Web Automated Testing using Selenium IDE

How To Document

February 2017

Prepared by: InterWorks

Contents

1 INTRODUCTION

This document introduces Selenium IDE, teaches its features, and presents commonly used best practices. The focus of this document is core functionalities and benefits of using Selenium IDE tool. Selenium IDE can be used to easily and quickly create automated tests for all web-based applications. These guidelines should help create and maintain automated tests using simple predefined Selenium features and commands.

Test automation with Selenium IDE has specific advantages for improving the long-term efficiency of a software testing process.

Selenium IDE test automation supports:

- Frequent regression testing
- Rapid feedback to developers
- Virtually unlimited iterations of test case execution
- Support for Agile and extreme development methodologies
- Disciplined documentation of test cases
- Customized defect reporting
- Finding defects missed by manual testing
- Exporting test suite results

2 WHAT IS SELENUIM IDE?

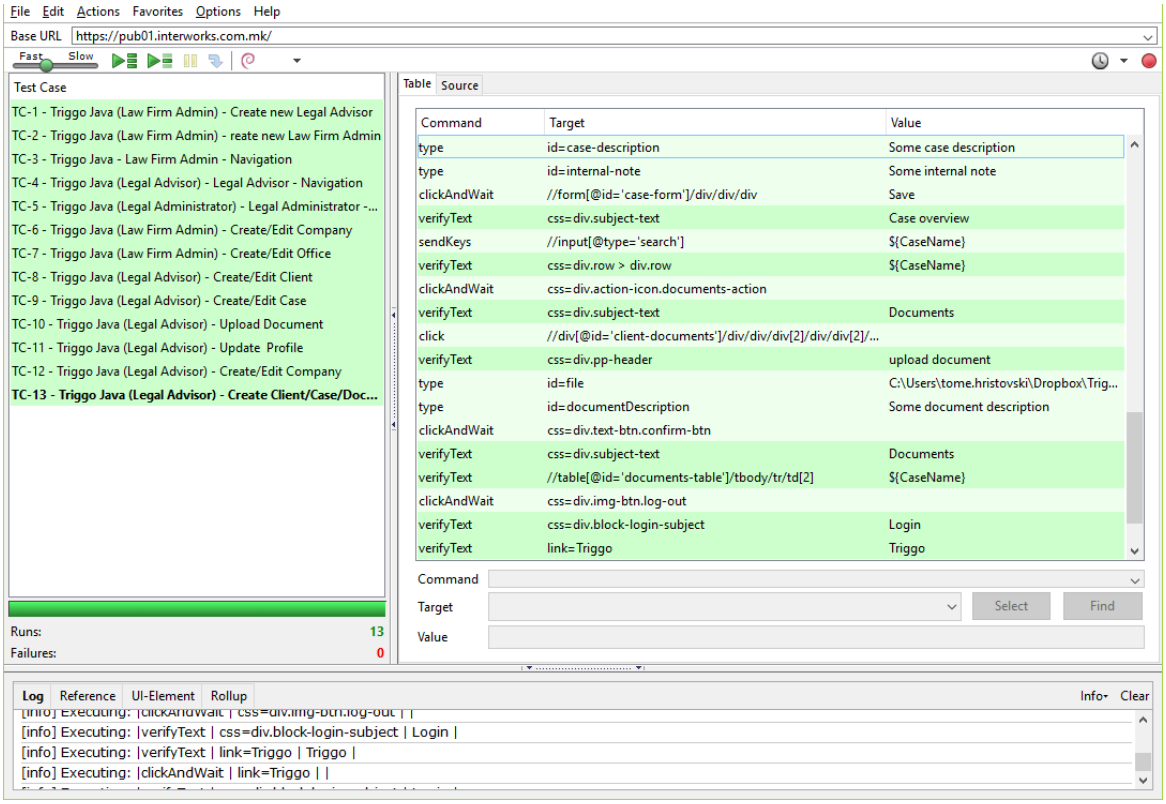
Many of today's software applications are written as web-based applications ran in an Internet browser. The effectiveness of testing these applications varies widely among companies and organizations. In an era of highly interactive and responsive software processes where many organizations are using some form of Agile methodology, test automation is frequently becoming a requirement for testing software.

Test automation means using a special software tool to run and control repeatable tests against an application and to compare actual with expected outcomes.

Selenium IDE is an integrated development environment for developing Selenium web based tests. It is implemented as a Firefox extension, and provides an efficient way to, develop, record and debug tests. The Selenium IDE includes the entire Selenium Core, allowing easily, quickly recording, and executing tests in the actual environment.

It also contains a context menu that allows selecting a UI element from the browser's displayed page and then selecting a Selenium command from a list with pre-defined parameters.

The tool simulates clicks and validation on a web page or a web application. It does that by interacting with previously recorded elements on the web page.



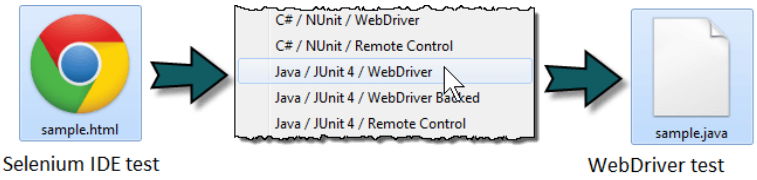
Selenium IDE main pane

3 SELENIUM IDE ARCHITECTURE & WORKFLOW

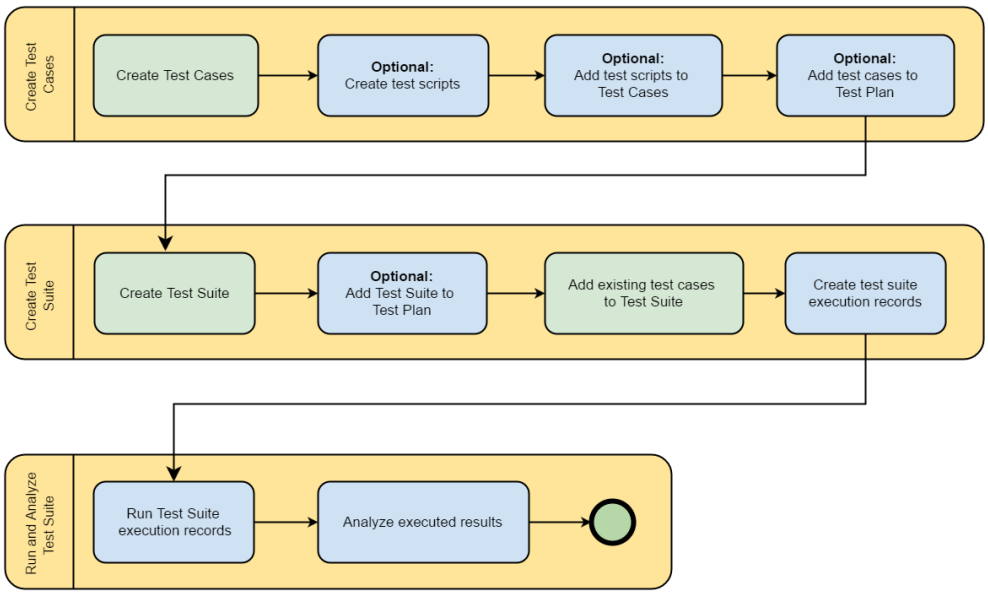
The tests in Selenium IDE are recorded in an HTML table based architecture using keywords. The code can be exported in different languages, like Ruby, Java etc.

