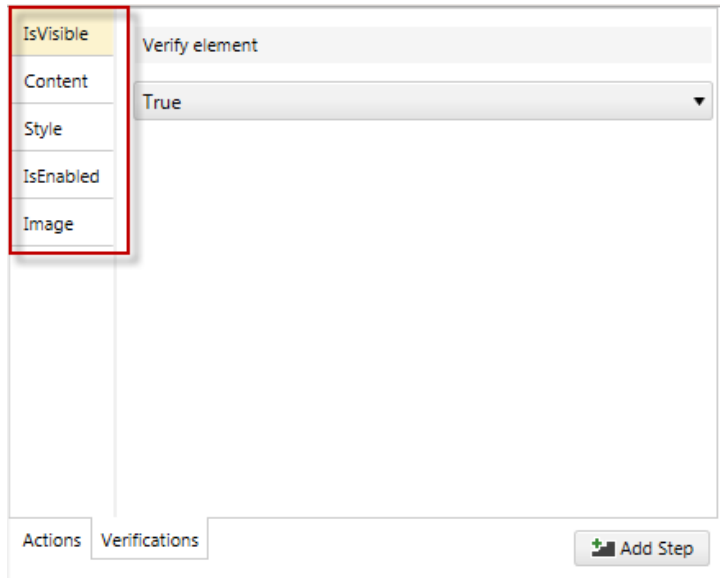


In the Recorder select **Verifications** node.

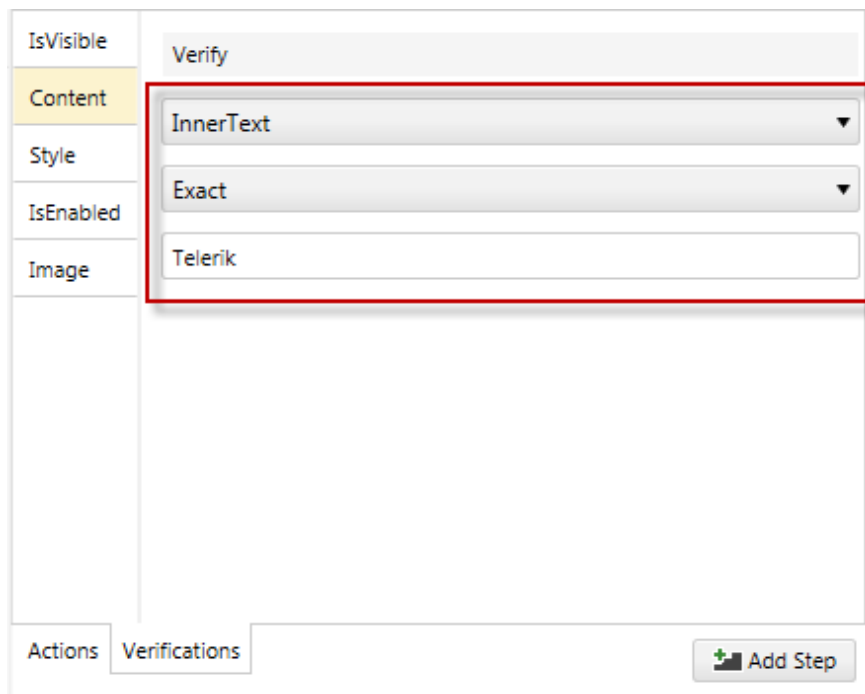


Start by selecting a type of Available Verifications:

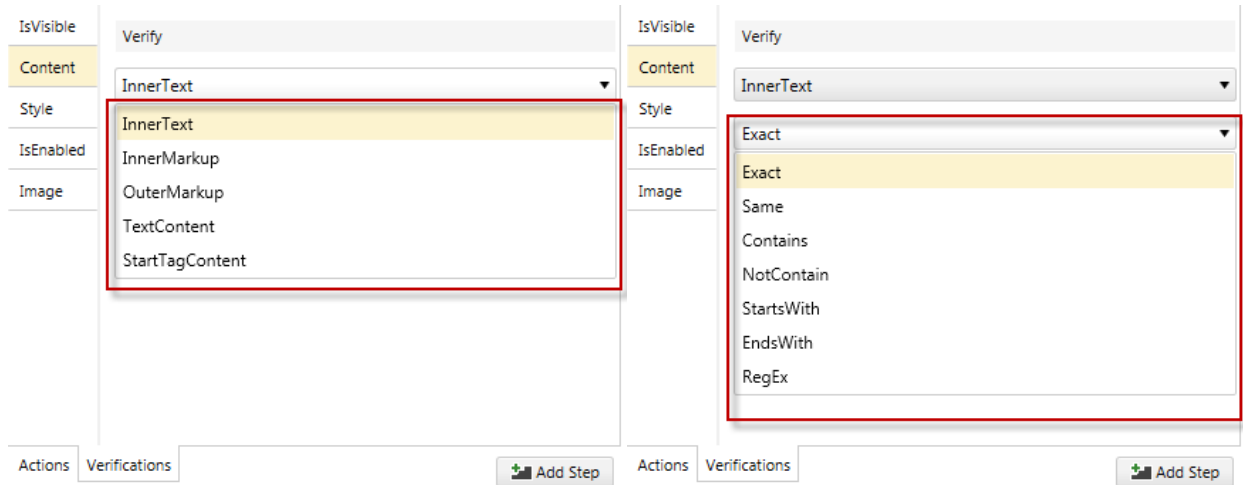
- IsVisible
- Content
- Style
- IsEnabled
- Image



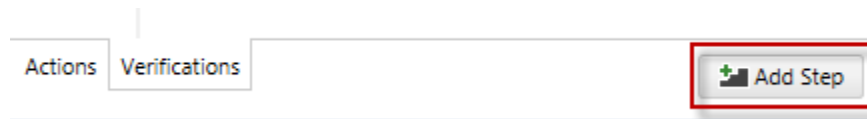
When crafting verifications, content is dynamically built against the currently selected element. As selections are made, default values are populated according to values the element contains. For example, choose Content as the verification type and three menu options appear.



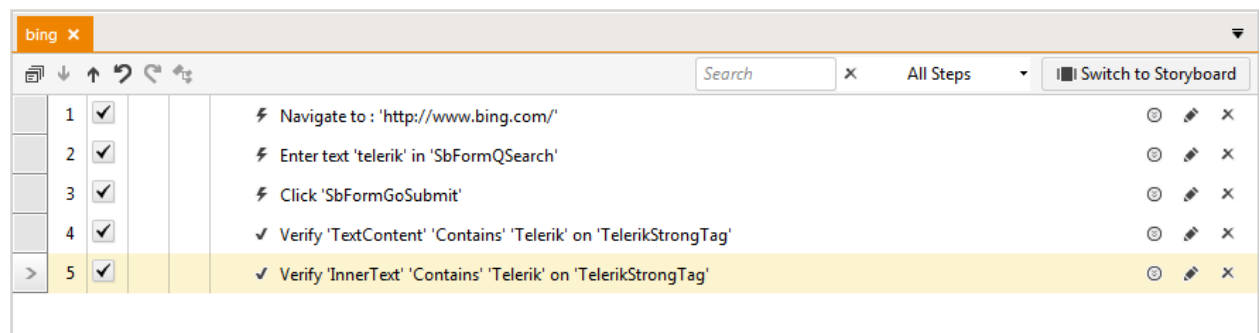
Click on the down arrow next to each option to see a list of possible values.



Once finished building the verification(s), click **Add Step** to add it as a step to the current test.



The newly created verification appears in the test:



We recommend against using the Content Markup validation types. They are fragile in the face of minor page changes, and different browsers may reorder the element attributes making them unreliable. For more information please see our Automated Testing blog entry on [Understanding Validation Content Element Types](#).

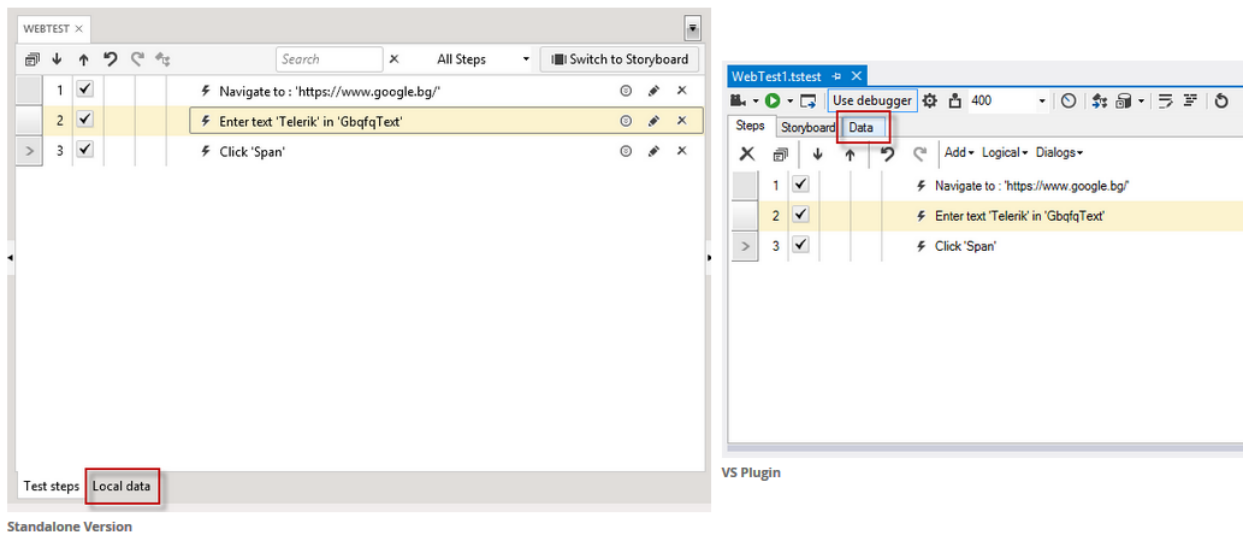
## CREATE A DATA DRIVEN TEST

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

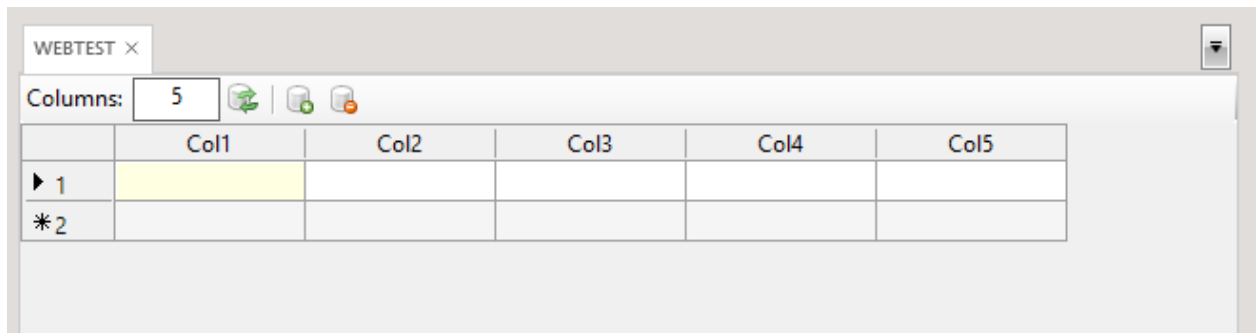
1. Create a new Web Test and click Record.
2. Navigate to [www.google.com](http://www.google.com), enter Telerik in the Google search box, and hit the Search button.
3. Click the **Pause Recording** button in the docked toolbar.



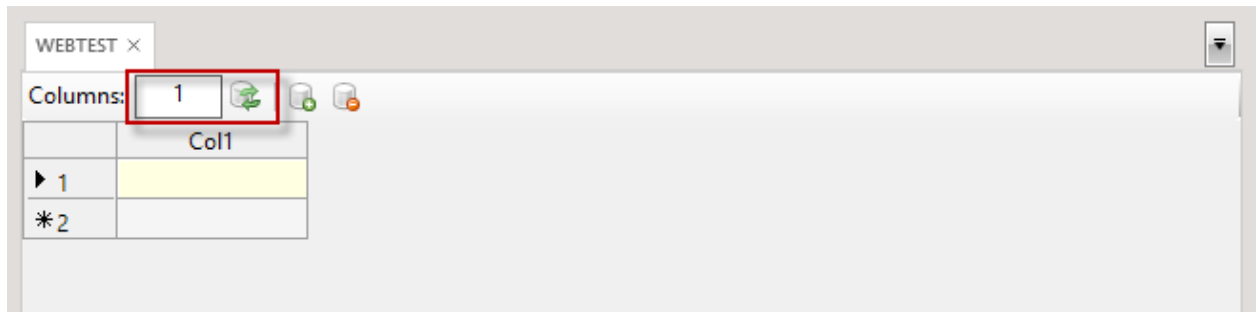
4. Click the Local Data button in the bottom of the test or the Data tab in Visual Studio.



