# Unit 6: Selenium

## Prerequisites

Before learning the concepts of Selenium, you should have a basic understanding of java or any other object-oriented programming language.

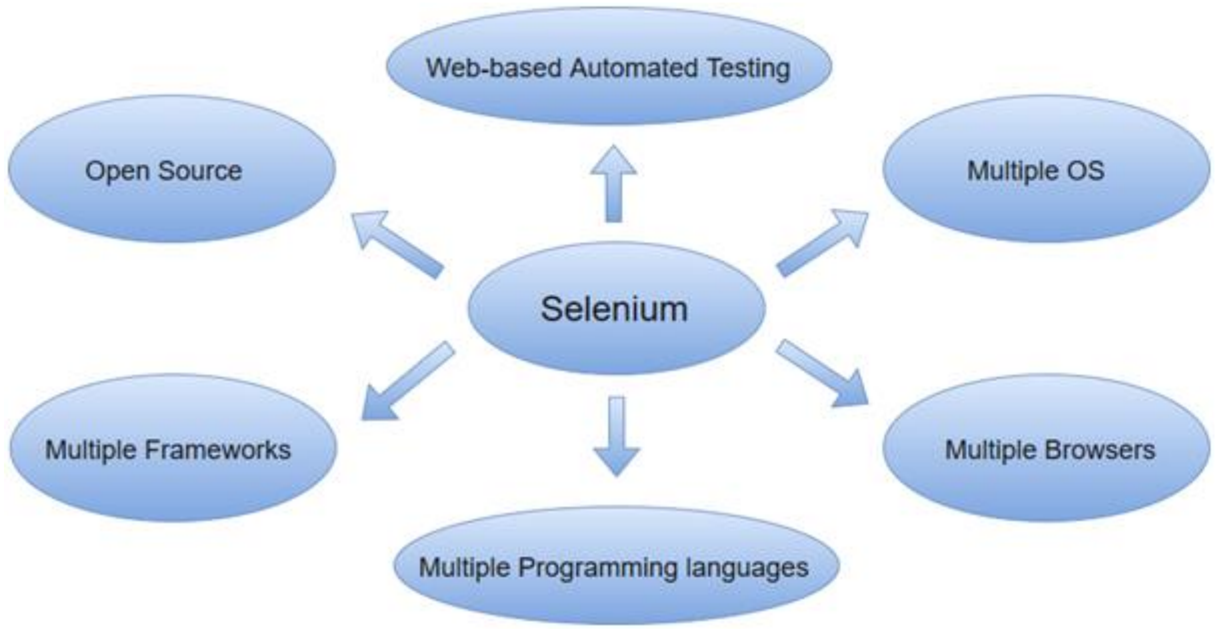
Languages supported by Selenium include C#, Java, Perl, PHP, Python and Ruby.Currently, Selenium Webdriver is most popular with Java and C#. So, if you know any of the languages then it won't be tough to understand the concepts of Selenium. In addition, you should have prior knowledge of software testing techniques like automation testing, functional testing, etc.

## What is Selenium

**Selenium is one of the most widely used open source Web UI (User Interface) automation testing suite.** It was originally developed by **Jason Huggins in 2004** as an internal tool at Thought Works. Selenium supports automation across different browsers, platforms and programming languages.

Selenium can be easily deployed on platforms such as Windows, Linux, Solaris and Macintosh. Moreover, it supports OS (Operating System) for mobile applications like iOS, windows mobile and android.

Selenium supports a variety of programming languages through the use of drivers specific to each language.Languages supported by Selenium include C#, Java, Perl, PHP, Python and Ruby.Currently, Selenium Web driver is most popular with Java and C#. Selenium test scripts can be coded in any of the supported programming languages and can be run directly in most modern web browsers. Browsers supported by Selenium include Internet Explorer, Mozilla Firefox, Google Chrome and Safari.



Selenium can be used to automate functional tests and can be integrated with automation test tools such as **Maven**, **Jenkins**, **& Docker** to achieve continuous testing. It can also be integrated with tools such as **TestNG**, & **JUnit** for managing test cases and generating reports.

## Automation Testing

Automation testing uses the specialized tools to automate the execution of manually designed test cases without any human intervention. Automation testing tools can access the test data, controls the execution of tests and compares the actual result against the expected result. Consequently, generating detailed test reports of the system under test.

Automation testing covers both functional and performance test on an application.

- Functional automation is used for automation of functional test cases. For example, regression tests, which are repetitive in nature, are automated.
- Performance automation is used for automation of non-functional performance test cases. For example, measuring the response time of the application under considerable (say 100 users) load.

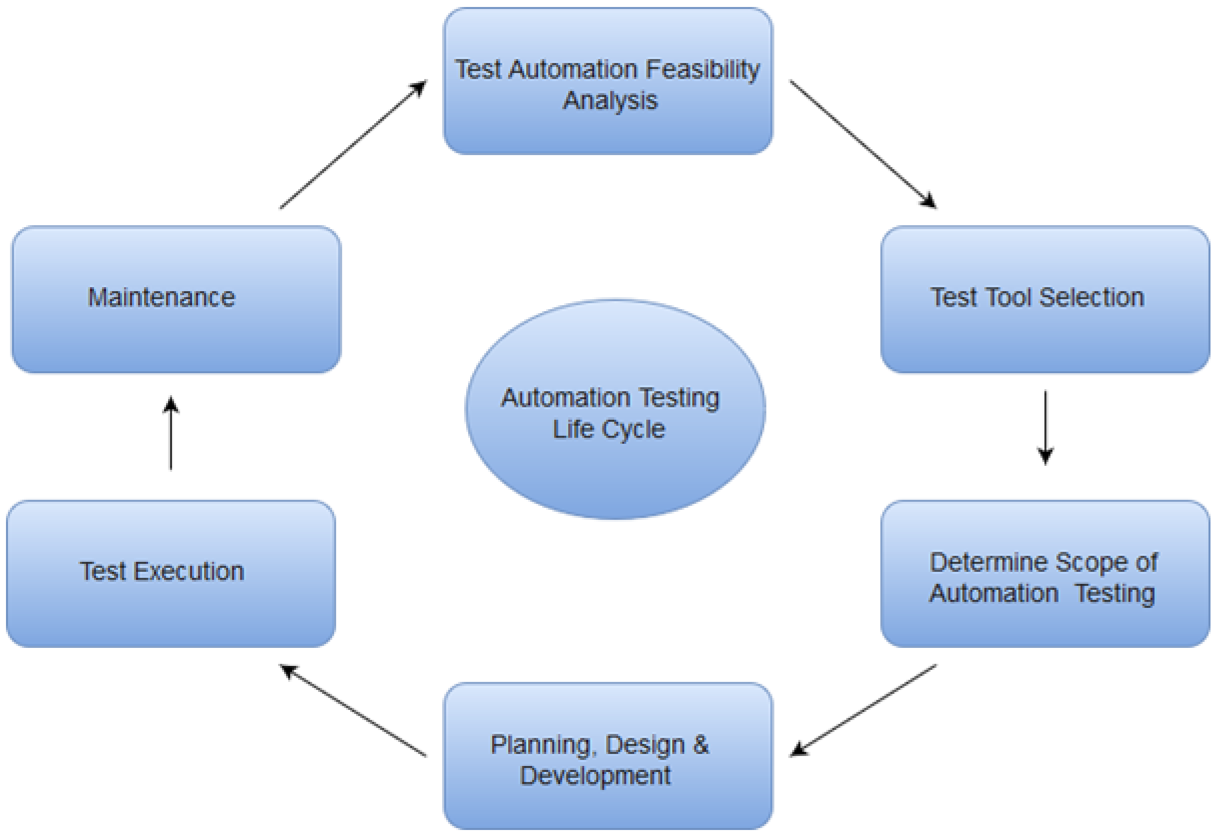
Automation Testing tools which are used for functional automation:

- Quick Test Professional, provided by HP.
- Rational Robot, provided by IBM.
- Coded UI, provided by Microsoft.
- Selenium, open source.
- Auto It, open Source.

Automation Testing tools which are used for non-functional automation:

- Load Runner, provided by HP.
- JMeter, provided by Apache.
- Burp Suite, provided by PortSwigger.
- Acunetix, provided by Acunetix.