Selenium IDE Architecture

Selenium IDE Workflow



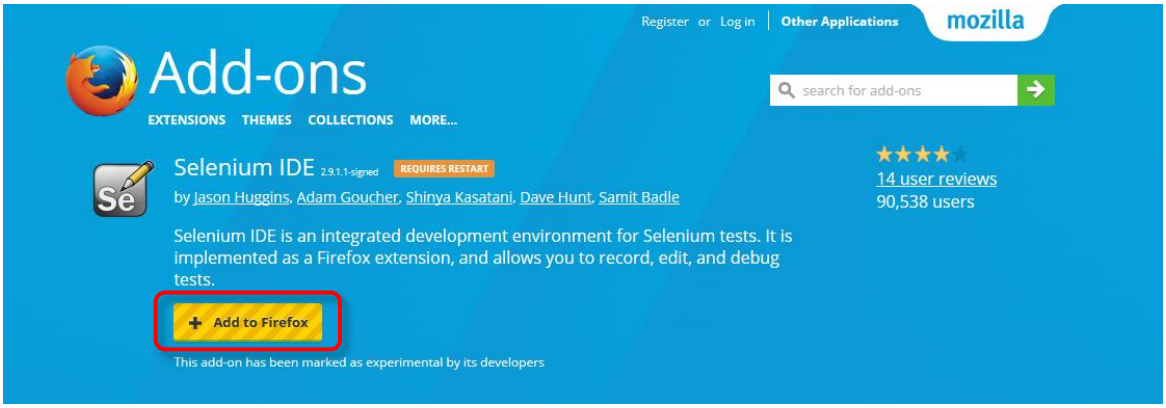
Exporting Selenium IDE tests



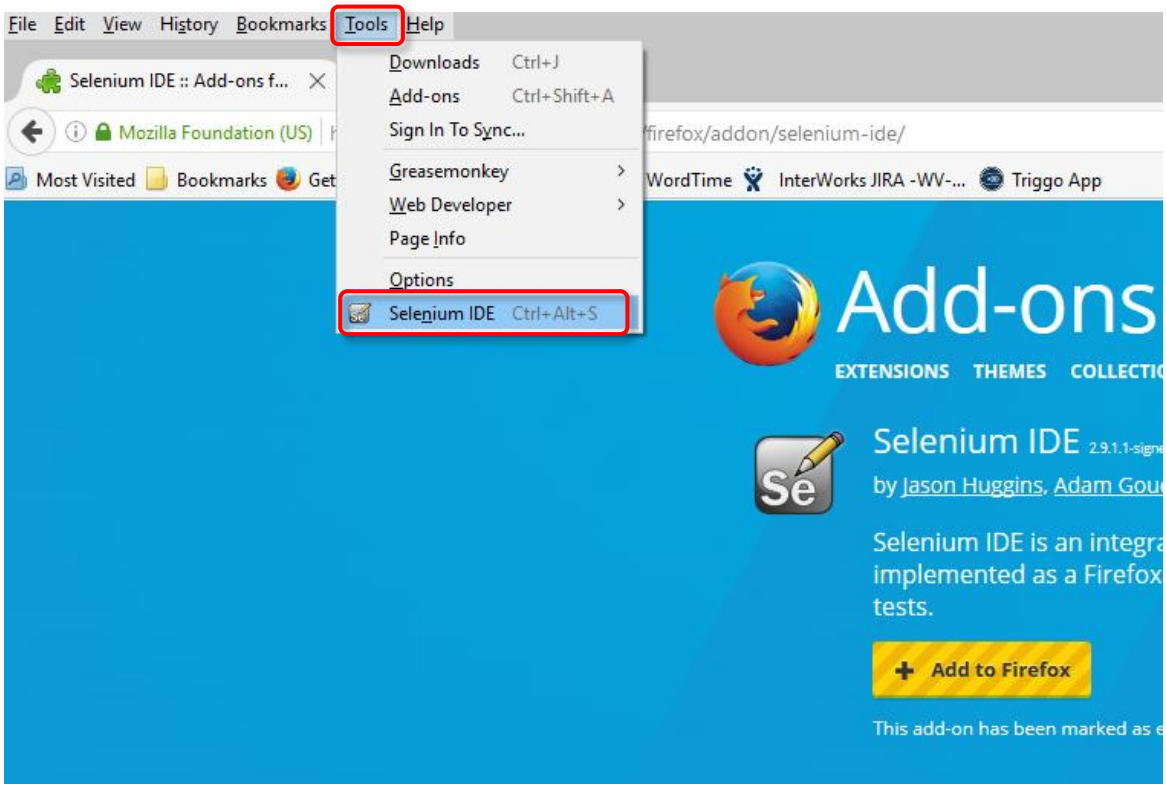
Create & Execute Workflow

4 INSTALLING THE SELENIUM IDE

Using Firefox, first download the IDE from the following downloads page.

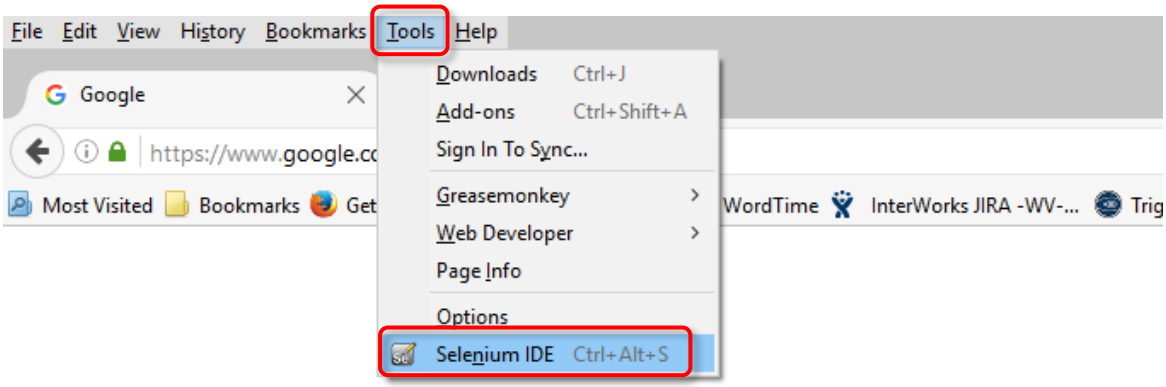


After installation of the Selenium IDE add-ons, restart Firefox. After Firefox reboots you will find the Selenium IDE listed under the Firefox Tools menu.