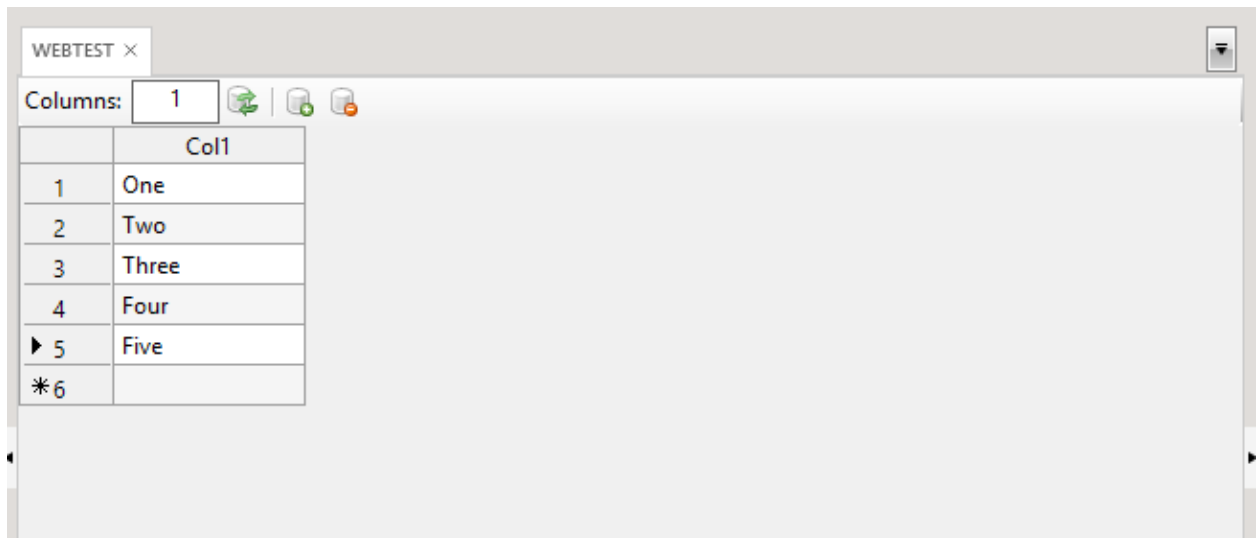
5. There are three buttons at the top of this pane. Only **Create a new data table** is enabled. Click it to add a new grid for data.



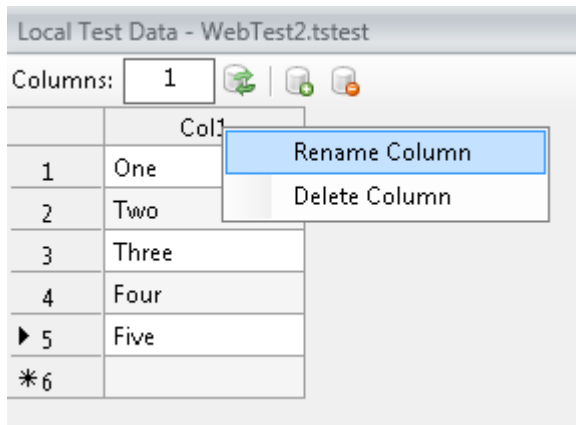
6. The default grid will have five columns. This example will execute five iterations of the test with different search text for each.
7. Change the columns text box to 1 and click **Update**.



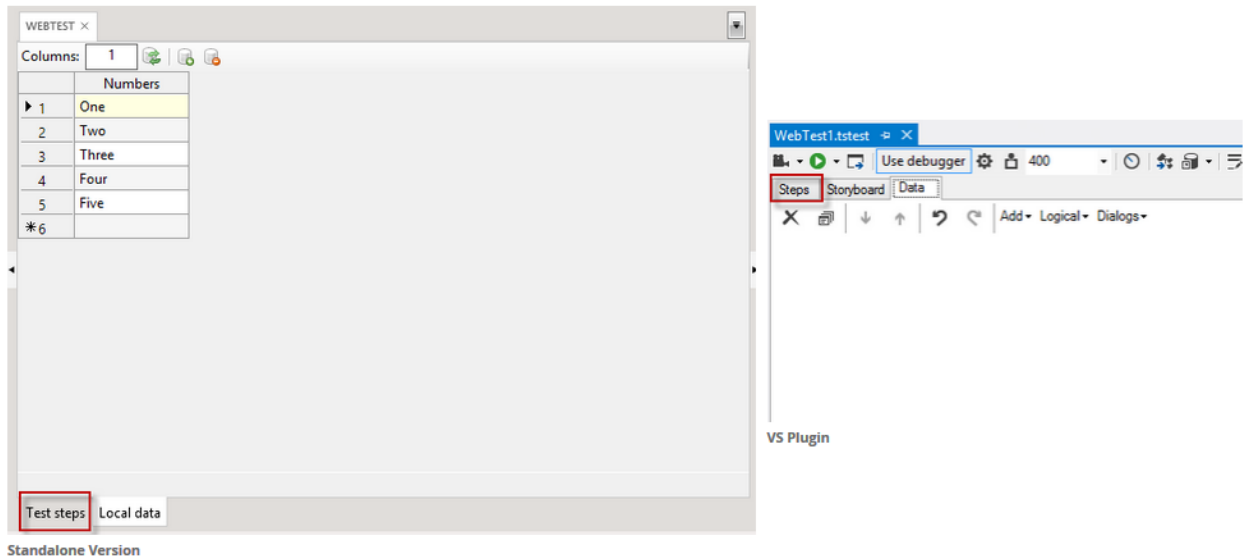
8. Enter any text into the first grid cell and hit the Enter or Tab key. The input will move to the second row.
9. Continue entering text for the remaining grid cells. New rows are added as you type.



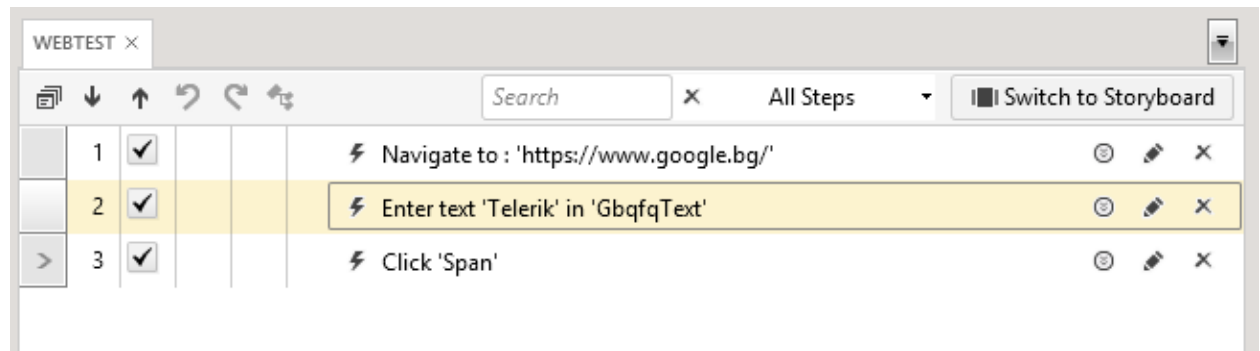
10. Right click *Col1* and choose **Rename Column**. In this example, Numbers is the new name.



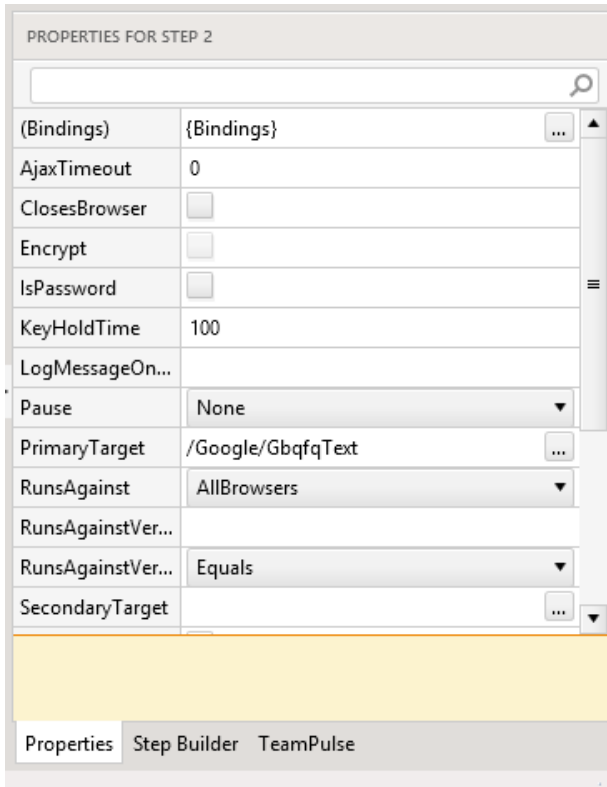
11. Save the test.
12. Data from an array can be used in recorded steps and code behind methods. To bind data from a data array to a recorded step, continue step 13. To use reference data from the data array in a code behind method, skip to step 22.
13. Click the **Test Steps** at the bottom of the test or the **Step** tab in Visual Studio to return to the Test Steps View



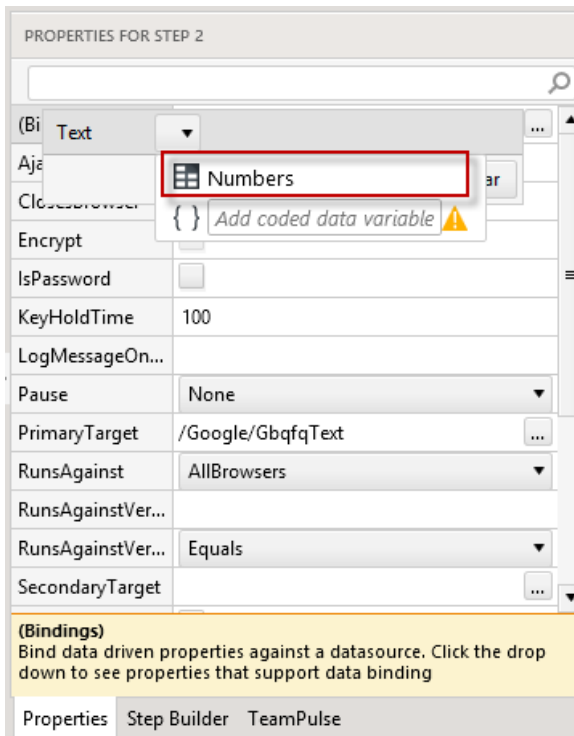
14. Highlight step 2. This is the recorded step that sets the value of the Google search text box.



15. The properties for this step appear in the Properties pane, located on the right of the screen.

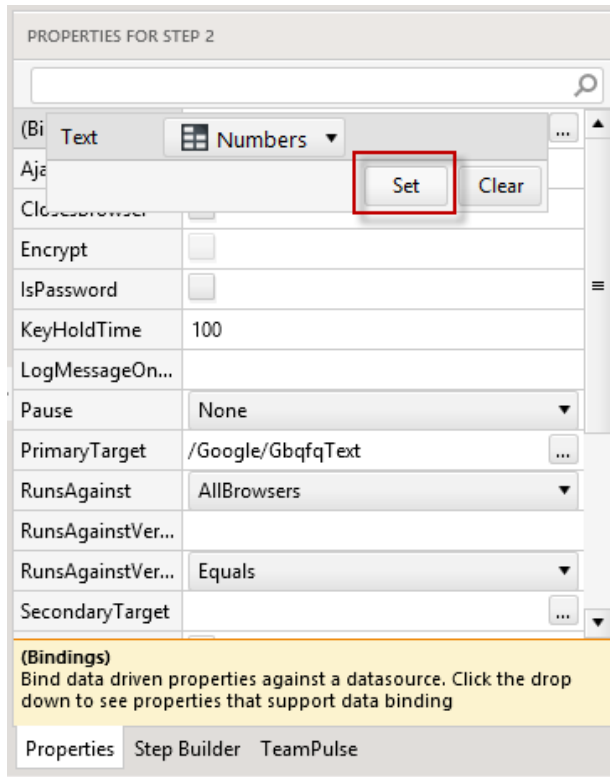


- 16. Click the '...' button for **(Bindings)**.
- 17. Choose the '**Numbers**' from the drop down for **Text**.