## Automation Testing Life Cycle


Test Automation Feasibility Analysis
Test Tool Selection
Determine Scope of Automation Testing
Planning, Design & Development
Test Execution
Maintenance
Automation Testing Life Cycle

## Why Automated Testing

Automation testing has specific advantages for improving long-term efficiency of any software. The key benefits of test automation are:

- Automated testing has long been considered beneficial for big software organizations. Although, it is often thought to be too expensive or difficult for smaller companies to implement.
- Automated testing tools can be programmed to build and execute test scripts at a specific time without involving any human intervention.For instance, automated test can be automatically kicked off overnight, and the testers can analyse the results of the automated the next morning.

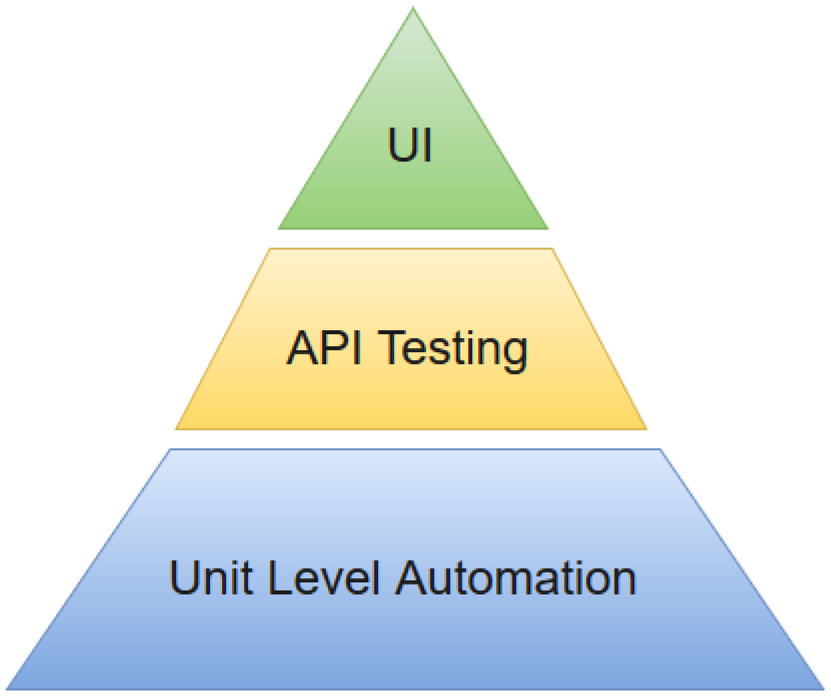
- Automated testing tools are able to playback pre-recorded and pre-defined actions.
- Automation testing supports frequent regression testing.
- It provides rapid feedback to developers.
- It provides unlimited iterations of test case execution.
- It provides disciplined documentation of test cases.
- Automated test generates customized defect reports.
- Less error prone as compared to manual testing.

## Test Automation for Web Applications

If we take a look at the type of software applications prevailing in current market scenario, most of the software applications are written as web-based applications to be run in an internet browser. The testing strategy for web-based 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 software projects.

The most effective manner to carry out test automation for web application is to adopt a pyramid testing strategy.This pyramid testing strategy includes automation tests at three different levels. Unit testing represents the base and biggest percentage of this test automation pyramid. Next comes, service layer, or API testing. And finally, GUI tests sit at the top. The pyramid looks something like this:

Test Automation Pyramid :



## Selenium Features

- Selenium is an open source and portable Web testing Framework.
- Selenium IDE provides a playback and record feature for authoring tests without the need to learn a test scripting language.
- It can be considered as the leading cloud-based testing platform which helps testers to record their actions and export them as a reusable script with a simple-to-understand and easy-to-use interface.
- Selenium supports various operating systems, browsers and programming languages. Following is the list:
    - Programming Languages: C#, Java, Python, PHP, Ruby, Perl, and JavaScript
    - Operating Systems: Android, iOS, Windows, Linux, Mac, Solaris.
    - Browsers: Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc.
- It also supports parallel test execution which reduces time and increases the efficiency of tests.
- Selenium can be integrated with frameworks like Ant and Maven for source code compilation.
- Selenium can also be integrated with testing frameworks like TestNG for application testing and generating reports.
- Selenium requires fewer resources as compared to other automation test tools.
- WebDriver API has been indulged in selenium whichis one of the most important modifications done to selenium.
- Selenium web driver does not require server installation, test scripts interact directly with the browser.
- Selenium commands are categorized in terms of different classes which make it easier to understand and implement.
- Selenium Remote Control (RC) in conjunction with WebDriver API is known as Selenium 2.0. This version was built to support the vibrant web pages and Ajax.

## Selenium Limitations

- Selenium does not support automation testing for desktop applications.
- Selenium requires high skill sets in order to automate tests more effectively.
- Since Selenium is open source software, you have to rely on community forums to get your technical issues resolved.
- We can't perform automation tests on web services like SOAP or REST using Selenium.
- We should know at least one of the supported programming languages to create tests scripts in Selenium WebDriver.
- It does not have built-in Object Repository like UTF/QTP to maintain objects/elements in centralized location. However, we can overcome this limitation using Page Object Model.
- Selenium does not have any inbuilt reportingcapability; you have to rely on plug-ins like **JUnit** and **TestNG** for test reports.

- It is not possible to perform testing on images. We need to integrate Selenium with **Sikuli** for image based testing.
- Creating test environment in Selenium takes more time as compared to vendor tools like UFT, RFT, Silk test, etc.
- No one is responsible for new features usage; they may or may not work properly.
- Selenium does not provide any test tool integration for Test Management.

## QTP Introduction

In general, the term QTP stands for **Quick Test Professional**, and nowadays, it is also known as UTF or Unified Functional Testing. It is generally considered to be an automated functional testing tool that helps testers while performing regression testing. However, it is considered one of the most advanced tools as it does not require the tester's attention all the time, or we can say in simple words that it does not force them to keep an eye always. In addition, it can work automatically as soon as the tester completes the script, which we will discuss later in this tutorial. Typically, UTF uses Visual Basic scripts to automate certain types of applications, and sometimes this type of script is also abbreviated as VB script.

## Selenium vs QTP

Selenium and QTP are the most frequently used automation test tools in the market. Hence, we have compared some of the features of Selenium over QTP.

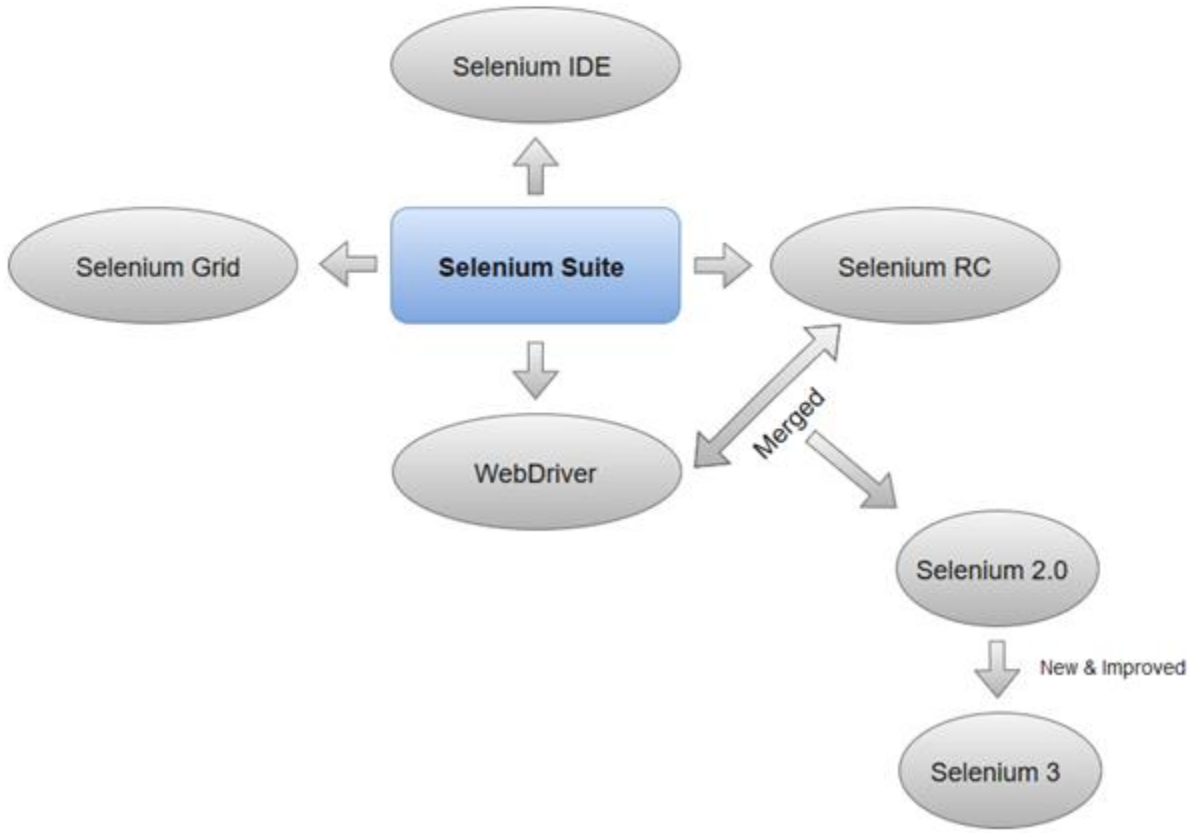
| **Features** | **Selenium** | **HP QTP** |
|---|---|---|
| License | Open source tool | Required |
| Customer support | Dedicated HP support | Selenium community forums |
| Test Support | Supportsautomation only for web-based applications. | Support tests on both web and desktop based applications. |
| Resource consumption during test scripts execution | Low resource consumption | High resource consumption |
| Supported programming languages | Java, C#, Ruby, Python, Perl, PHP and JavaScript | VB Script. |
| Supported Environments | Android, iOS, Windows, Linux, Mac, Solaris. | Only for Windows |