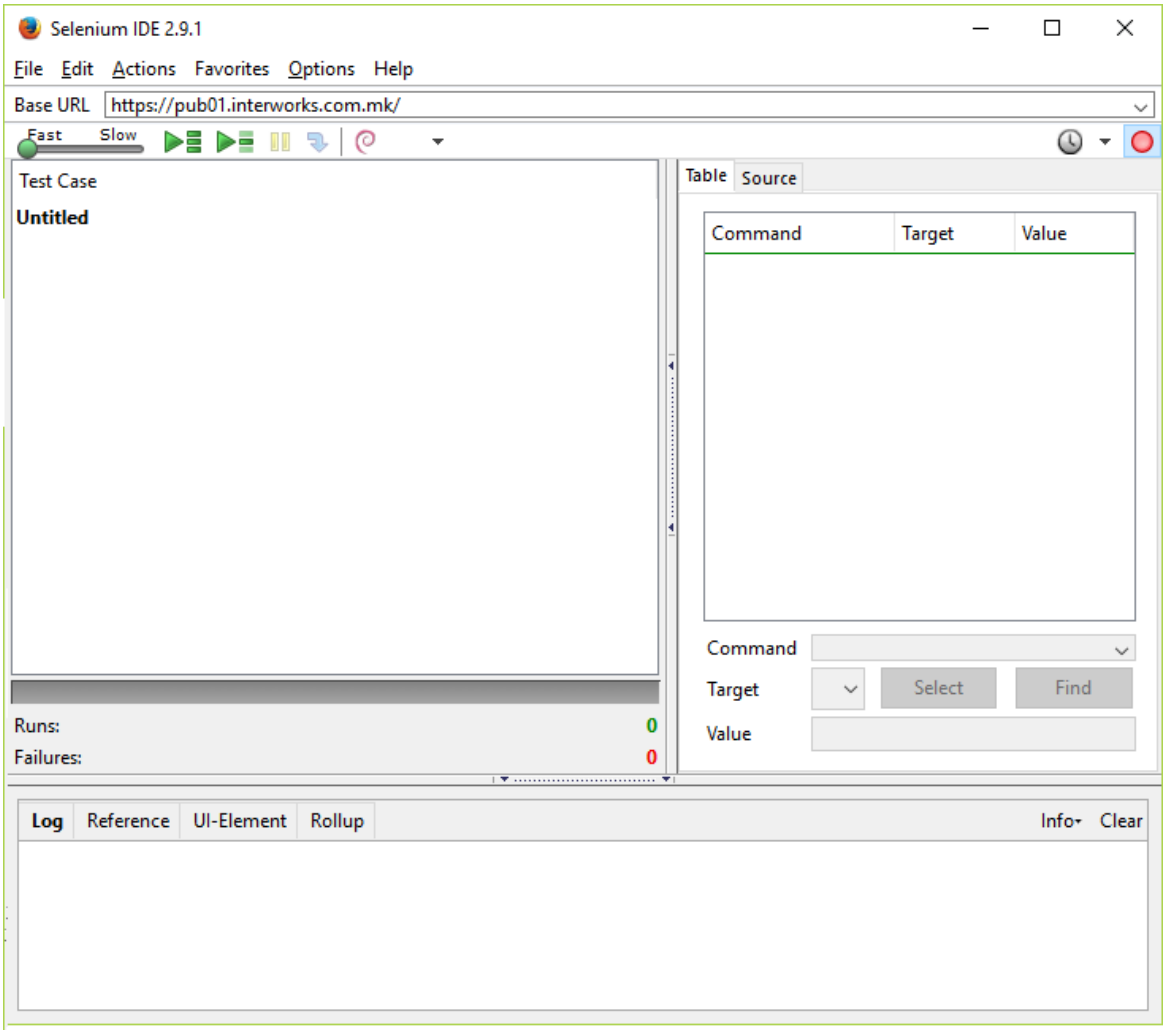


5 OPENING THE SELENIUM IDE

To run the Selenium IDE, simply select it from the Firefox Tools menu. It opens as follows with an empty script-editing window and a menu for loading, or creating new test cases.



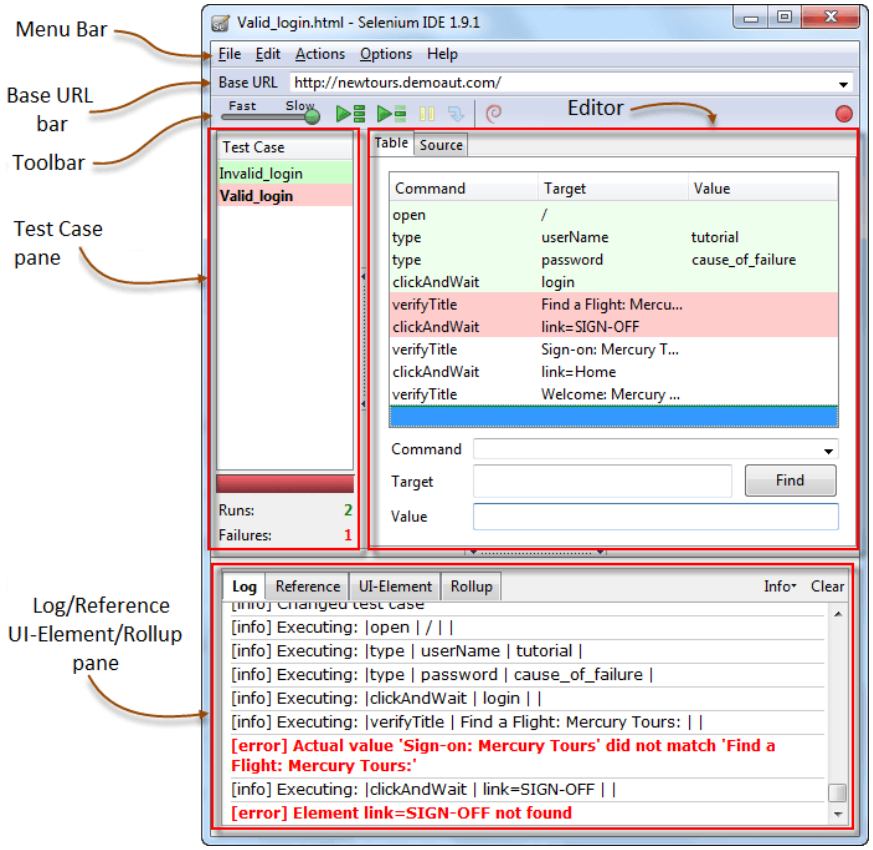
After opening the Selenium IDE, the new panel will be opened.



6 SELENIUM IDE FEATURES

Menu Bar

The File menu has options for Test Case and Test Suite. Using these, you can add a new Test Case, open a new or recent Test Case, save a Test Case and export Test Case in a different language. All these options are also available for Test Suite.

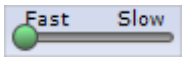


Main pane of Selenium IDE

The Edit menu allows to copy, paste, delete, undo, and select all operations for editing the commands in the test case. The Options menu allows changing the settings. You can set the timeout value for certain commands, add user-defined user extensions to the base set of Selenium commands, and specify the format (language) used when saving the test cases.