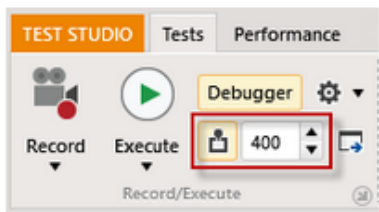
18. Click the **Set** button.



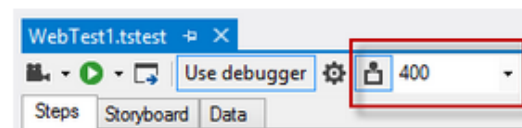
19. The data for the column named Numbers from the data array is now bound to the Text property for that step. Instead of entering Telerik into the search box, the data stored in the array will be entered.

20. Save and execute the test. Note that the test will execute for each row in the data array, for a total of five iterations.

21. 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.



Standalone Version

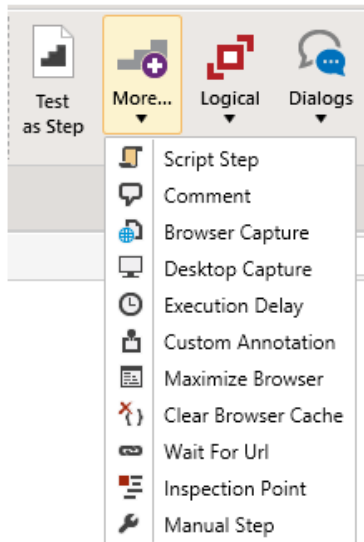


VS Plugin

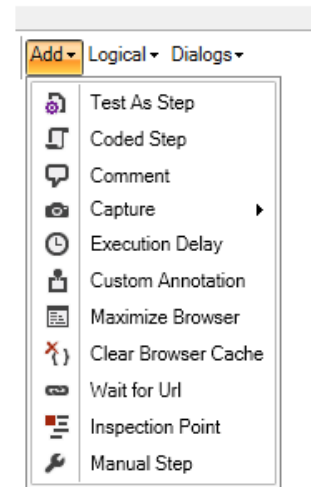
## CUSTOM STEPS

The Recording Surface can help build a wide range of automation and verification quickly and without having to resort to manual configuration. However, there are some steps that need to be added manually. For these, use the "Add" ribbon in the Standalone version and the "Add" button in Visual Studio.

### Web

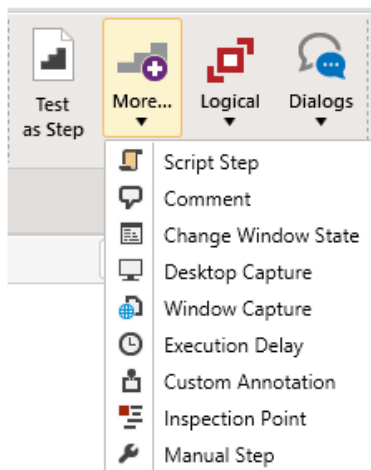


Standalone Version

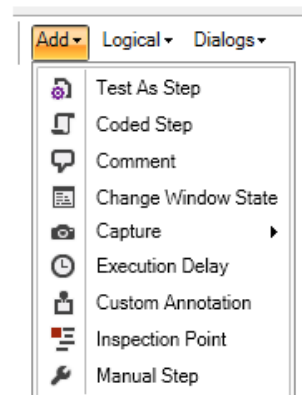


VS Plugin

### WPF



Standalone Version

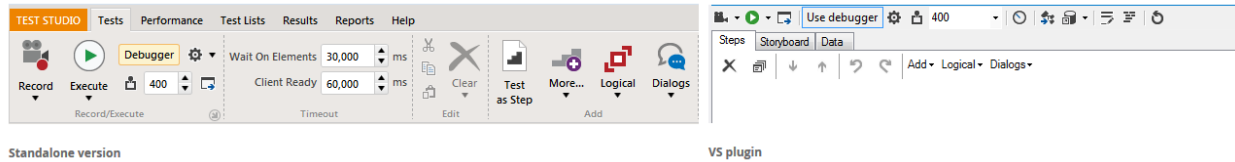


VS Plugin

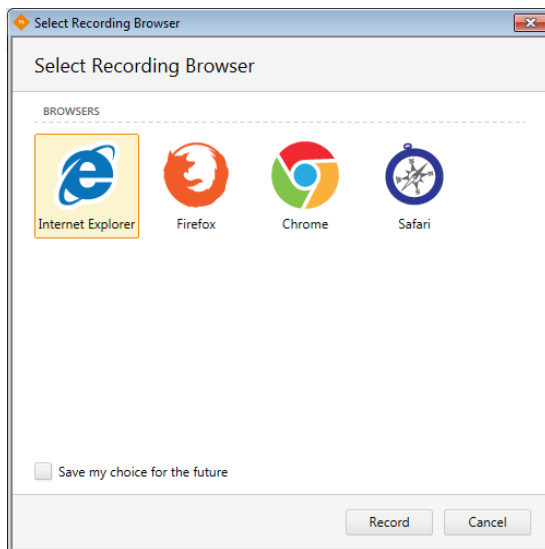
1. [Script Step](#) - add a coded step and open the code editor.
2. [Test as Step](#) - run an existing test as a single step.
3. [Comment](#) - display a text comment as a single test step and in the test log.
4. [Browser and Desktop Capture](#) - take a screenshot of only the browser or the entire desktop.
5. [Execution Delay](#) - pause the test for a specified amount of time.
6. [Custom Annotation](#) - add a note to display when running "Quick Execution" with Annotation turned on.
7. [Maximize Browser](#) - maximize, minimize or restore the active browser (Change Window State for WPF).
8. [Clear Browser Cache](#) - clears all cookies, temp files or history from the active browser depending on the user's choice.
9. [Wait For URL](#) - suspend the test until the specified URL is loaded into the browser address bar.
10. [Inspection Point](#) - pause the test and display the DOM Explorer.
11. [Manual Step](#) - display a dialog box to enter directions for a manual step.
12. [Change Window State](#) - alter the state of the WPF application window.

## TEST EXECUTION (QUICK EXECUTION)

1. Click the Execute button in the Quick Execution ribbon.



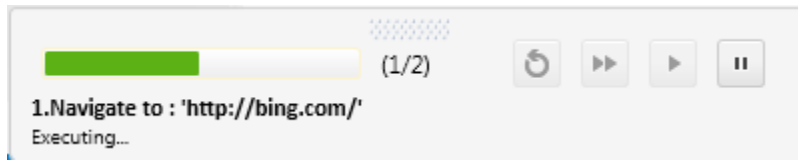
For a Web Test, you can select an alternative web browser first.



2. The Test Studio Test Runner launches first in a command prompt window. This calls the applicable browser or application.



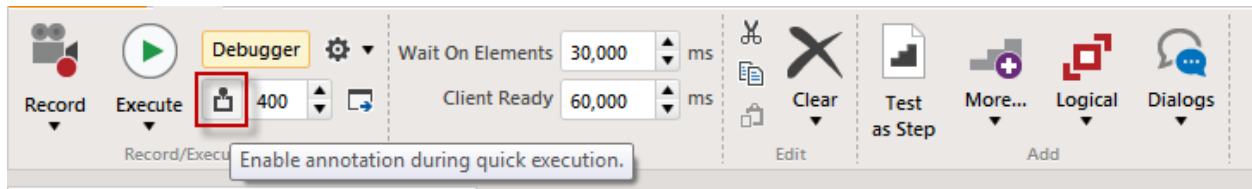
3. Notice the display in the lower right of your screen. It indicates the current step, includes play and pause ability, and shows additional Debug Options if you set a Breakpoint.



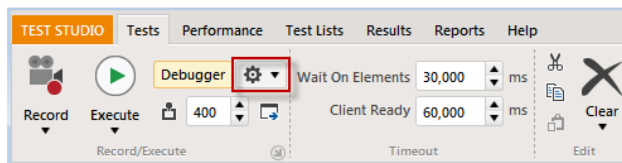
4. Afterwards, test results are automatically displayed. Click View Log for more information.



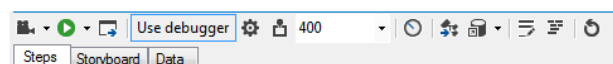
5. Click **Toggle Annotation** button to have the browser annotate each step with a brief message and by highlighting that step's element. This will also slow the test run down by inserting a delay between steps (in milliseconds) you set from either from the drop-down menu or by entering a custom value.



6. Click **Debugging Options** icon in the Quick Execution ribbon or the Visual Studio toolbar to Customize Auto-Pause Options if errors occur.

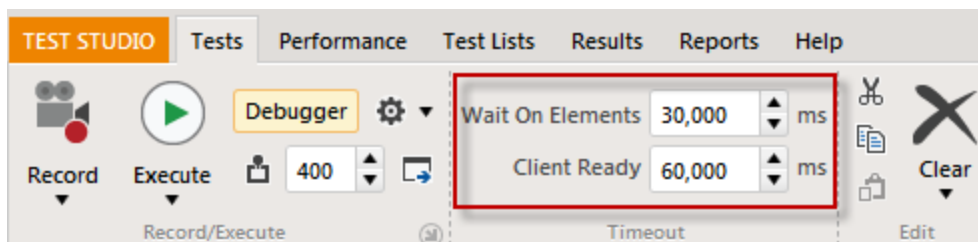


Standalone version



VS plugin

7. An easy way to change the default Timeouts for **Wait on elements** and **Client ready**.



---

## TEST LISTS

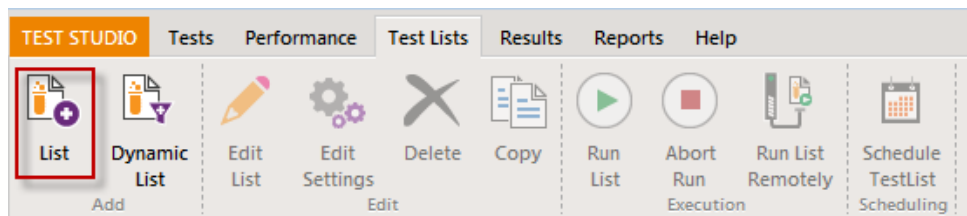
In the Standalone version you can execute one or more tests through a Test List. There are two types of Test Lists: Static and Dynamic.

- A Static Test List contains a fixed, predetermined list of tests.
- A Dynamic Test List contains a list of tests that is dynamically generated upon execution based on test properties.

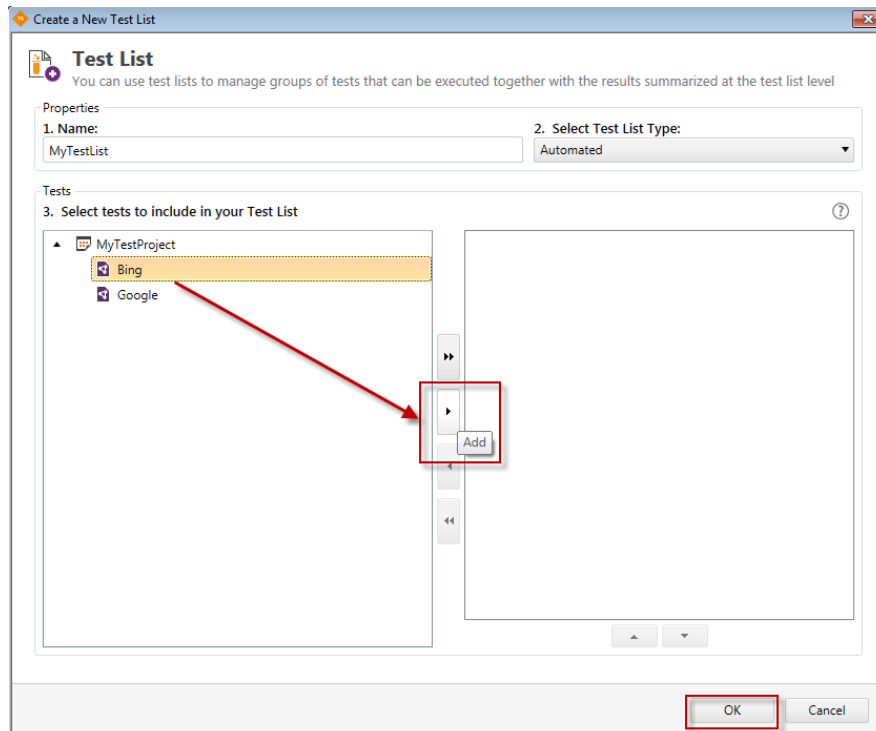
---

## STATIC TEST LIST

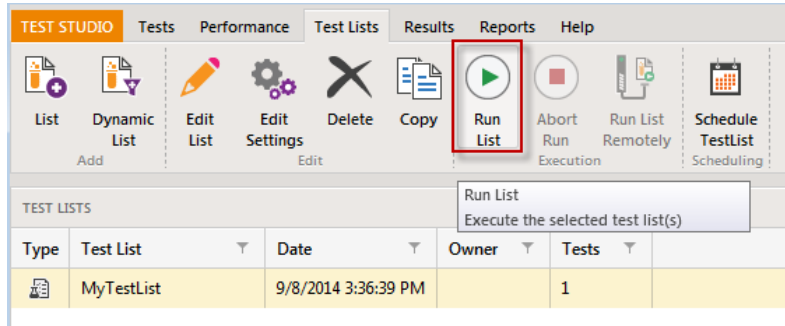
1. Click the **Test Lists** tab.
2. Click **List** in the **Add** ribbon.