| | | |
|---|---|---|
| Supported Browsers | Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc. | Specific versions of Google Chrome, Mozilla Firefox and Internet Explorer. |
| Object Repository/Recovery Scenario | Absent | Built-in object repository and recovery scenario. |
| Browser Controls | None | Controls like favourites bar, backward and forward buttons can be accessed within the browser. |
| Test Report Generation | It relies on external tool for generating test reports. | Built-in test report generation within the tool. |
| Parameterization | You have to rely on any one of the supported programming language for parameterization. | Built-in tools are available for parameterization. |

## Selenium Tool Suite

Selenium is not just a single tool but a suite of software, each with a different approach to support automation testing. It comprises of four major components which include:

1. Selenium Integrated Development Environment (IDE)
2. Selenium Remote Control (Now Deprecated)
3. WebDriver
4. Selenium Grid



## 1.Selenium Integrated Development Environment (IDE)

Selenium IDE is implemented as Firefox extension which provides record and playback functionality on test scripts. It allows testers to export recorded scripts in many languages like HTML, Java, Ruby, RSpec, Python, C#, JUnit and TestNG. You can use these exported script in Selenium RC or Webdriver.

>Selenium IDE has limited scope and the generated test scripts are not very robust and portable.

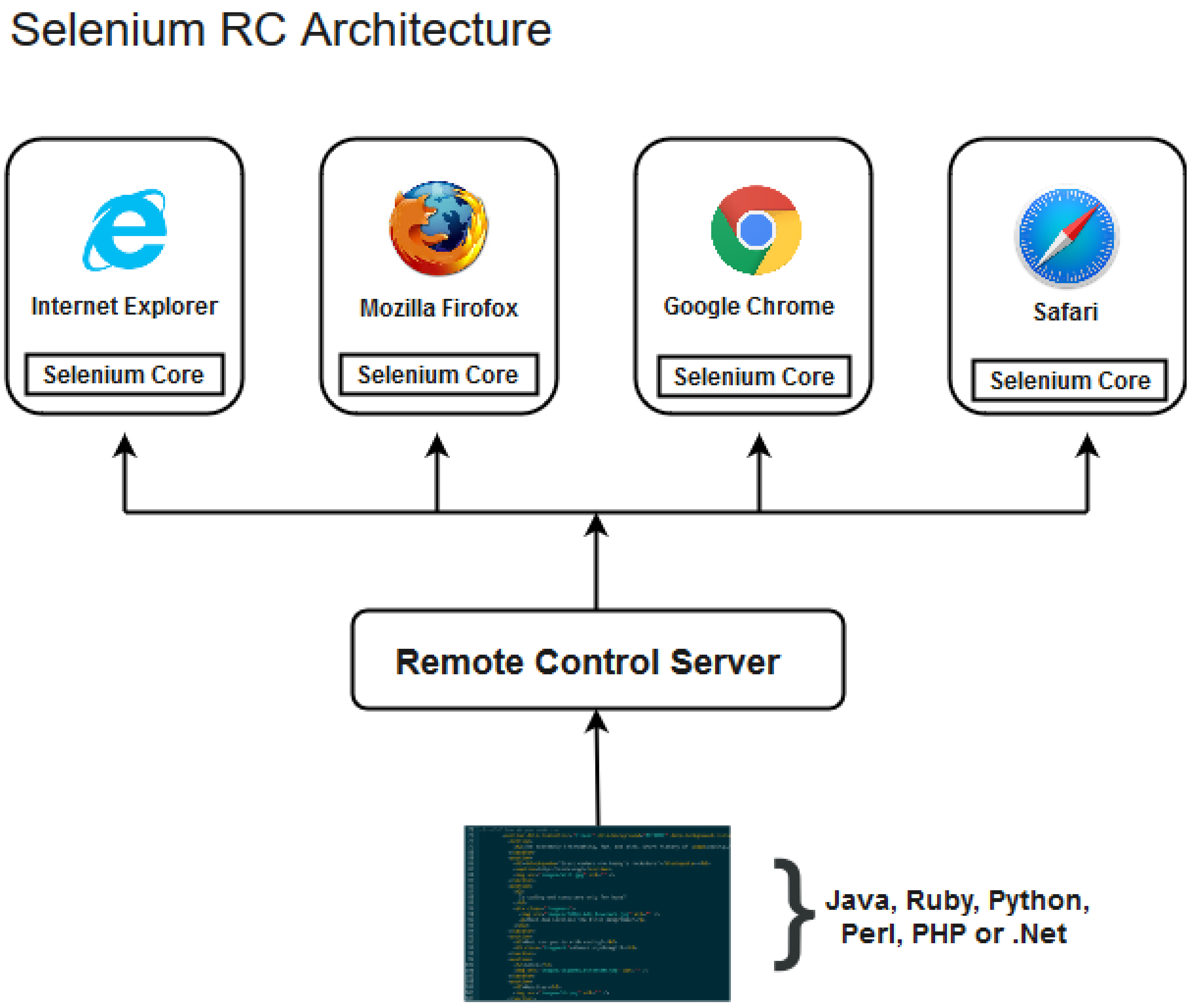
## 2. Selenium Remote Control

Selenium RC (officially deprecated by selenium)allows testers to write automated web application UI test in any of the supported programming languages. It also involves an HTTP proxy server which enables the browser to believe that the web application being tested comes from the domain provided by proxy server.

Selenium RC comes with two components.

1. Selenium RC Server (acts as a HTTP proxy for web requests).
2. Selenium RC Client (library containing your programming language code).

The figure given below shows the architectural representation of Selenium RC.



Selenium RC had been considered quite effective for testing complex AJAX-based web user interfaces under a Continuous Integration System.

## 3. Selenium WebDriver

Selenium WebDriver (Selenium 2) is the successor to Selenium RC and is by far the most important component of Selenium Suite. SeleniumWebDriverprovides a programming interface to create and execute test cases. Test scripts are written in order to identify web elements on web pages and then desired actions are performed on those elements.

Selenium WebDriver performs much faster as compared to Selenium RC because it makes direct calls to the web browsers. RC on the other hand needs an RC server to interact with the web browser.