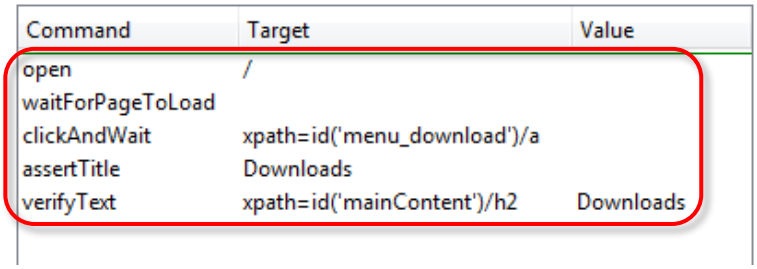
Toolbar

The toolbar contains buttons for controlling the execution of the test cases/suites, including a step feature for debugging the test cases. The right-most button (the one with the red-dot) is the record button.

	**Speed Control:** Controls how fast a test case runs
	**Run All:** Runs the entire test suite when a test suite with multiple test cases is loaded
	**Run:** Runs the selected test
	**Pause:** Allows stopping of a running test case
	**Resume:** Allows re-starting of a running test case
	**Step:** Allows "stepping" through a test case, used for debugging tests
	**TestRunner Mode:** Allows running the test case in a browser loaded with the Selenium Core TestRunner
	**Apply Rollup Rules:** This advanced feature allows repetitive sequences of Selenium commands to be grouped into a single action
	**Record:** Records the user's browser actions

Test Case Pane

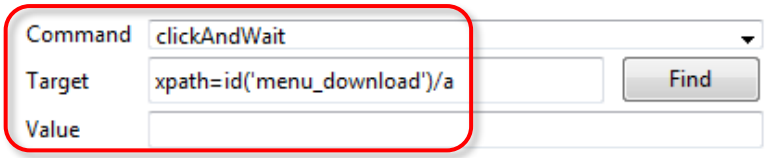
The script is displayed in the test case pane. It has two tabs, one for displaying the command and their parameters in a readable "table" format.



Command	Target	Value
open	/	
waitForPageToLoad		
clickAndWait	xpath=id('menu_download')/a	
assertTitle	Downloads	
verifyText	xpath=id('mainContent')/h2	Downloads

The other tab Source displays the test case in the native format in which the file will be stored. By default, this is HTML format. The Source view also allows one to edit the test case in its raw form, including copy, cut and paste operations.

The Command, Target, and Value entry fields display the currently selected command along with its parameters. These are entry fields where you can modify the currently selected command.

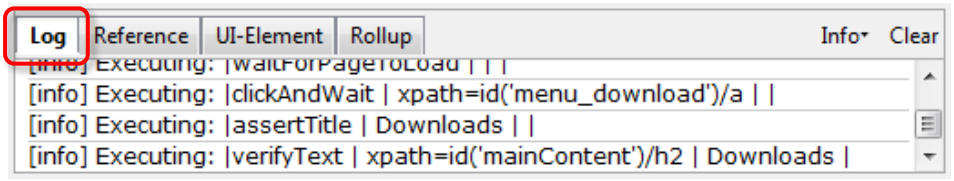


Log/Reference/UI-Element/Rollup Pane

The bottom pane is used for four different functions: Log, Reference, UI-Element, and Rollup depending on which tab is selected.

Log

When you run the test case, error messages and information messages are displayed in this pane automatically, even if you do not first select the Log tab. These messages are often useful for test case debugging. Notice the Clear button for clearing the Log. In addition, notice the Info button is a drop-down allowing selection of different levels of information to log.



Reference

The Reference tab is the default selection when entering or modifying Selenese commands and parameters in Table mode. In Table mode, the Reference pane will display documentation of the current command. When entering or modifying commands, whether from Table or Source mode, it is critically important to ensure that the parameters specified in the Target and Value fields match those specified in the parameter list in the Reference pane. The number of parameters provided must match the number specified, the order of parameters provided must match the order specified, and the type of parameters provided must match the type specified. If there is a mismatch in any of these three areas, the command will not run correctly.

UI-Element and Rollup

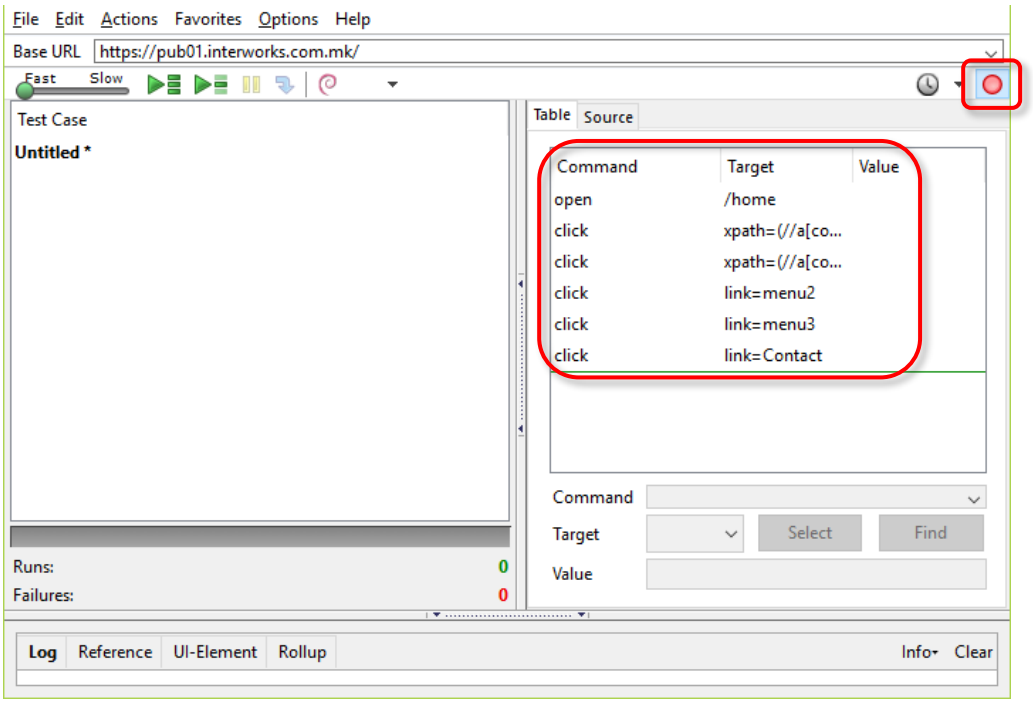
Detailed information on these two panes, which cover advanced features, can be found in the UI-Element Documentation on the Help menu of Selenium IDE.

7 BUILDING TEST CASES

There are several primary methods for developing test cases. Frequently, a test developer will require all three techniques.

Recording

Many first-time users begin by recording a test case from their interactions with a website. When Selenium IDE is first opened, the record button is ON by default. If you do not want Selenium IDE to begin recording automatically, then turn this OFF by going under Options > Options... and deselecting "Start recording immediately on open."