3. Name the Test List.
4. Add the test(s) to the list from left to right.

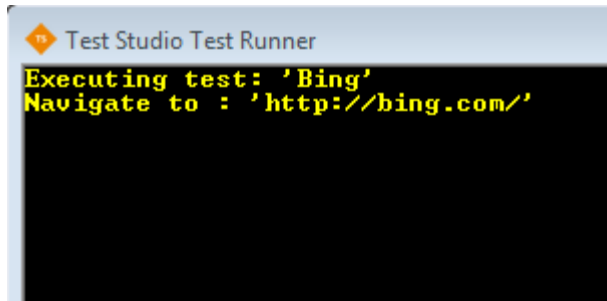


- Click **OK** to save the new Test List.
- Click **Run List** in the **Execution** ribbon.

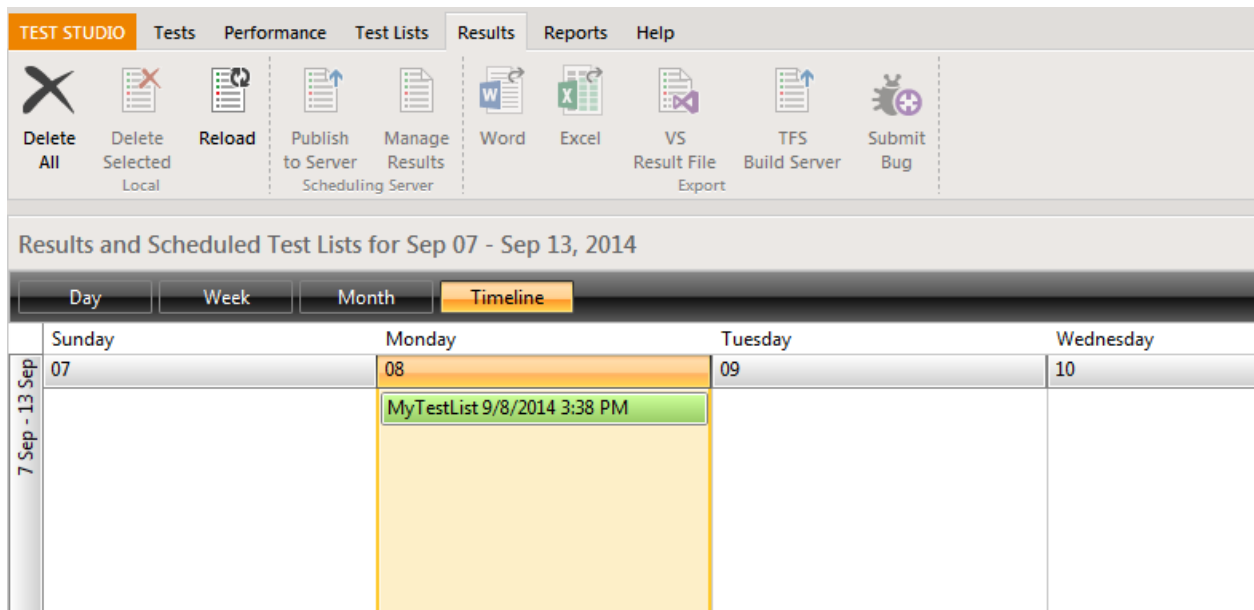


Select multiple Test Lists using the Shift key or Ctrl + Click, then click Run List to execute those lists in their order of selection.

- The Test Studio Test Runner launches first in a command prompt window. This calls each test in the list.



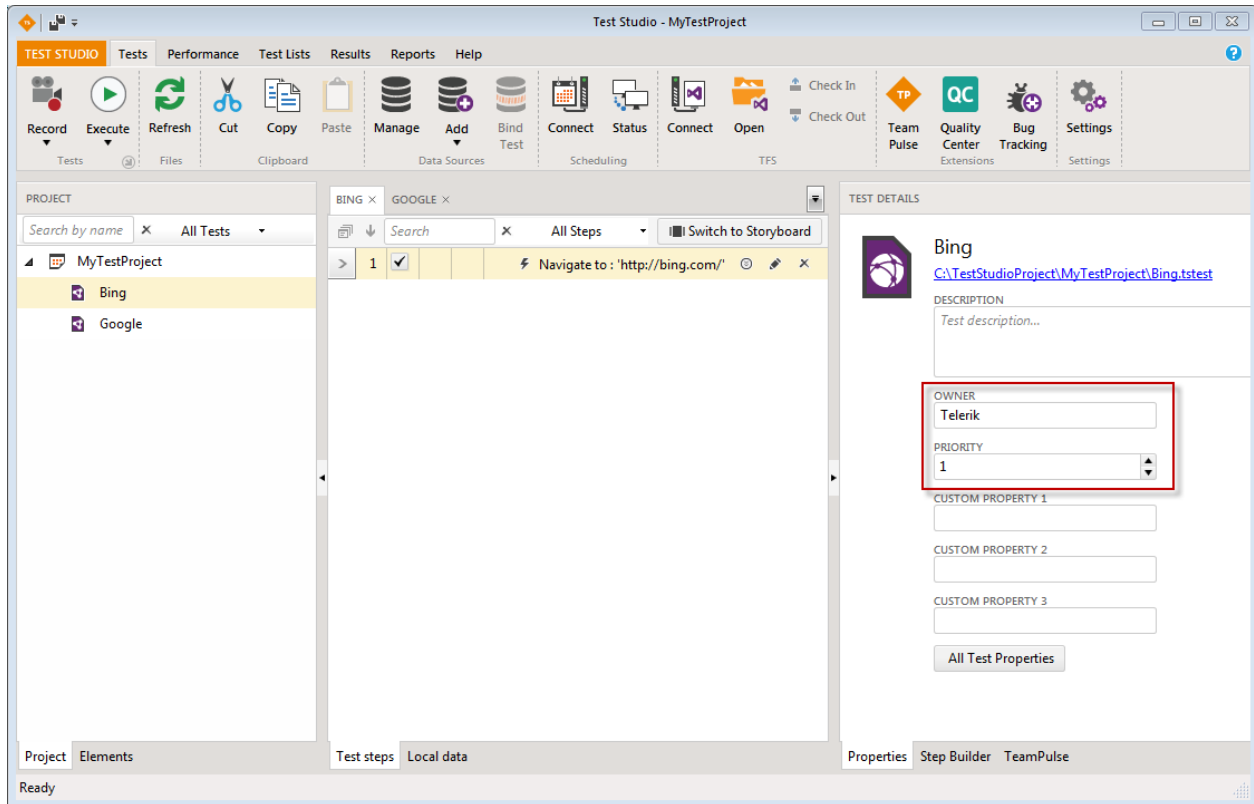
- A browser window or WPF app opens and each test executes in sequence. Upon completion, the Results tab opens.



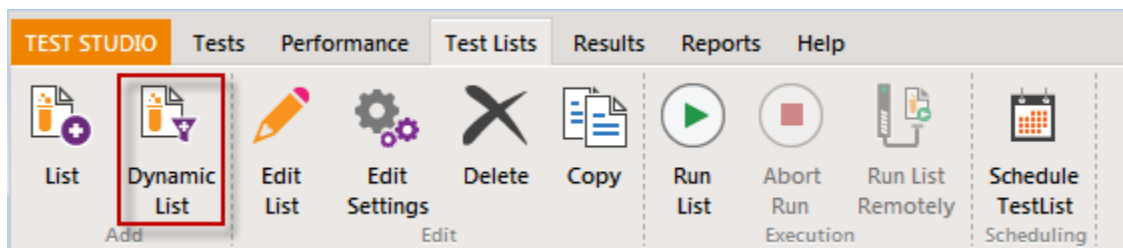
- To view the test results, double click the test result entry in the Timeline view (MyTestList in this example).

## DYNAMIC TEST LIST

1. Click the Project tab. Select a test and refer to the Test Details pane on the right. Below I have set Owner to Telerik and Priority to 1.



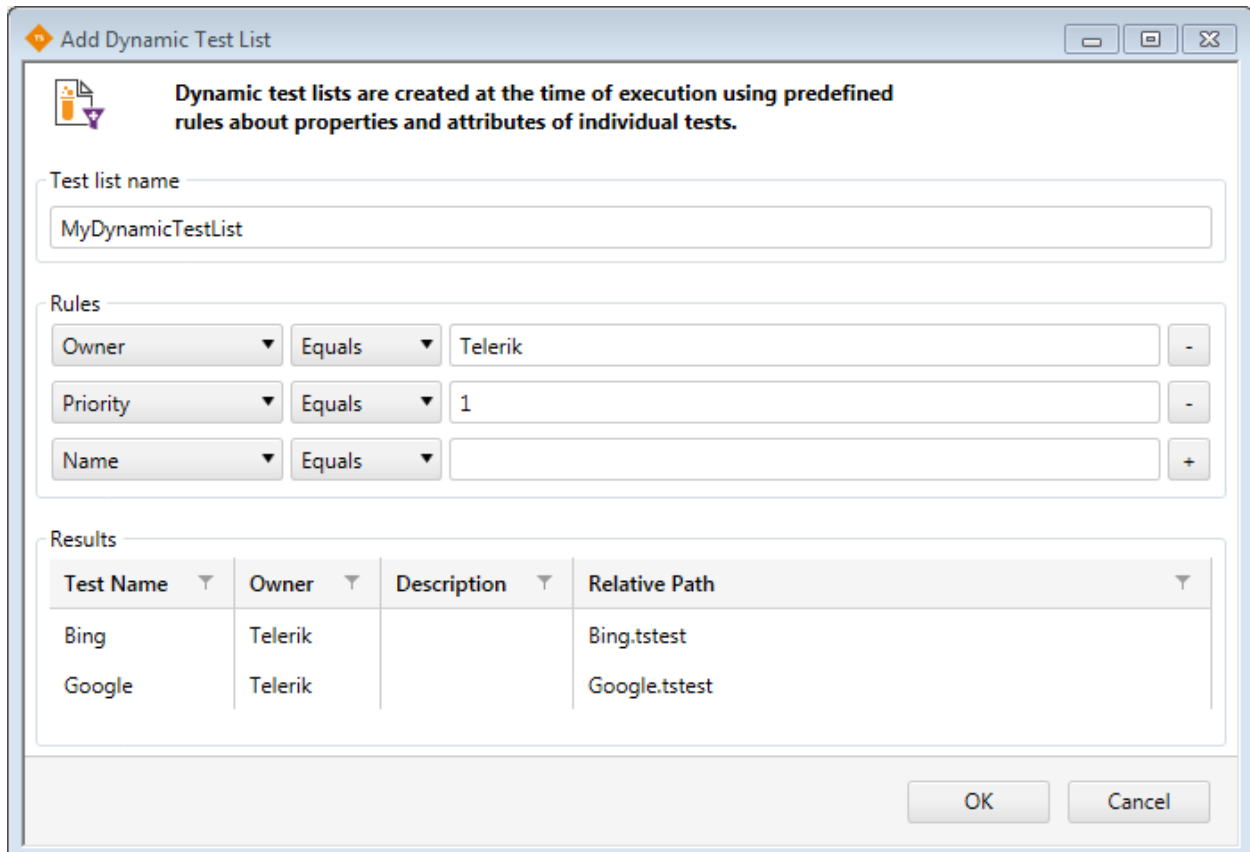
2. Click the Test Lists Tab
3. Click Dynamic List in the Add ribbon.