Since, WebDriver directly calls the methods of different browsers hence we have separate driver for each browser. Some of the most widely used web drivers include:

- Mozilla Firefox Driver (Gecko Driver)
- Google Chrome Driver
- Internet Explorer Driver
- Opera Driver
- Safari Driver
- HTML Unit Driver (a special headless driver)

Note:Selenium version 2 merged the best features of Selenium RC and Selenium WebDriver into Selenium WebDriver. The latest release Selenium 3 has new added features and functionalities

### 4. Selenium Grid

Selenium Grid is also an important component of Selenium Suite which allows us to run our tests on different machines against different browsers in parallel. In simple words, we can run our tests simultaneously on different machines running different browsers and operating systems.

Selenium Grid follows the **Hub-Node Architecture** to achieve parallel execution of test scripts. The Hub is considered as master of the network and the other will be the nodes. Hub controls the execution of test scripts on various nodes of the network.

## Selenium Integrated Development Environment (IDE)

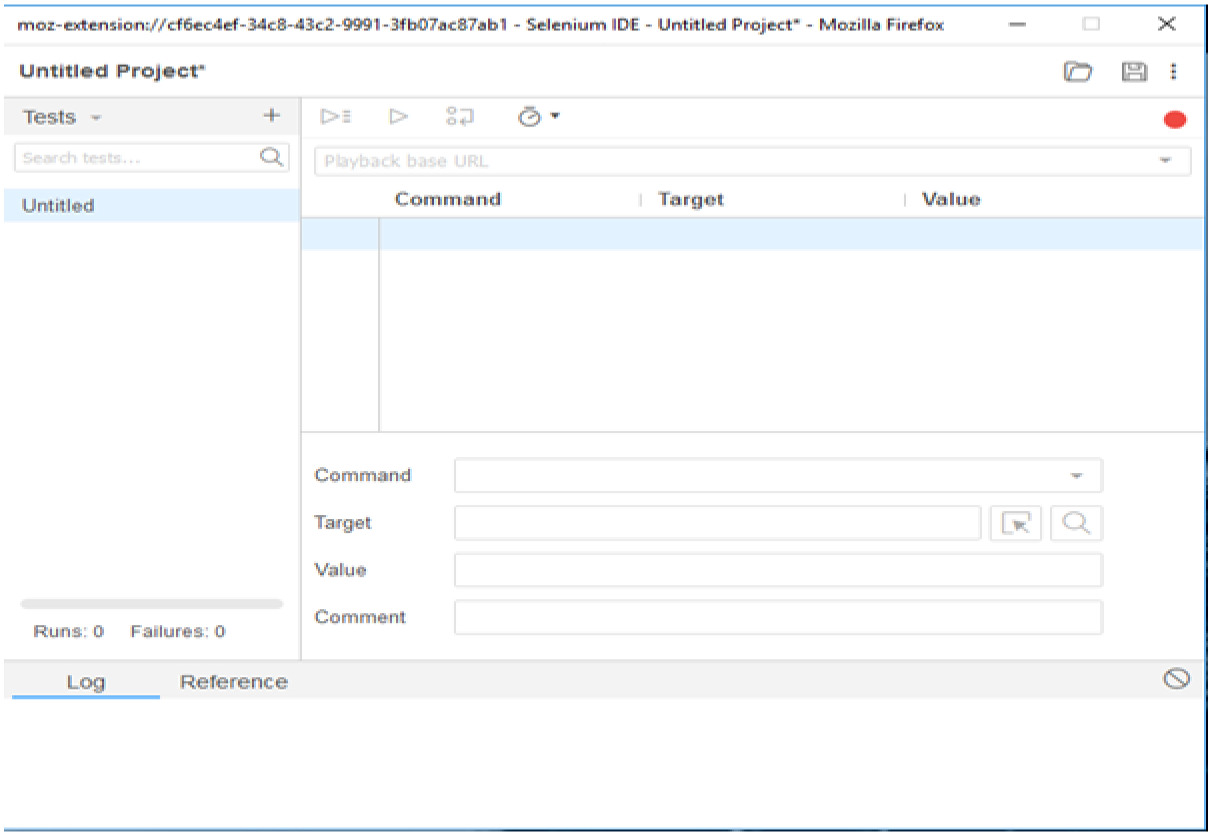
Selenium IDE (Integrated Development Environment) is an open source web automation testing tool under the Selenium Suite. Unlike Selenium WebDriver and RC, it does not require any programming logic to write its test scripts rather you can simply record your interactions with the browser to create test cases. Subsequently, you can use the playback option to re-run the test cases.

Perhaps, creating test cases on Selenium IDE does not require any programming language but when you get to use selenese commands like **runScript**, a little knowledge prior to JavaScript would prove beneficial for you to understand the concepts more clearly.

*Note: Selenium IDE is available only as Mozilla Firefox and Chrome plug-in, which means you can't record your test cases on browsers other than Firefox and Chrome. The recorded test scripts can also be exported to programming languages like C#, Java, Ruby or Python.*

The following image shows the default interface of Selenium IDE:

ADVERTISEMENT



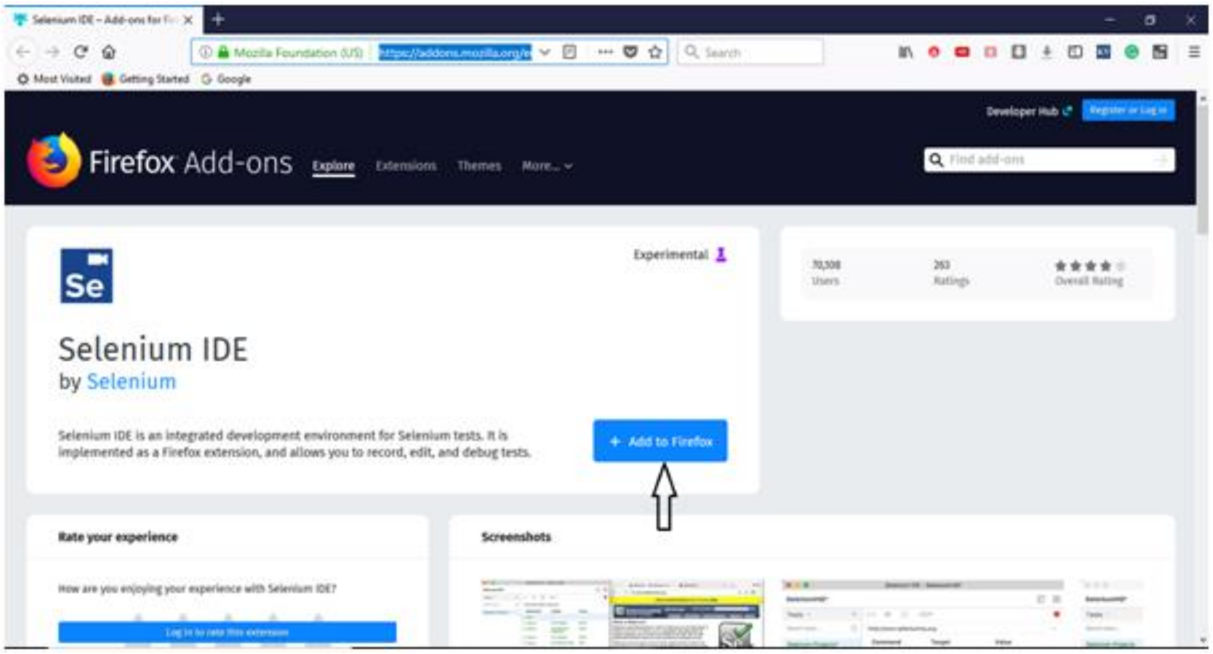
## Selenium IDE-Installation

Since, Selenium IDE is available only as Firefox and Chrome plug-in, we assume that you have already installed Mozilla Firefox browser in your system. However, you can download the latest version of Firefox through their official website provided under the link given below.