During recording, Selenium IDE will automatically insert commands into test case based on selected actions. Typically, this will include:

- clicking a link - *click* or *clickAndWait* commands
- entering values - *type* command
- selecting options from a drop-down listbox - *select* command
- clicking check boxes or radio buttons - *click* command

Here are some "gotchas" to be aware of:

- The type command may require clicking on some other area of the web page for it to record.
- Following a link usually records a click command. You will often need to change this to clickAndWait to ensure your test case pauses until the new page is completely loaded. Otherwise, test case will continue running commands before the page has loaded all its UI elements. This will cause unexpected test case failures.

Adding Verifications and Asserts with the Context Menu

The test cases will also need to check the properties of a web page. This requires *assert* and *verify* commands.

With Selenium IDE recording, go to the browser displaying the test application and right click anywhere on the page. You will see a context menu showing *verify* and/or *assert* commands

The first time you use Selenium, there may only be one Selenium command listed. As you use the IDE however, you will find additional commands that will be quickly added to this menu. Selenium IDE will attempt to predict what command, along with the parameters, you will need for a selected UI element on the current web page.

Let's see how this works. Open a web page of your choosing and select a block of text on the page. A paragraph or a heading will work fine. Now, right-click the selected text. The context menu should give you a *verifyTextPresent* command and the suggested parameter should be the text itself.

Also, notice the Show All Available Commands menu option. This shows more commands, along with suggested parameters, for testing your currently selected UI element.

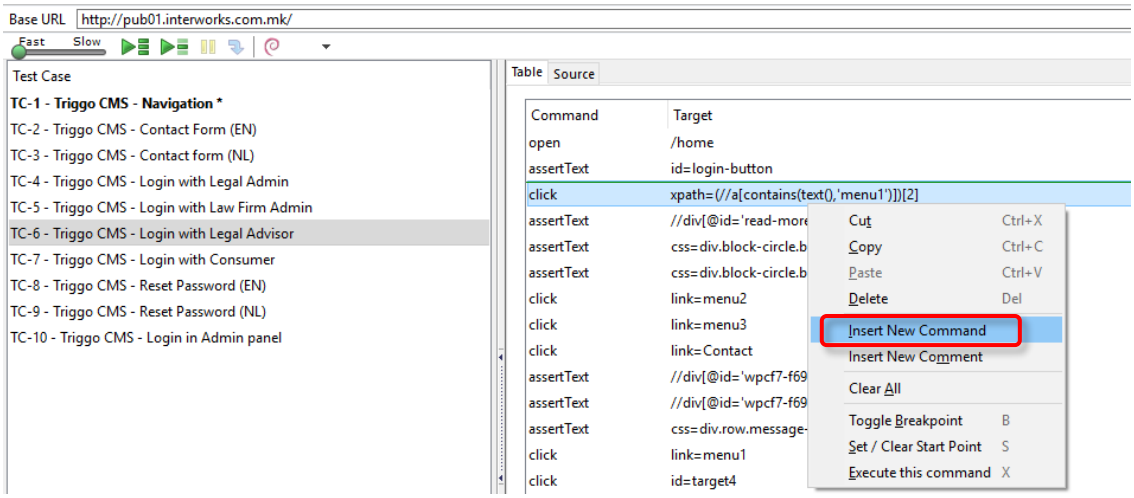
Try a few more UI elements. Try right clicking an image, or a user control like a button or a check box. You may need to use Show All Available Commands to see options other than *verifyTextPresent*. Once you select the other options, the more commonly used ones will show up on the primary context menu.

For example, selecting *verifyElementPresent* for an image should later cause that command to be available on the primary context menu next time you select an image and right-click.

Inserting/Editing Commands and Comments

Insert Command

Select the point in the test case where the command needs to be inserted. To do this, in the Test Case Pane, left-click on the line where the command needs to be inserted. Right-click and select Insert Command the IDE will add a blank line just ahead of the line you selected. Now use the command editing text fields to enter the new command and its parameters.

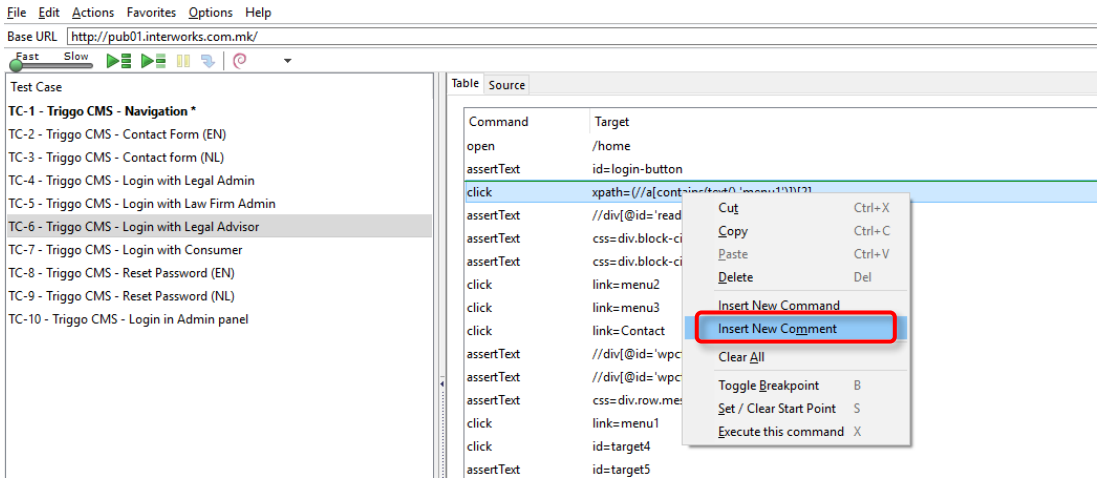


Insert Comment

Comments may be added to make the test case more readable. The comments are ignored when the test case is run.

Comments may also be used to add vertical white space in the tests. An empty command will cause an error during execution, an empty comment will not.

Select the line in the test case where you want to insert the comment. Right-click and select Insert Comment. Now use the Command field to enter the comment. The comment will appear in purple text.