4. Name the Test List.



5. Craft one or more Rules to filter on specific criteria, clicking plus button after each one. The current results are displayed real-time.



6. Click OK to save the new Test List.

**NOTE:** Each time you click Run List in the Execution ribbon, Test Studio dynamically queries the project and executes the tests that meet the criteria of the Rules.

## TEST RESULTS

The Test Results panel allows you to traverse test execution results, drilling down to the individual test step and back up again to the test list level.

1. To see the results for a test list, double-click the result in the calendar.
2. The **Test Results** panel appears on the right. The test results panel shows results for the entire test list.
3. The **Passed/Total** column shows that all steps of both tests passed. Double-clicking a test in this list drills down to show the steps of that test.

The screenshot shows the 'Results and Scheduled Test Lists for Friday, January 16' interface. On the left is a calendar view for Friday, January 16, with a time slot from 3 PM to 5:30 PM highlighted. On the right is the 'Test Results' panel. At the top of the panel, it says 'Bundle Tests' and 'Fail - 1 tests passed out of total 2 executed. Executed on 'ITTODOROV' machine.' Below this is a table with the following data:


Test Path	Browser	End Time	Passed/Total	Result
drag.tstest	InternetExplorer	1/16/2015 2:45:44 PM	2/2	Passed
WebTest.tstest	InternetExplorer	1/16/2015 2:45:50 PM	2/3	Failed











4. Double-click a test in the Test Results view to see the result of each test step. The bread crumb trail at the top of the panel now shows the test list followed by the test name.

The screenshot shows the 'Test Results' panel with a breadcrumb trail at the top: 'Search 2 Tests - Test 1 > Bing Steps'. Below the breadcrumb, it says 'Fail - 4 steps passed out of total 5 executed.' Below this is a table with the following data:

	Description	More	Result
1	Navigate to : 'http://www.bing.com/'		Passed
2	Click 'NopeItSSpan'		Passed
3	Click 'ThIdH4942426141229098AmpPid17AmpW239A...		Passed
4	Click 'IoIMditLink'		Passed
5	Connect to pop-up window : 'http://blogof.francesc...		Failed
6	Keyboard (KeyPress) - C (1 times) on 'SbFormQText'		Failed
7	Click 'ThIdH4942426141229115AmpPid17AmpW362A...		Failed

5. Click the top level in the bread crumb trail to jump directly back to the test list result.

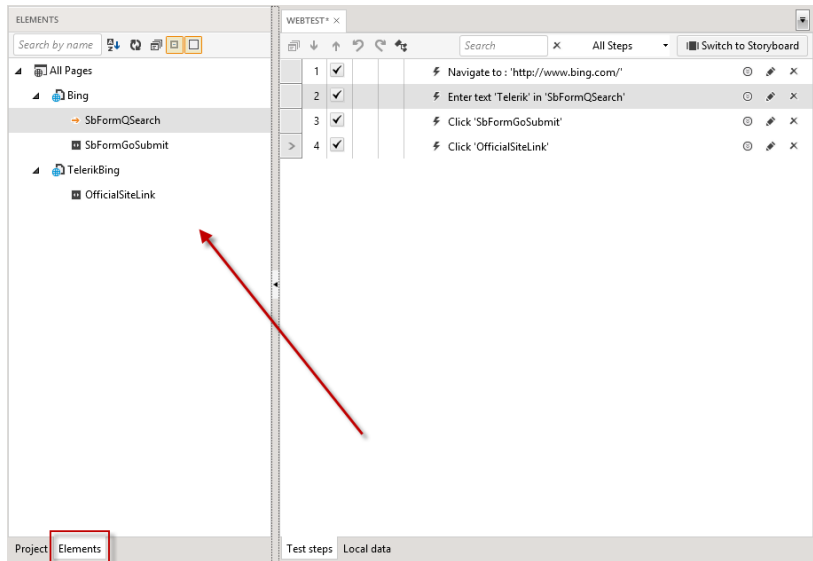
If the test contains a Test as Step,  appears next to the Test as Step. Double click that step to view the step results for that Test as Step.

Test Results			
Search 2 Tests - Test 2		DuckDuckGo Steps	
Fail - 2 steps passed out of total 3 executed.  			
	Description	More	Result
1	Navigate to : 'http://duckduckgo.com/'		
2	Keyboard (KeyPress) - Enter (1 times) on 'SearchButt...		
3	Execute test 'Bing.tstest'		
4	Click 'NETUILink'		
5	Click 'TestStudioH2Tag'		
6	Click 'SupportLink'		
7	Click 'DocumentationLink'		

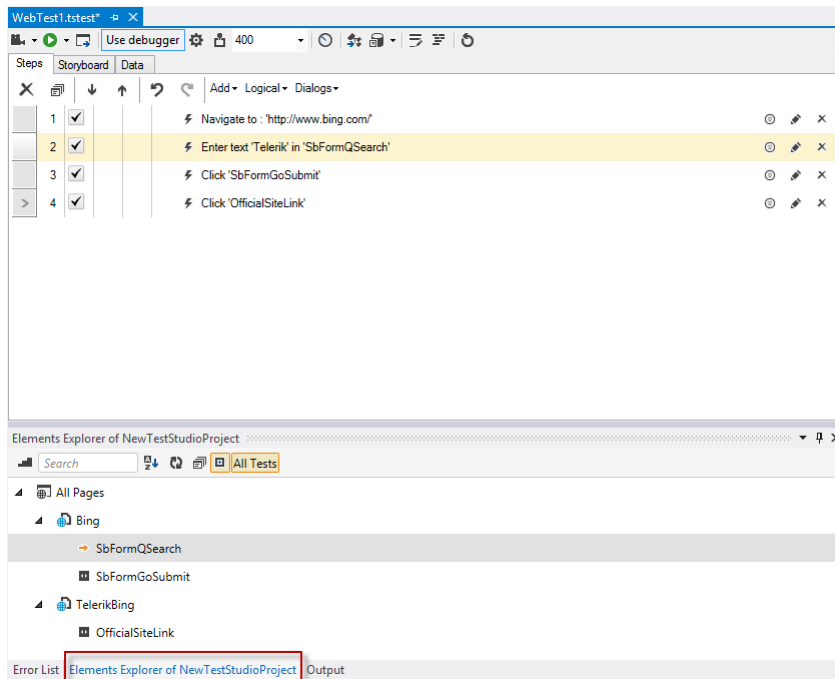
## ELEMENTS EXPLORER

The Elements Explorer displays a tree of elements, the Elements Explorer only contains elements you want to use in your tests. Also, the elements in the tree view have properties that are more specific to testing. Although elements may be used in several tests and test steps, each element is shown only once in the Elements Explorer.

You can find the Elements Explorer under the Elements tab on the left bottom pane.



### Standalone version

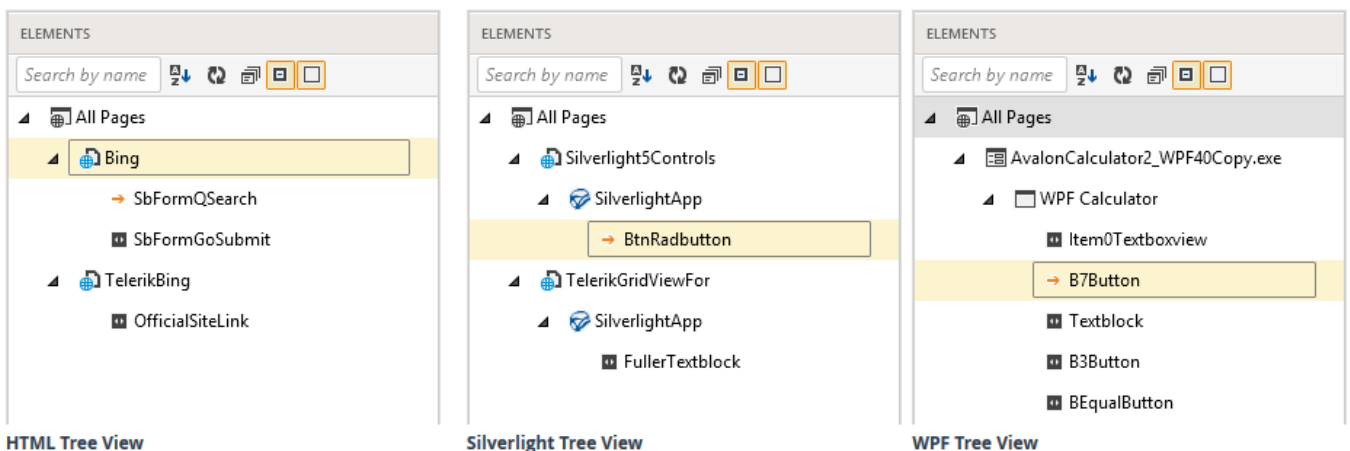


### VS Plugin

The Elements tab, 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 the following buttons:

- **Search** - search the Explorer based on the element Friendly Name property.
- **Sort** - organize the elements in an ascending or descending order, or clear the sorting.
- **Refresh** - refresh the display of elements in the Explorer. You seldom should have to do this because Test Studio normally refreshes the window properly.
- **Enable/Disable Highlighting** - control the highlighting of elements on the recording surface as they are selected in the tree view. When enabled, an element highlighted in the Elements pane will also be highlighted in the active browser.
- **Expand/Collapse** - show or hide all elements under their respective page nodes.
- **This Test/All Tests** - show elements for the currently loaded test only, or the elements for every test in the project.



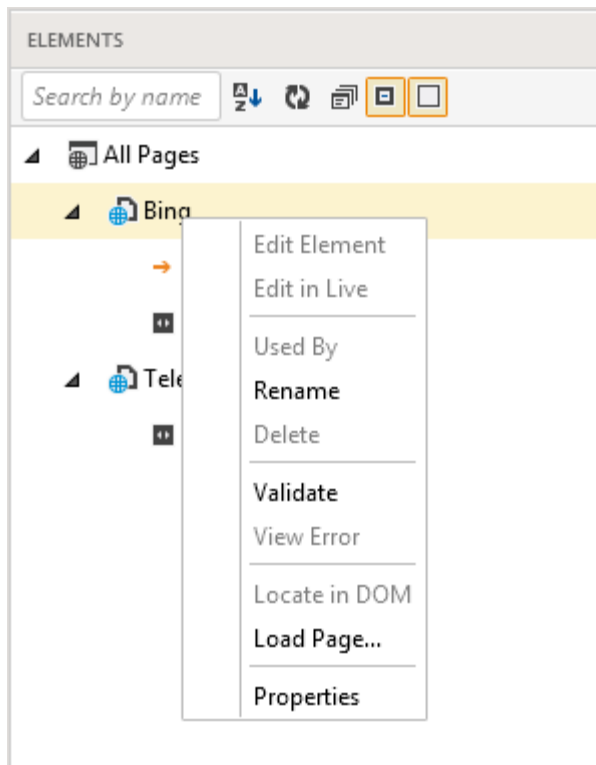
- The HTML tree view is organized by **Page > Frame > Test Regions > Element**.
- The Silverlight tree view is organized by **Page > Frame > SilverlightApp > Element**.
- The WPF tree view is organized by **Application > Window > Element**.

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 a Page node to see a context menu with these active choices:

- **Validate** - validate all elements in the page node against the currently loaded page. Requires the page to be loaded in the recording window. Results indicated with green checks and red X's.

- **Rename** - alters the Friendly Name.
- **Load Page** - loads the URL to which the page belongs in the recording window.
- **Properties** - makes the Properties pane active.