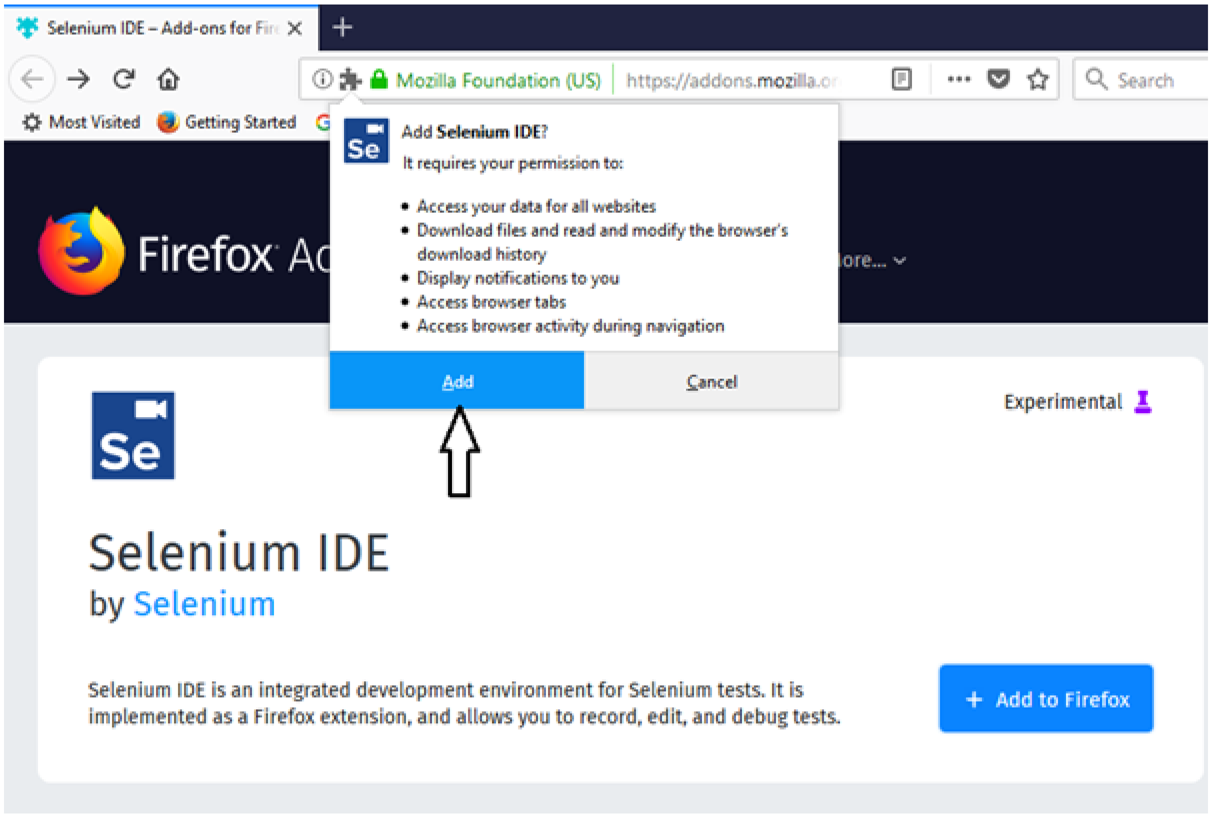
https://www.mozilla.org/en-US/firefox/new/

## Selenium IDE Download and Install

- Launch Mozilla Firefox browser.
- Open URLhttps://addons.mozilla.org/en-us/firefox/addon/selenium-ide/It will redirect you to the official add-on page of Firefox.
- Click on "Add to Firefox" button.


Firefox Add-ons
Explore
Extensions
Themes
More...
Experimental
Selenium IDE
by Selenium
Selenium IDE is an integrated development environment for Selenium tests. It is implemented as a Firefox extension, and allows you to record, edit, and debug tests.
+ Add to Firefox
Rate your experience
Screenshots

- A pop-up dialog box will be appeared asking you to add Selenium IDE as extension to your Firefox browser.
- Click on "Add" button.

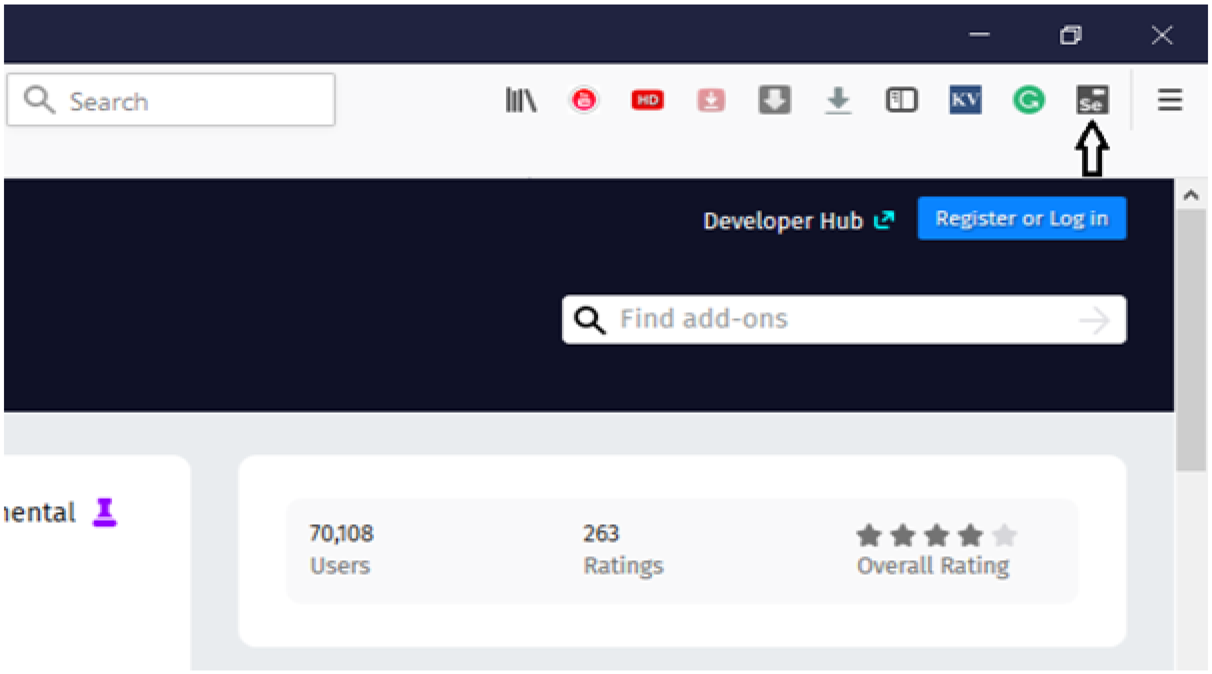

Add Selenium IDE?

It requires your permission to:

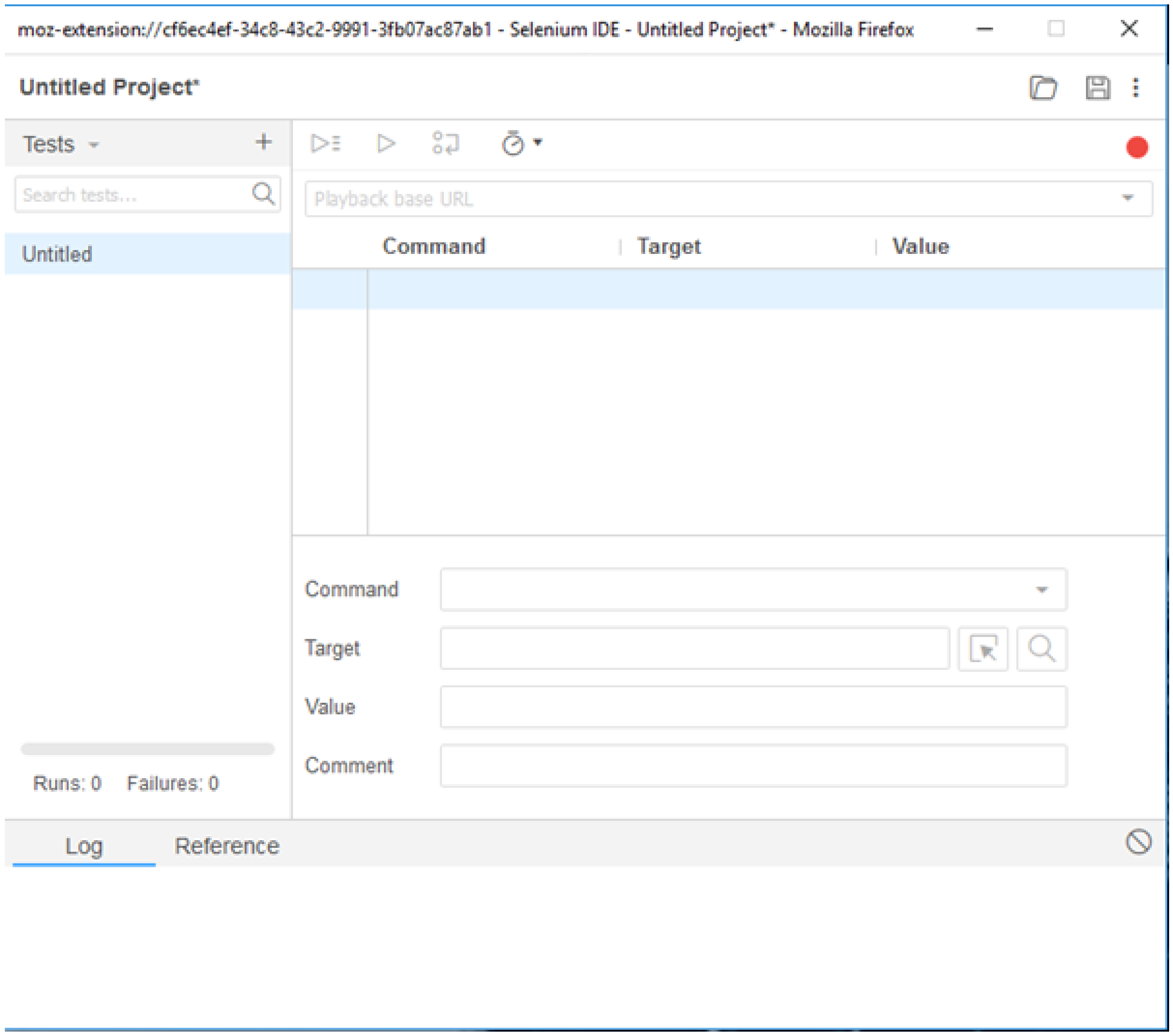
- Access your data for all websites
- Download files and read and modify the browser's download history
- Display notifications to you
- Access browser tabs
- Access browser activity during navigation