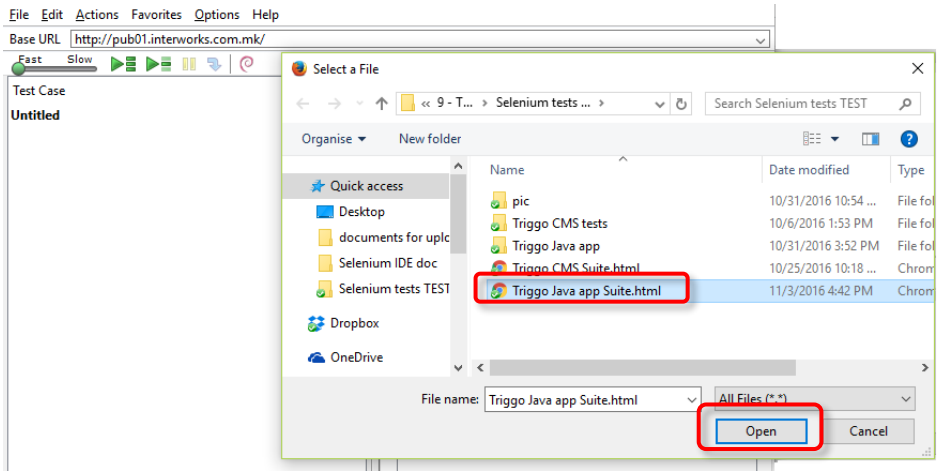


Edit a Command or Comment

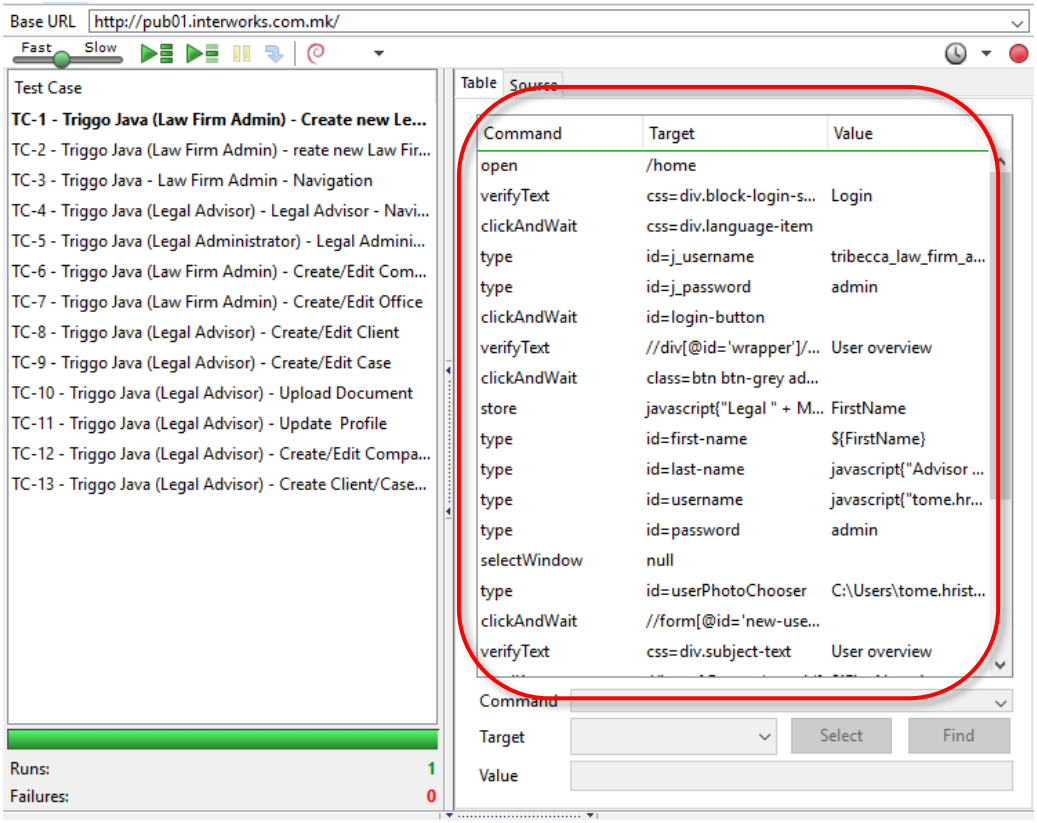
Simply select the line to be changed using the Command, Target, and Value fields.

Opening and Saving a Test Case

Like most programs, there are Save and Open commands under the File menu. However, Selenium distinguishes between test cases and test suites. To save the Selenium IDE tests for later use, save the individual test cases, or save the test suite. If the test cases have not been saved, tool will prompt to save them before saving the test suite.



When open an existing test case or suite, Selenium IDE displays its Selenium commands in the Test Case Pane.

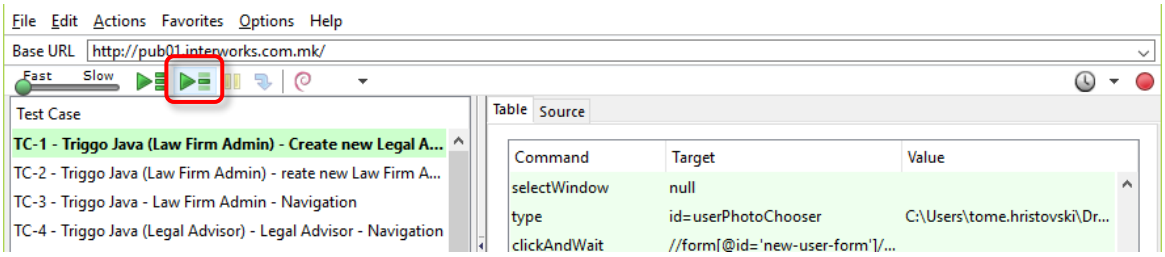


8 RUNNING TEST CASES

The IDE allows many options for running the test case. Test cases can be run all at once, stop and start it, run it one line at a time, run a single command, and run batch of an entire test suite. Execution of test cases is very flexible in the IDE.

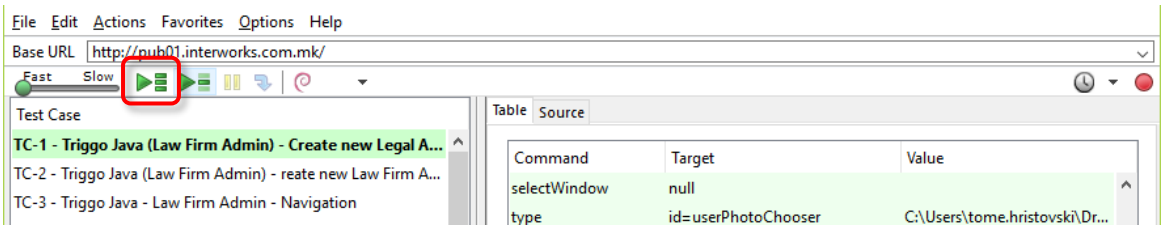
Run a Test Case

To run the currently displayed test case, click on the *Run* button.



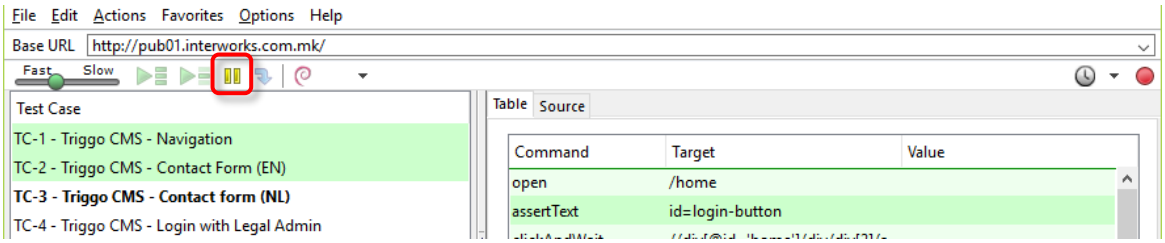
Run a Test Suite

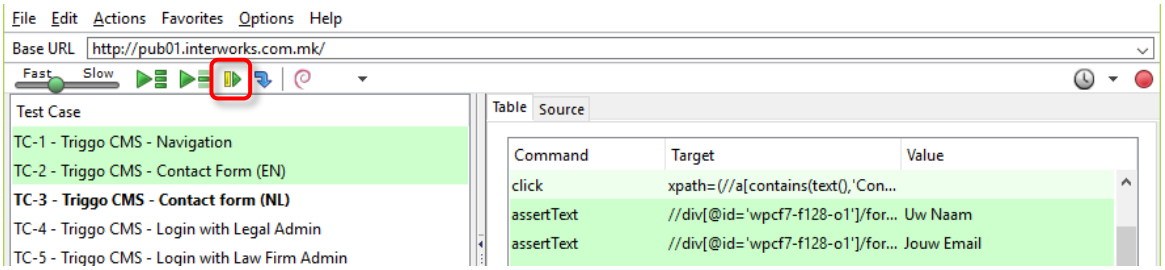
To run all the test cases in the currently loaded test suite, click on the *Run All button*.