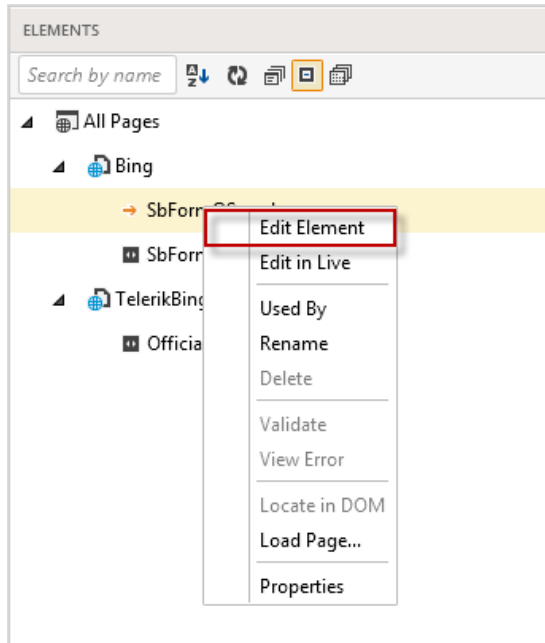
Each Element node has a context menu with these active choices:

- **Edit Element** - loads the Find Element menu to choose where and how to locate this element in your web page or application.
- **Edit in Live** - locate the element in the page currently loaded in the recording browser. Only available during recording.
- **Used By** - loads the Test Step Selector and displays all tests and their steps that contain this element.
- **Validate** - validate all elements in the page node against the currently loaded page. Requires the page to be loaded in the recording window. Results indicated with green checks and red X's.
- **Rename** - alters the element's Friendly Name (an easy way to identify the node). This name will also be used for code generation as a variable name or a collection indexer.
- **Delete** - remove the element from the Explorer. The element must not be linked to any test steps.
- **View Error** - show the erroneous find logic for an element that cannot be found.
- **Locate in DOM** - jump to this element's position in the DOM when the recording window is loaded.
- **Load Page** - loads the URL to which the element belongs in the recording window.
- **Properties** - makes the Properties pane active.

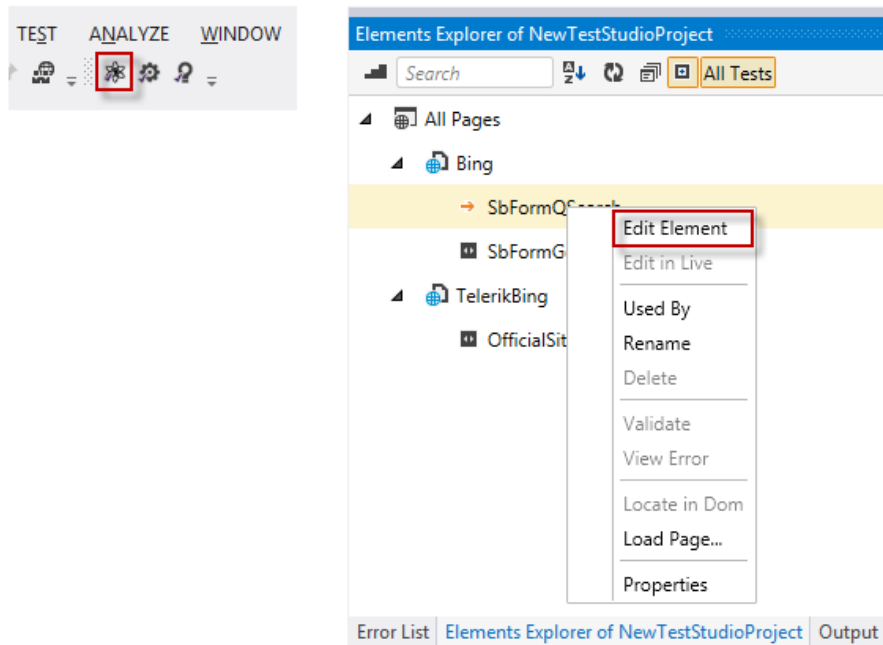
## CHANGE HOW AN ELEMENT IS FOUND

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 Test Studio uses to find that specific element on the web page.

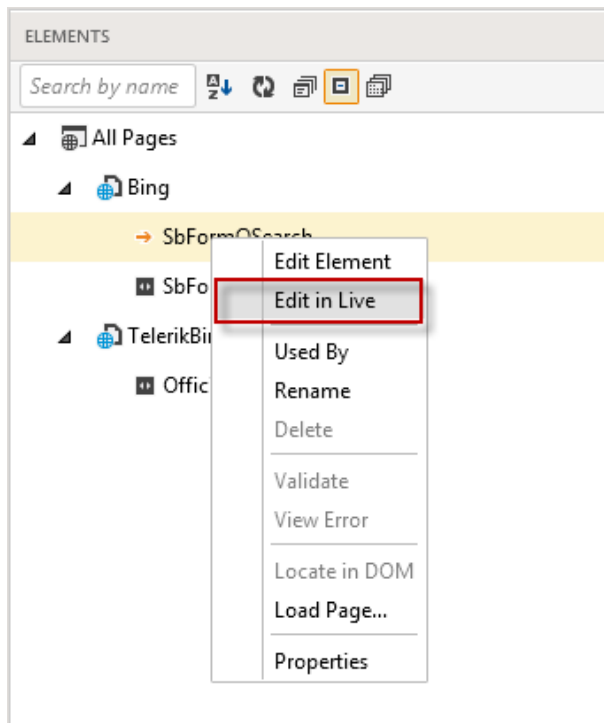
To change how an element is found, right click on the element in the Explorer and select Edit Element.



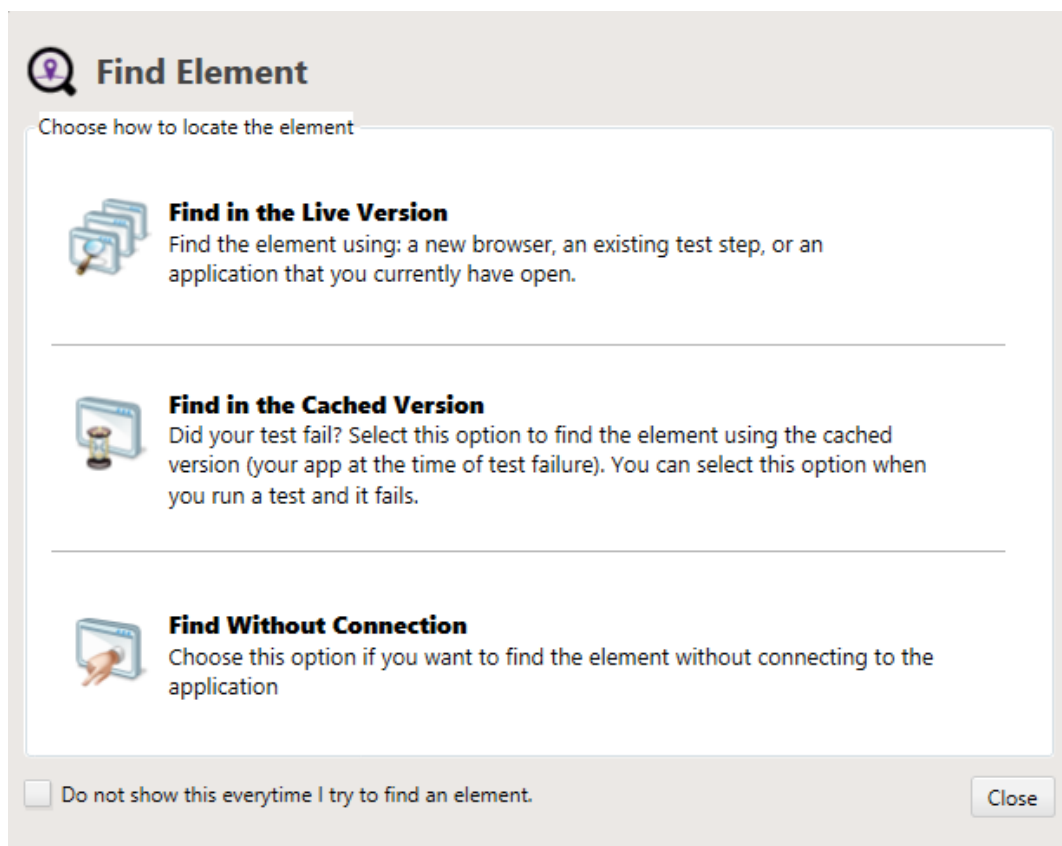
In the VS plugin, click the Show Element Explorer icon in the toolbar and locate the Explorer at the bottom of the screen.



Select **Edit in Live** to directly open the Find element dialog for the page open in the current recorder.



The **Find Element** splash screen appears. You have three options for how to locate the element:





**Find in the Live Version** - find the element using the latest version of a new browser window, an existing test step, or an application that you currently have open.

- **New Browser** - this will launch a new instance of your application. You may need to manually navigate to the element. Click Browse & Navigate to proceed.
- **Existing Test Step** - use an existing step from a test to get to the element. Click Choose Test Step to proceed.
- **Current Page** - select where the element is available from a list of currently running browser instances or WPF applications. Click Go to proceed.

**Find Element**

Connect to the Live Version

**New Browser**  
This will launch a new instance of your application. Depending on where your element is located, you may need to manually navigate to the element. Make sure that recording is paused to avoid adding extra steps to your test. **Browse & Navigate**

**Existing Test Step**  
Use an existing step from a test to get to the element. **Choose Test Step**

**Current Page**  
Select where the element is available:  
 **Go >**

**← Back**

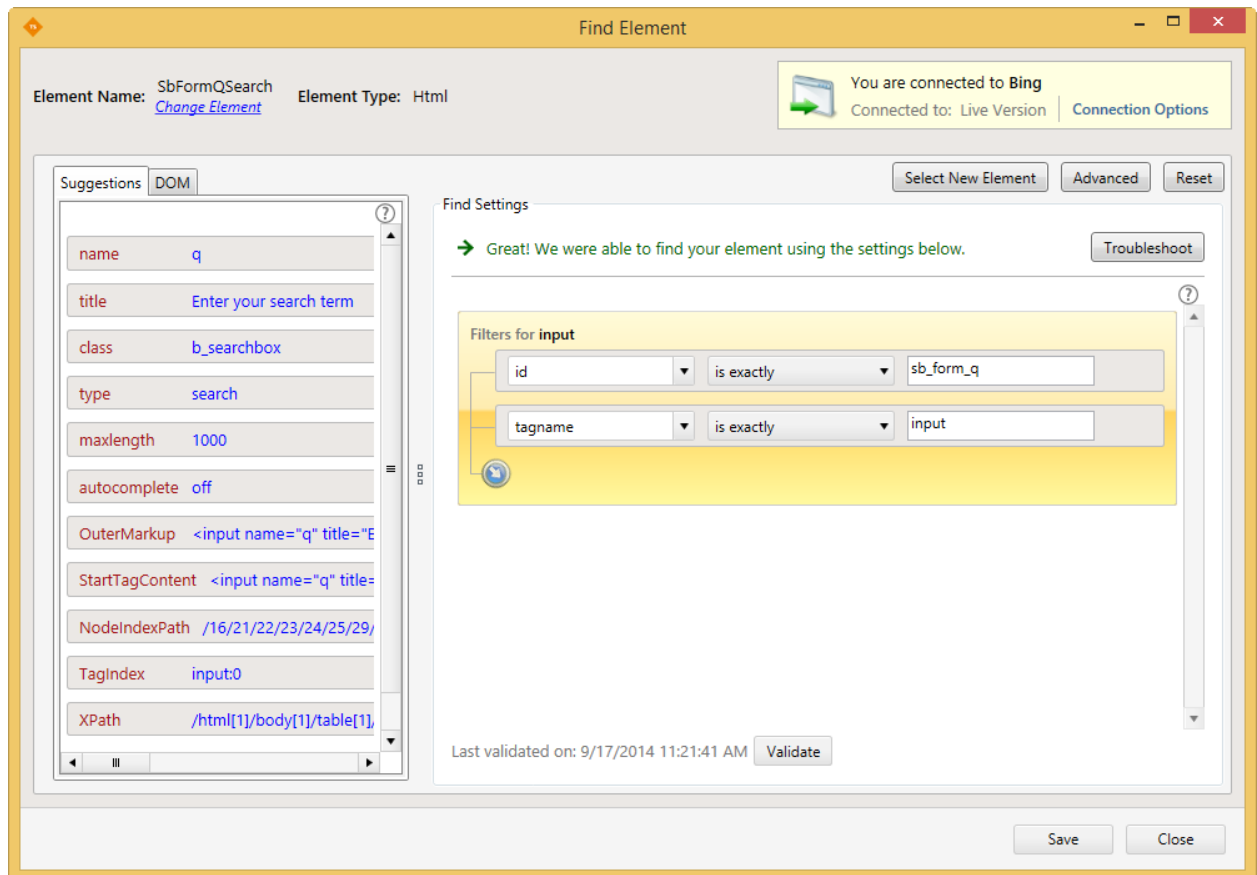
Do not show this everytime I try to find an element. **Close**

**Find in the Cached Version** - if your test failed, you can find the element using the cached version of the application at the time of failure. Accessible only through the **Resolve Failure** tab in the Step Failure Details.