Selenium IDE is an integrated development environment for Selenium tests. It is implemented as a Firefox extension, and allows you to record, edit, and debug tests.

- Restart you Firefox browser.
- Go to the top right corner on your Firefox browser and look for the Selenium IDE icon.



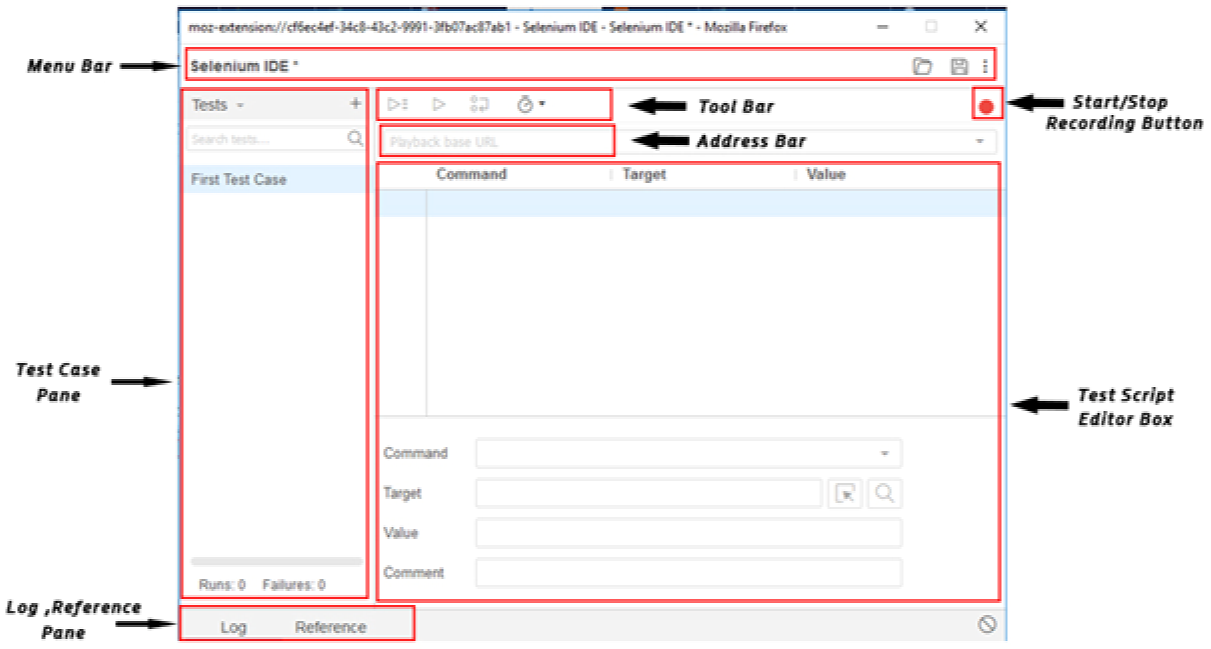
- Click on that icon to launch Selenium IDE.



## Selenium IDE-Features

Selenium IDE is divided into different components, each having their own features and functionalities.We have categorized seven different components of Selenium IDE, which includes:

5. Menu Bar
6. Tool Bar
7. Address Bar
8. Test Case Pane
9. Test Script Editor Box
10. Start/Stop Recording Button
11. Log, Reference Pane

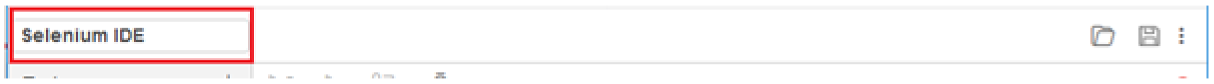


Now, we will look at the features and functionalities of each component in detail.

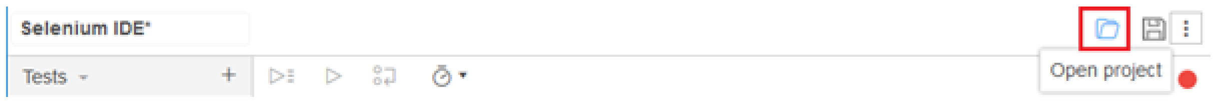
## 1. Menu Bar

Menu bar is positioned at the top most portion of the Selenium IDE interface. The most commonly used modules of menu bar include:

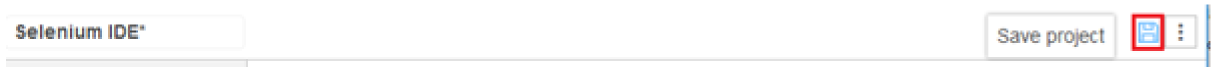
- Project Name
  It allows you to rename your entire project.



- Open Project
  It allows you to load any existing project from your personal drives.



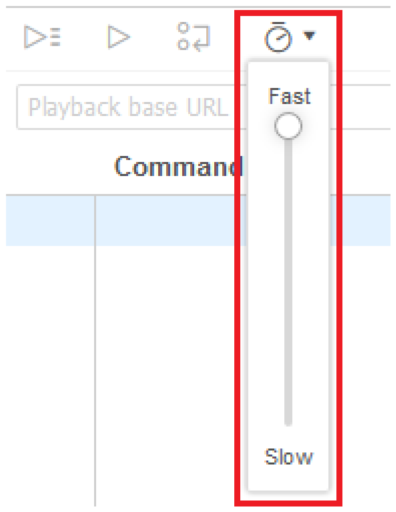
- Save Project
  It allows you to save the entire project you are currently working on.



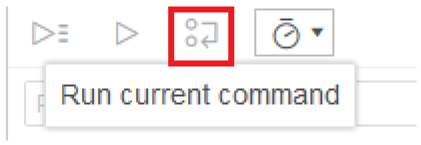
## 2. Tool Bar

The Tool bar contains modules for controlling the execution of your test cases. In addition, it gives you a step feature for debugging you test cases. The most commonly used modules of Tool Bar menu include:

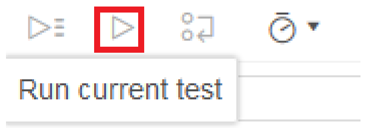
- Speed Control Option
  It allows you to control the execution speed of your test cases.



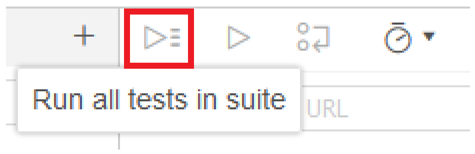
- Step Feature
  It allows you to "step" through a test case by running it one command at a time. Use for debugging test cases.