Stop and Start

The Pause button can be used to stop the test case, which is running. To continue click Resume.





Stop in the Middle

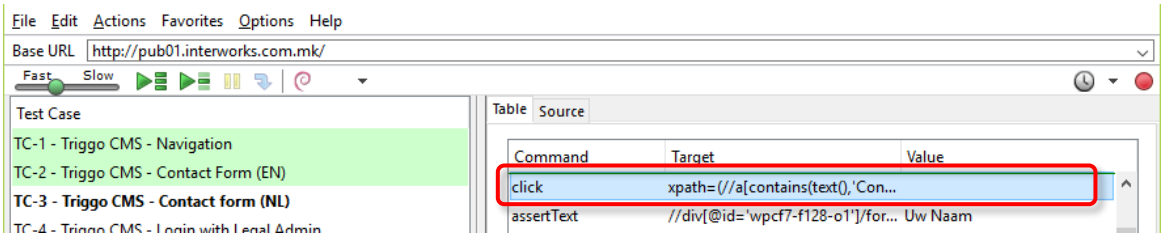
Breakpoint in the test case can be set to cause it to stop on a particular command. This is useful for debugging test case. To set a breakpoint, select a command, right-click, and from the context menu select Toggle Breakpoint.

Start from the Middle

Selenium IDE can begin running from a specific command in the middle of the test case. This also is used for debugging. To set a start point, select a command, right-click, and from the context menu select Set/Clear Start Point.

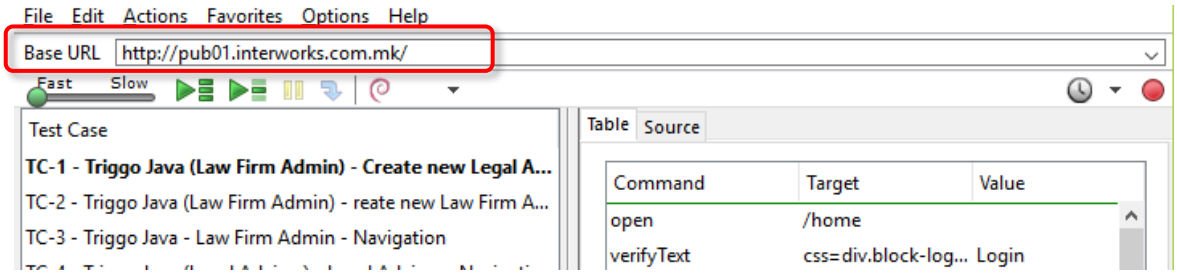
Run Any Single Command

Double-click any single command to run it by itself. This is useful when writing a single command. It lets you to immediately test a command that is constructing. Use double-click to see if it runs correctly. This is also available from the context menu.

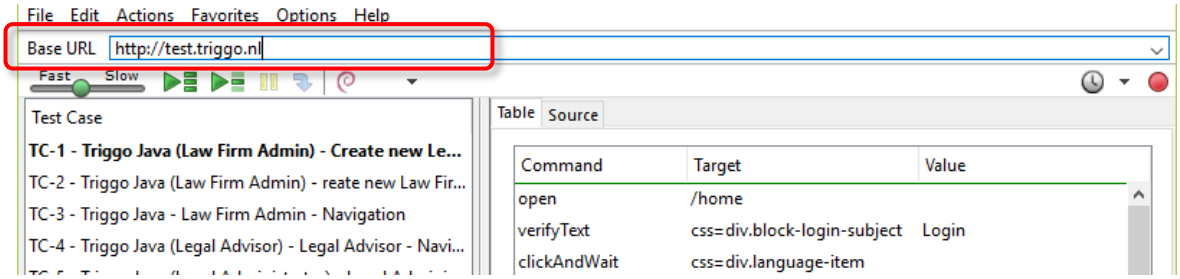


Using Base URL to Run Test Cases in Different Domains

The Base URL field at the top of the Selenium IDE window is very useful for allowing test cases to be run across different domains. Suppose that a site named http://pub01.interworks.com.mk/ had an in-house beta site named http://test.triggo.nl. Any test case for these sites that begins with an *open* statement should specify a *relative URL* as the argument to *open,* rather than an *absolute URL*. Selenium IDE will then create an absolute URL by appending the *open* command's argument onto the end of the value of Base URL. For example, the test case below would be run against http://pub01.interworks.com.mk/



This same test case with a modified Base URL setting would be run against http://test.triggo.nl.



9 TEST SUITES

A test suite is a collection of tests. Often one will run all the tests in a test suite as one continuous batch-job.

When using Selenium IDE, test suites can also be defined using a simple HTML file. The syntax again is simple. An HTML table defines a list of tests where each row defines the filesystem path to each test.

```
<html>
<head>
<title>Test Suite Function Tests - Priority 1</title>
</head>
<body>
<table>
  <tr><td><b>Suite Of Tests</b></td></tr>
  <tr><td><a href="./Login.html">Login</a></td></tr>
  <tr><td><a href="./SearchValues.html">Test Searching for Values</a></td></tr>
  <tr><td><a href="./SaveValues.html">Test Save</a></td></tr>
</table>
</body>
</html>
```

A file similar to this would allow running the tests all at once, one after another, from the Selenium IDE.

Test suites can also be maintained when using Selenium RC. This is done via programming and can be done in a number of ways. Commonly Junit is used to maintain a test suite if one is using Selenium RC with Java. If using an interpreted language like Python with Selenium-RC then some simple programming would be involved in setting up a test suite.

Writing a Test Suite

A test suite is a collection of test cases, which is displayed in the leftmost pane in the IDE. The test suite pane can be manually opened or closed via selecting a small dot halfway down the right edge of the pane.

The test suite pane will be automatically displayed when an existing test suite is opened *or* when the user selects the New Test Case item from the File menu. In the latter case, the new test case will appear immediately below the previous test case.

Selenium IDE also supports loading pre-existing test cases by using the File -> Add Test Case menu option. This allows you to add existing test cases to a new test suite.