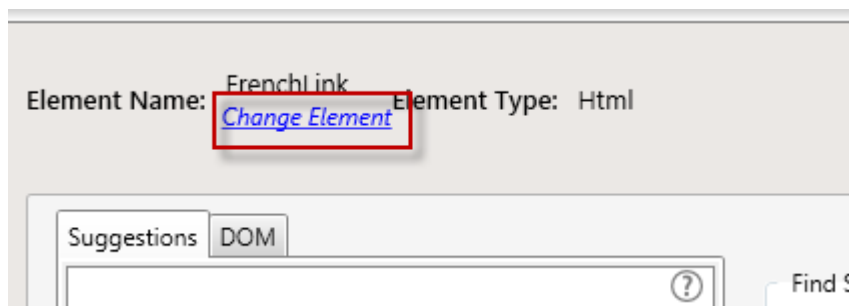
**Find Without Connection** - choose this option to find the element without connecting to the application.

**NOTE:** To skip this splash screen the next time you load the Find Element dialog, check the box at the bottom and click Close.

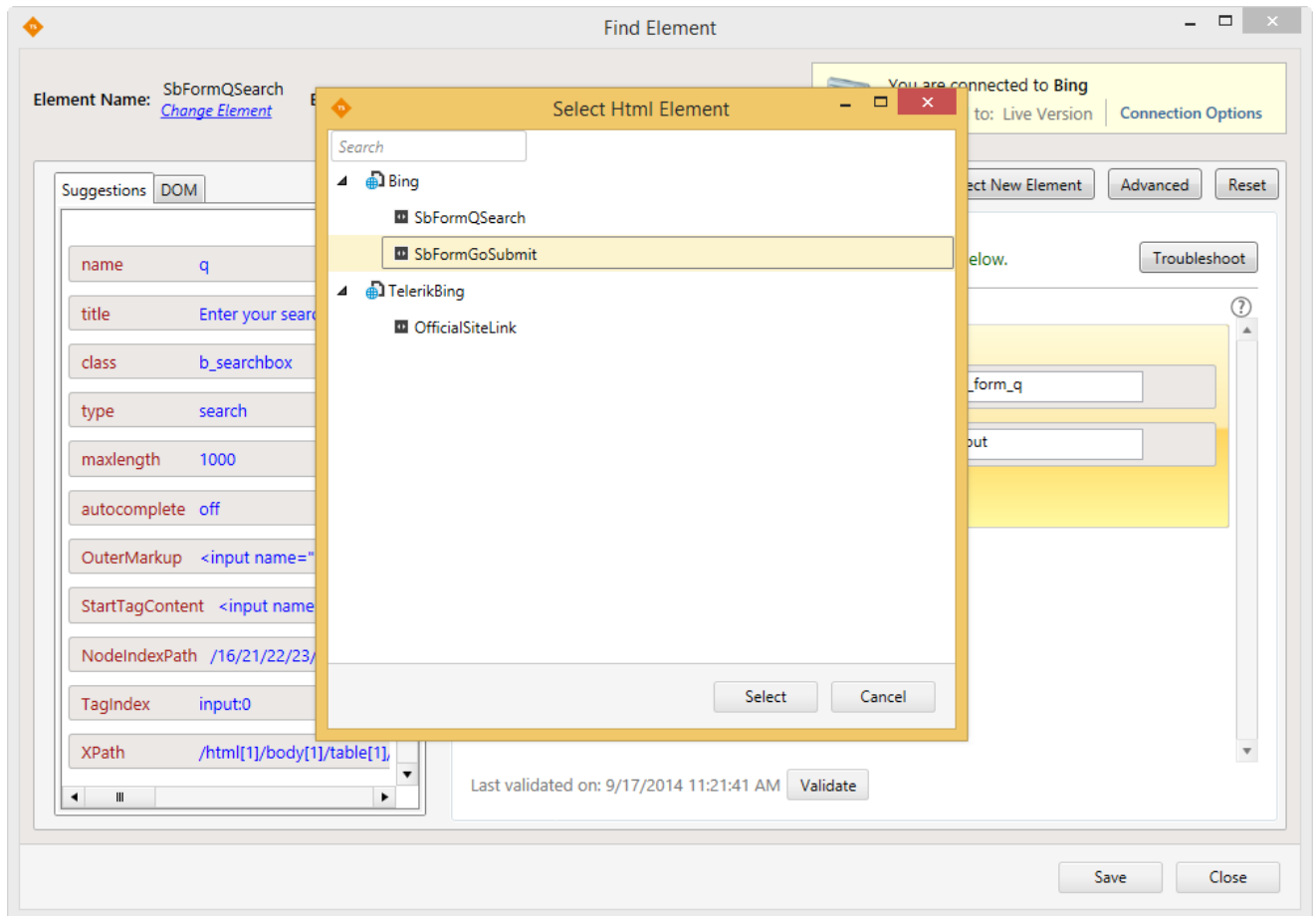
The Find Element dialog appears. The Element Name, Element Type, and Connection Status are at the top.



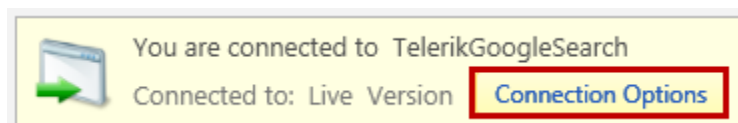
The **Change Element** link opens the **Select New Element** dialog.



In the Select Html Element dialog, you can open the Find Element dialog for a different element in the Elements Repository.

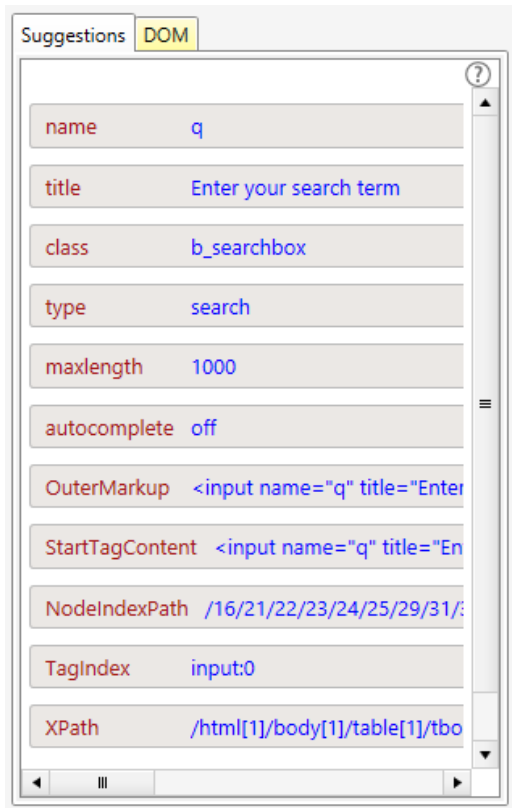


The **Connection Options** button takes you back to the **Find Element** splash screen.

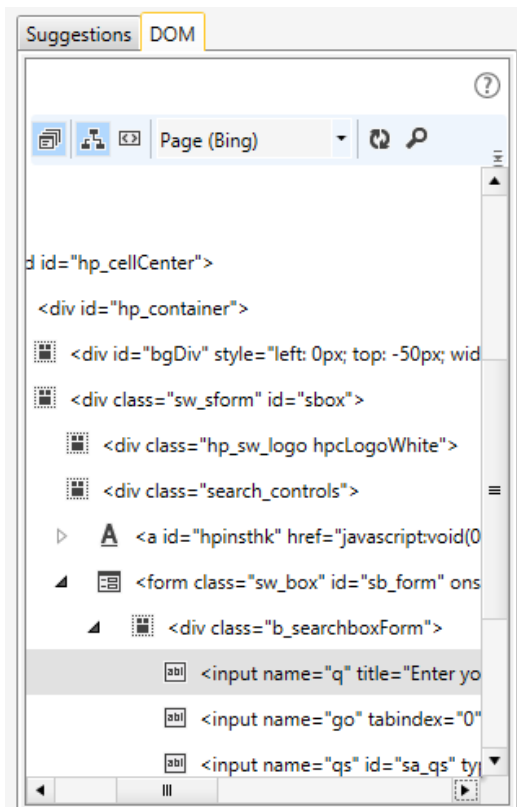


The **Suggestions** and **DOM** views are on the left side.

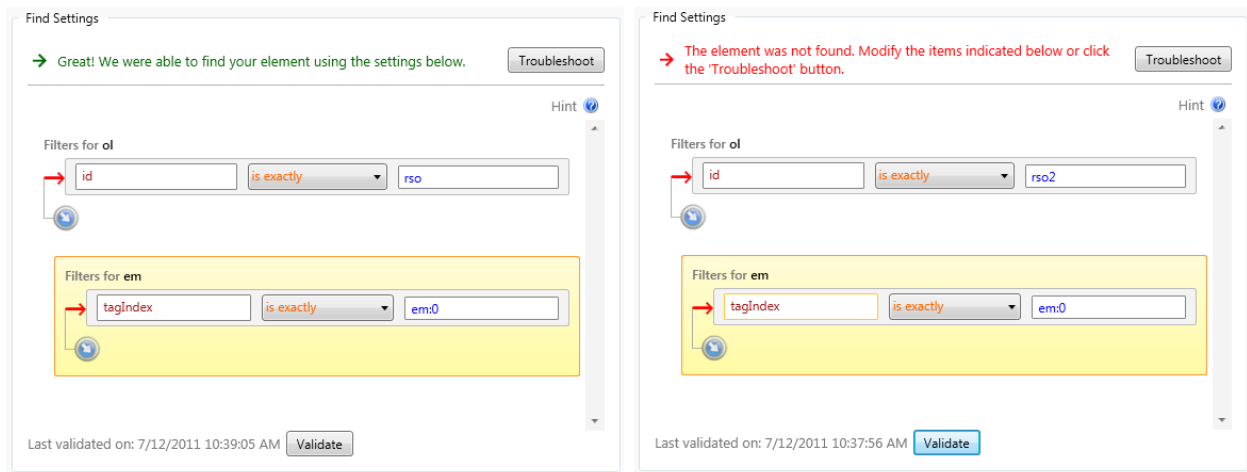
- **Suggestions** - these are the suggested items to help you find the element in the application. Click an item to add it to your Find Settings.



- **DOM** - use the DOM as a reference when creating your Find Settings. This view is helpful in determining where your element is located relative to the DOM tree.

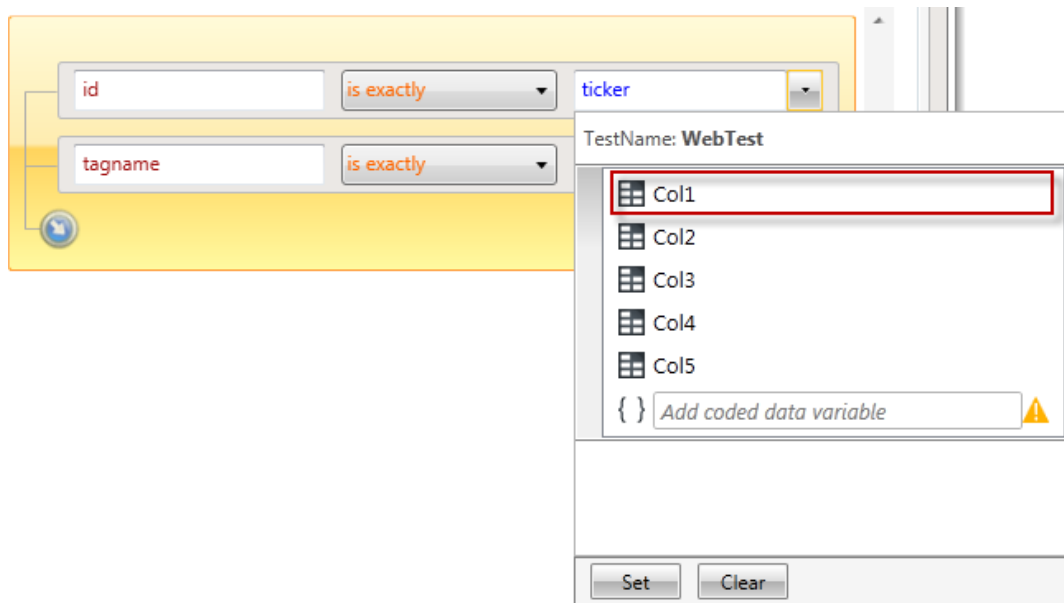


The **Find Settings** view is on the right side. You can edit these settings by typing in new properties, selecting a new modifier in the drop-down menu, or changing the values. Click Validate to confirm whether the element can be found using the current Find Settings.