- Run Tests
  It allows you to run the currently selected test. When only a single test is loaded "Run Test" button and "Run all" button have the same effect.

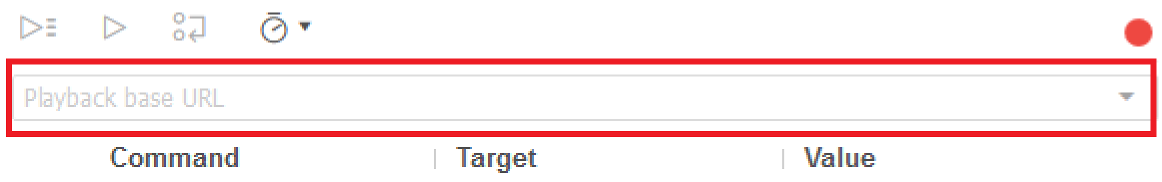


- Run All
  It allows you to run the entire test suite when a test suite with multiple test cases is loaded.



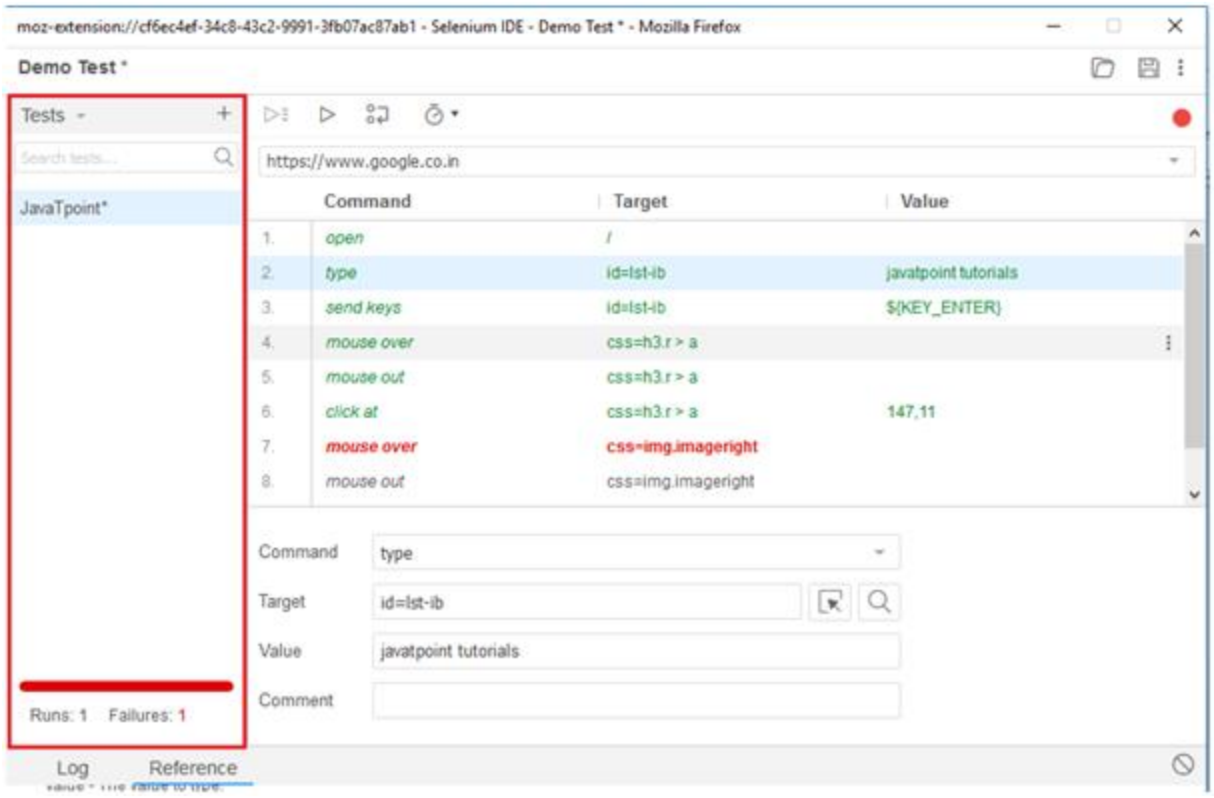
## 3. Address Bar

This module provides you a dropdown menu that remembers all previous values for base URL. In simple words, the base URL address bar remembers the previously visited websites so that the navigation becomes easy later on.



## 4. Test Case Pane

This module contains all the test cases that are recorded by IDE. In simple words, it provides the list of all recorded test cases at the same time under the test case pane so that user could easily shuffle between the test cases.


moz-extension://cf6ec4ef-34c8-43c2-9991-3fb07ac87ab1 - Selenium IDE - Demo Test * - Mozilla Firefox

Demo Test *

Tests

JavaTpoint*

https://www.google.co.in

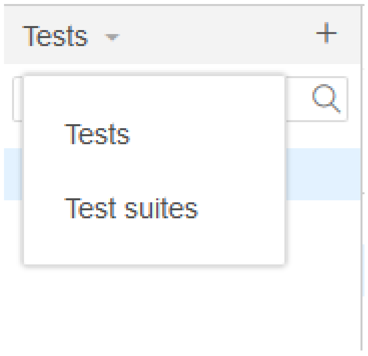
| | Command | Target | Value |
|---|---|---|---|
| 1. | open | / | |
| 2. | type | id=lst-ib | javatpoint tutorials |
| 3. | send keys | id=lst-ib | ${KEY_ENTER} |
| 4. | mouse over | css=h3.r > a | |
| 5. | mouse out | css=h3.r > a | |
| 6. | click at | css=h3.r > a | 147,11 |
| 7. | mouse over | css=img.imageright | |
| 8. | mouse out | css=img.imageright | |

| | |
|---|---|
| Command | type |
| Target | id=lst-ib |
| Value | javatpoint tutorials |
| Comment | |

Runs: 1 Failures: 1

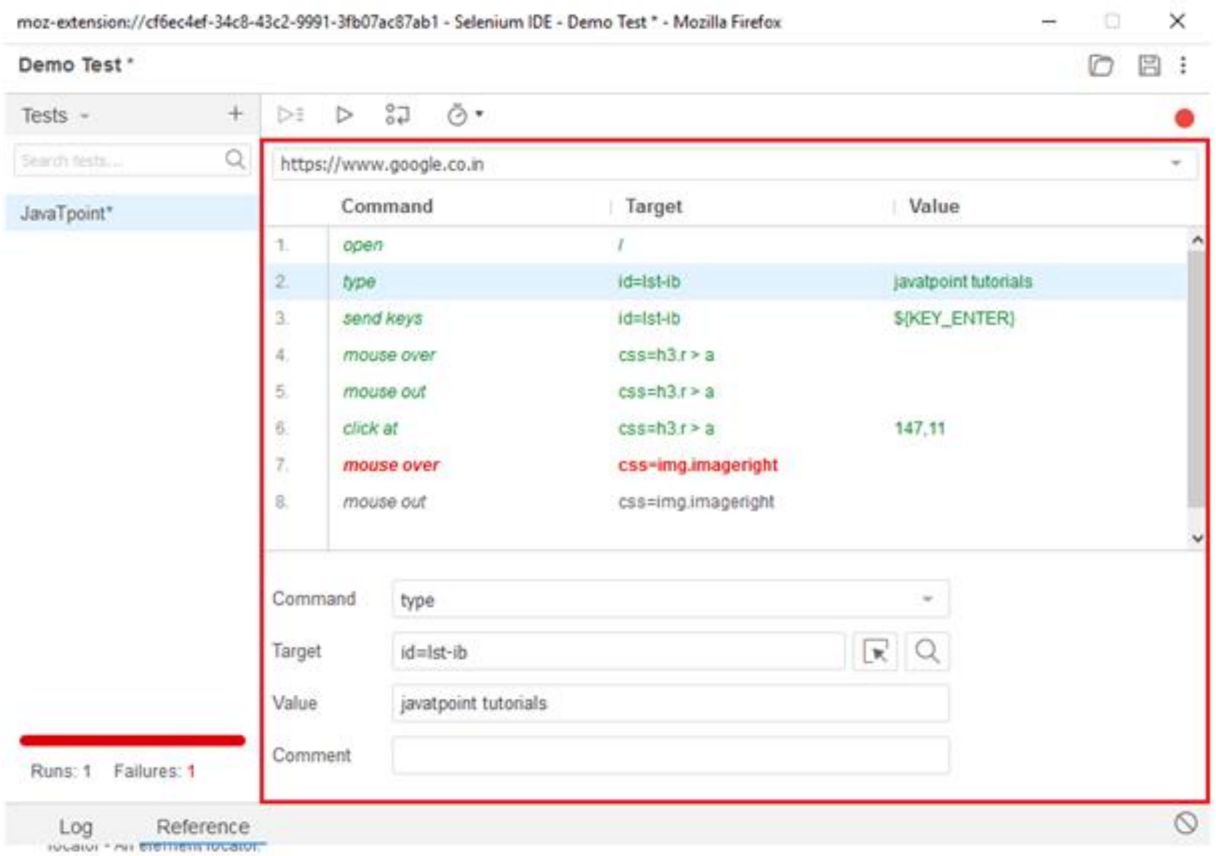
At the bottom portion of the Test Case Pane, you can see the test execution result summary which includes the pass/fail status of various test cases.