A test suite file is an HTML file containing a one-column table. Each cell of each row in the <tbody> section contains a link to a test case. The example below is of a test suite containing four test cases:

```
<html>
```

```
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>Sample Selenium Test Suite</title>
</head>
<body>
   <table cellpadding="1" cellspacing="1" border="1">
      <thead>
         <tr><td>Test Cases A-Z Directory Links</td></tr>
      </thead>
   <tbody>
      <tr><td><a href="./a.html">A Links</a></td></tr>
      <tr><td><a href="./b.html">B Links</a></td></tr>
      <tr><td><a href="./c.html">C Links</a></td></tr>
      <tr><td><a href="./d.html">D Links</a></td></tr>
   </tbody>
   </table>
</body>
</html>
```

Note
Test case files shouldn't be co-located with the test suite file that invokes them.

10 SELENIUM COMMANDS

Selenium commands, often called *selenese*, are the set of commands that run the tests. A sequence of these commands is a *test script*. Selenium provides a rich set of commands for fully testing a web-app in virtually any way. These commands essentially create a testing language.

In selenese, one can test the existence of UI elements based on their HTML tags, test for specific content, for broken links, input fields, selection list options, submitting forms, and table data among other things. In addition, Selenium commands support testing of window size, mouse position, alerts, Ajax functionality, pop up windows, event handling, and many other web-application features. The Command Reference lists all the available commands. A command tells Selenium what to do.

Selenium commands come in three categories:

- **Actions** are commands that generally manipulate the state of the application. They do things like "click this link" and "select that option". If an Action fails, or has an error, the execution of the current test is stopped.

  Many Actions can be called with the "AndWait" suffix, e.g. "clickAndWait". This suffix tells Selenium that the action will cause the browser to make a call to the server, and that Selenium should wait for a new page to load.

- **Accessors** examine the state of the application and store the results in variables, e.g. "storeTitle". They are also used to automatically generate Assertions.

- **Assertions** are like Accessors, but they indicate the state of the application conforms to what is expected. Examples include "make sure the page title is X" and "verify that this checkbox is checked".

  All Selenium Assertions can be used in three modes:

  - assert
  - verify
  - waitFor

  For example: "assertText", "verifyText" and "waitForText". When an "assert" fails, the test is aborted. When a "verify" fails, the test will continue execution, logging the failure. This allows a single "assert" to ensure that the application is on the correct page, followed by a bunch of "verify" assertions to test form field values, labels, etc.

  "waitFor" commands wait for some condition to become true (which can be useful for testing Ajax applications). They will succeed immediately if the condition is already true. However, they will fail and halt the test if the condition does not become true within the current timeout setting (see the setTimeout action below).

Script Syntax

Selenium commands are simple, they consist of a command and two parameters. For example:

verifyText	//div//a[2]	Login

The parameters are not always required it depends on the command. Here are a couple more examples:

goBackAndWait		
verifyTextPresent		Welcome to My Home Page
type	id=phone	(555) 666-7066
type	id=address1	${myVariableAddress}

The command reference describes the parameter requirements for each command. Parameters vary, however they are typically:

- a *locator* for identifying a UI element within a page.
- a *text pattern* for verifying or asserting expected page content
- a *text pattern* or a selenium variable for entering text in an input field or for selecting an option from an option list.

Selenium scripts that will be run from Selenium IDE will be stored in an HTML text file format. This consists of an HTML table with three columns. The first column identifies the Selenium command, the second is a target, and the final column contains a value. The second and third columns may not require values depending on the chosen Selenium command, but they should be present. Each table row represents a new Selenium command. Here is an example of a test that opens a page, asserts the page title and then verifies some content on the page:

```
<table>
   <tr><td>open</td><td>/download/</td><td></td></tr>
   <tr><td>assertTitle</td><td></td><td>Downloads</td></tr>
   <tr><td>verifyText</td><td>//h2</td><td>Downloads</td></tr>
</table>
```

Rendered as a table in a browser this would look like the following:

open	/download/	
assertTitle		Downloads
verifyText	//h2	Downloads

The Selenese HTML syntax can be used to write and run tests without requiring knowledge of a programming language. With a basic knowledge of selenese and Selenium IDE you can quickly produce and run test cases.

Commonly Used Selenium Commands
These are probably the most commonly used commands for building tests:

Command	Explanation
Open	Opens a page using a URL
click/clickAndWait	Performs a click operation, and optionally waits for a new page to load
verifyTitle/assertTitle	Verifies an expected page title
verifyTextPresent	Verifies expected text is somewhere on the page
verifyElementPresent	Verifies an expected UI element, as defined by its HTML tag, is present on the page
verifyText	Verifies expected text and its corresponding HTML tag are present on the page
verifyTable	Verifies a table's expected contents
waitForPageToLoad	Pauses execution until an expected new page loads. Called automatically when clickAndWait is used
waitForElementPresent	Pauses execution until an expected UI element, as defined by its HTML tag, is present on the page
Store	The plain *store* command is the most basic of the many store commands and can be used to simply store a constant value in a selenium variable
echo	Echo statements can be used to print the contents of Selenium variables
storeLocation	Store current selected window's URL in selenium IDE software testing tool

Verifying Page Elements

Verifying UI elements on a web page is probably the most common feature of the automated tests. Selenese allows multiple ways of checking for UI elements. It is important to understand these different methods because they define what is actually tested.

For example:

1. an element is present somewhere on the page?
2. specific text is somewhere on the page?
3. specific text is at a specific location on the page?

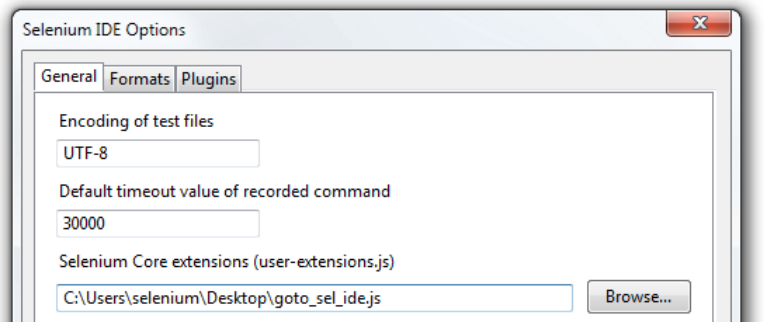
For example, if you are testing a text heading, the text and its position at the top of the page are probably relevant for your test. If, however, you are testing for the existence of an image on the home page, and the web designers frequently change the specific image file along with its position on the page, then you only want to test that *an image* (as opposed to the specific image file) exists *somewhere on the page*.

User Extensions

User extensions are JavaScript files that allow one to create his or her own customizations and features to add additional functionality. Often this is in the form of customized commands although this extensibility is not limited to additional commands.There are a number of useful extensions created by users.

Perhaps the most popular of all Selenium IDE extensions is one that provides flow control in the form of while loops and primitive conditionals. This extension is the goto_sel_ide.js. For an example on how to use the functionality provided by this extension, look at the page created by its author.

To install this extension, put the pathname to its location on the computer in the Selenium Core extensions field of Selenium-IDE's Options=>Options=>General tab.



After selecting the OK button, close and reopen Selenium IDE in order for the extensions file to be read. Any change that is made to an extension will also require closing and reopening Selenium IDE.