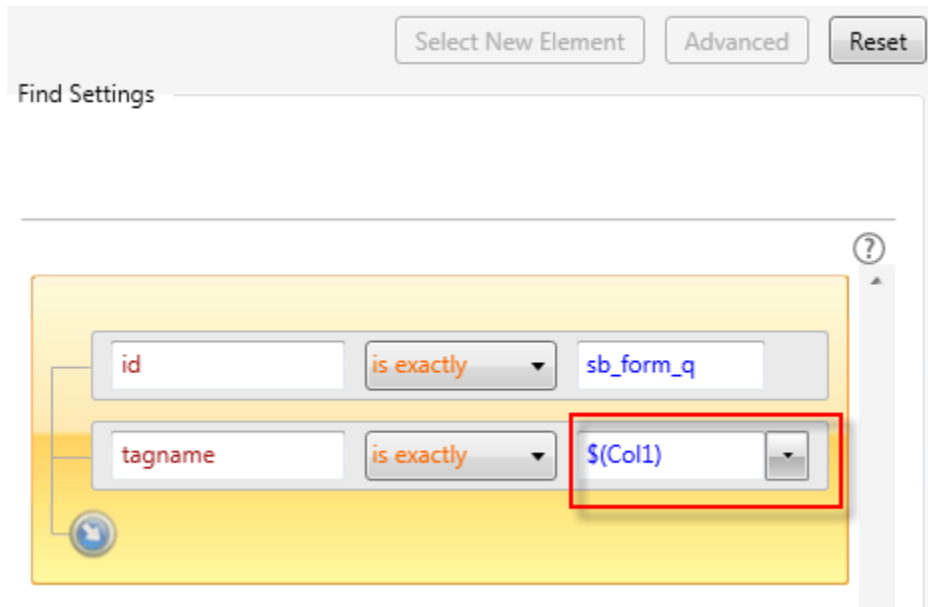


## DATA-DRIVEN FIND LOGIC

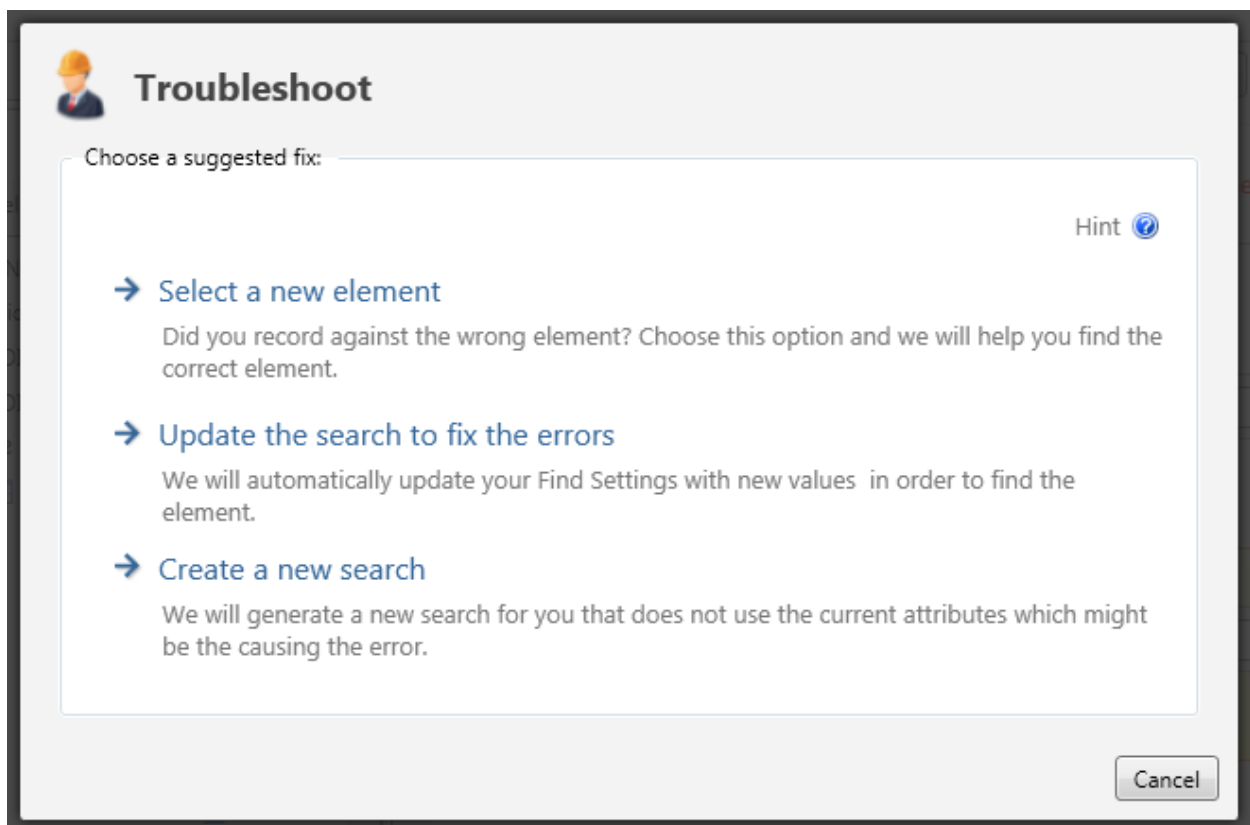
Here, you can also data drive the element find expression. If your test has an attached data source, the value fields of the find expressions for your elements will include a drop-down list displaying columns from your data source.



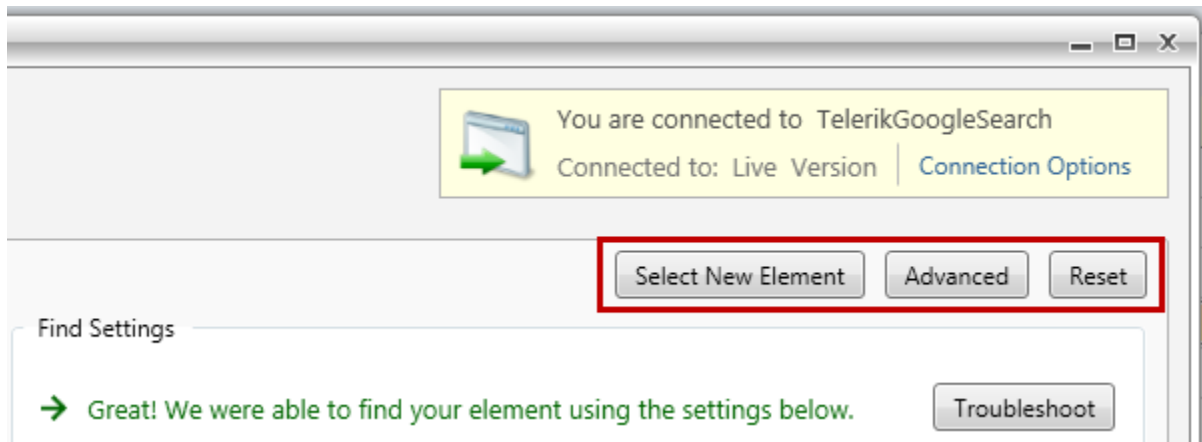
Selecting a column databinds the column to the value of the find expression rule.



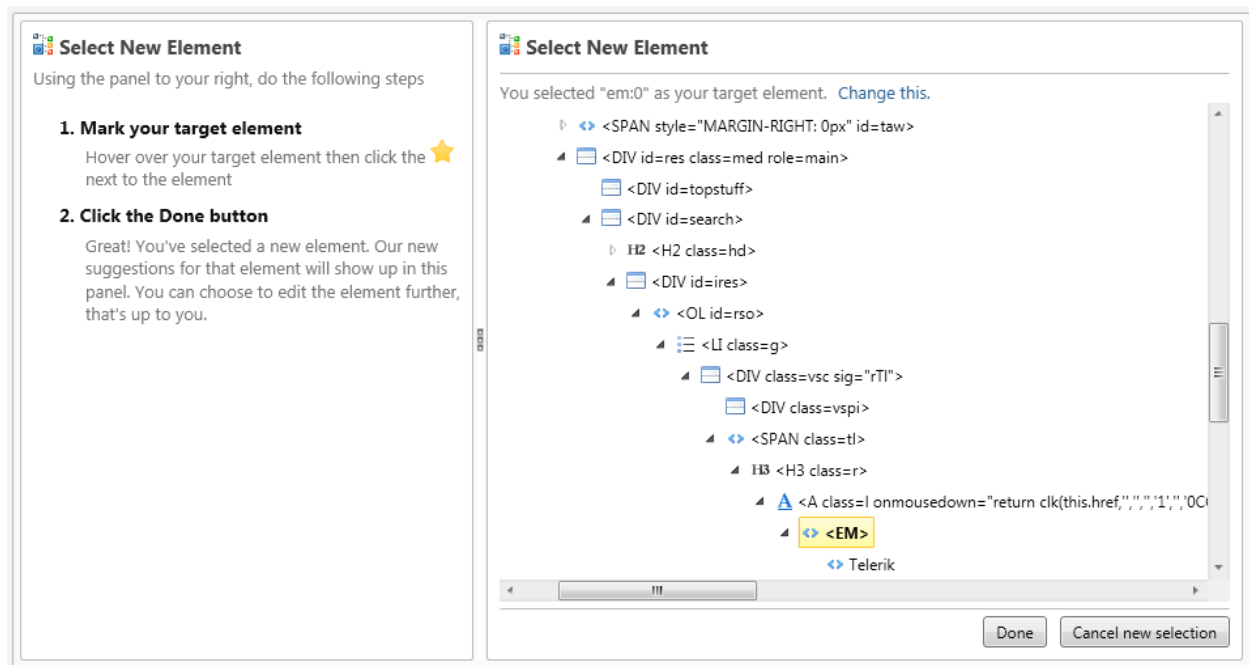
If the element was not found, click the **Troubleshoot** button. Choose a suggested fix from the Troubleshoot screen.



There are three additional buttons in the upper right of the Find Element dialog:



- **Select New Element** - choose this option if the incorrect element was selected for this step. This will clear your current settings and Test Studio provides new filters for the new element.



- **Advanced** - edit the Find Settings using a string-based expression builder.

Advanced

**Reference:**

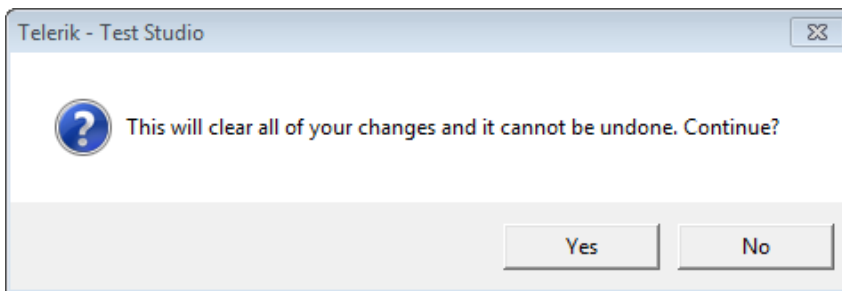
- ~ = Contains //foo=~bar : foo 'contains' bar
- ! = NotContain //foo=!bar : foo 'does not contain' bar
- ^ = StartsWith //foo=^bar : foo 'starts with' bar
- ? = EndsWith //foo=?bar : foo 'ends with' bar
- # = RegEx //foo=#ba\* : foo 'matches regex' ba\*
- , = And //include additional search
- | = Then //narrow search within a particular element

**Please type in your custom find expression.**

Use the Reference above to help create your new expression.

id=rso,|tagIndex=em:0

- **Reset** - restores fields to their original settings.



Once you've confirmed you are targeting the correct element and it is correctly found, click Save and Close. If the modified element is used by multiple automation steps, you are prompted to select the steps you want to persist changes to.