Test Case Pane also includes features like Navigation panel which allow users to navigate between test cases and test suites.



## 5. Test Script Editor Box

Test Script Editor Box displaysall of the test scripts and user interactions that were recorded by the IDE. Each user interaction is displayed in the same order in which they are performed. The Editor box is divided into three columns:Command, Target and Value.

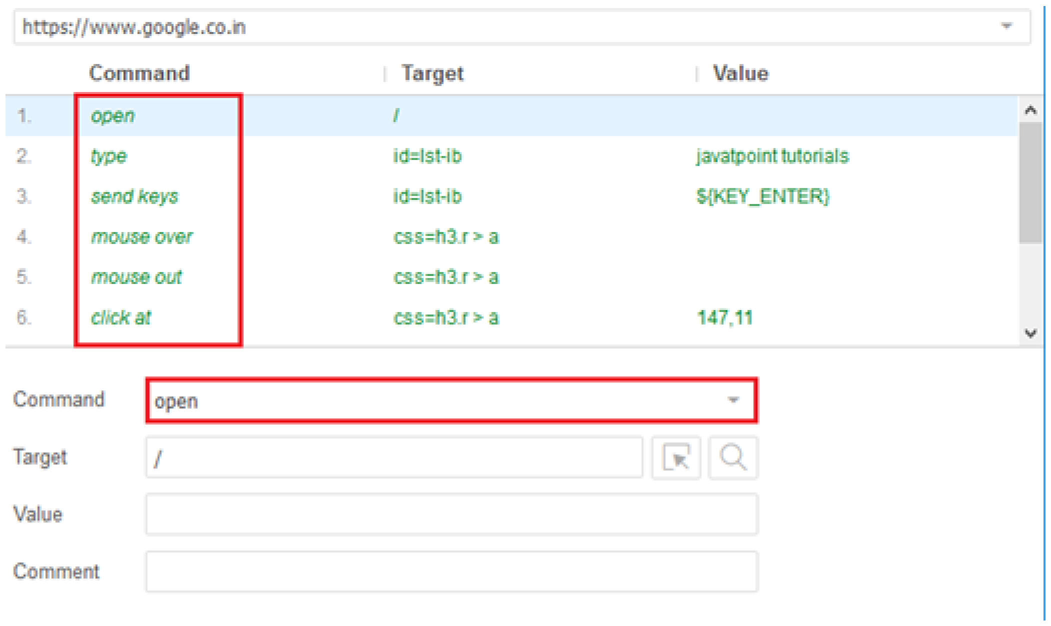

moz-extension://cf6ec4ef-34c8-43c2-9991-3fb07ac87ab1 - Selenium IDE - Demo Test * - Mozilla Firefox
Demo Test *
Tests
+
Search tests...
JavaTpoint*
Runs: 1
Failures: 1
Log
Reference

https://www.google.co.in

| | Command | Target | Value |
|---|---|---|---|
| 1. | open | / | |
| 2. | type | id=lst-ib | javatpoint tutorials |
| 3. | send keys | id=lst-ib | ${KEY_ENTER} |
| 4. | mouse over | css=h3.r > a | |
| 5. | mouse out | css=h3.r > a | |
| 6. | click at | css=h3.r > a | 147,11 |
| 7. | mouse over | css=img.imageright | |
| 8. | mouse out | css=img.imageright | |

Command: type
Target: id=lst-ib
Value: javatpoint tutorials
Comment:

- Command:
  Command can be considered as the actual operation/action that is performed on the browser elements. For instance, if you are opening a new URL, the command will be 'open'; if you are clicking on a link or a button on the web page, then the command will be 'clicked'.

- 

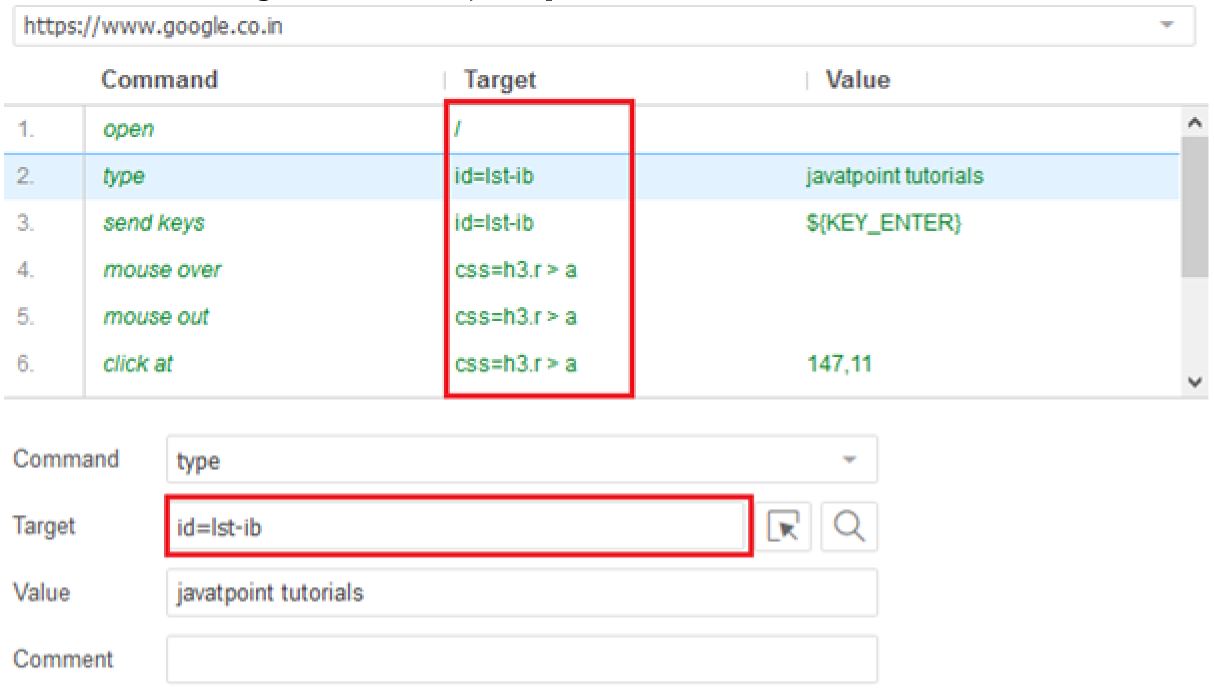
https://www.google.co.in

| | Command | Target | Value |
|---|---|---|---|
| 1. | open | / | |
| 2. | type | id=lst-ib | javatpoint tutorials |
| 3. | send keys | id=lst-ib | ${KEY_ENTER} |
| 4. | mouse over | css=h3.r > a | |
| 5. | mouse out | css=h3.r > a | |
| 6. | click at | css=h3.r > a | 147,11 |

| | |
|---|---|
| Command | open |
| Target | / |
| Value | |
| Comment | |

-

- Target:
  Target specifies the web element on which the operation has to be performed along with a locator attribute. For instance, if you are clicking on a button called javaTpoint, then the target link will be 'javaTpoint'.



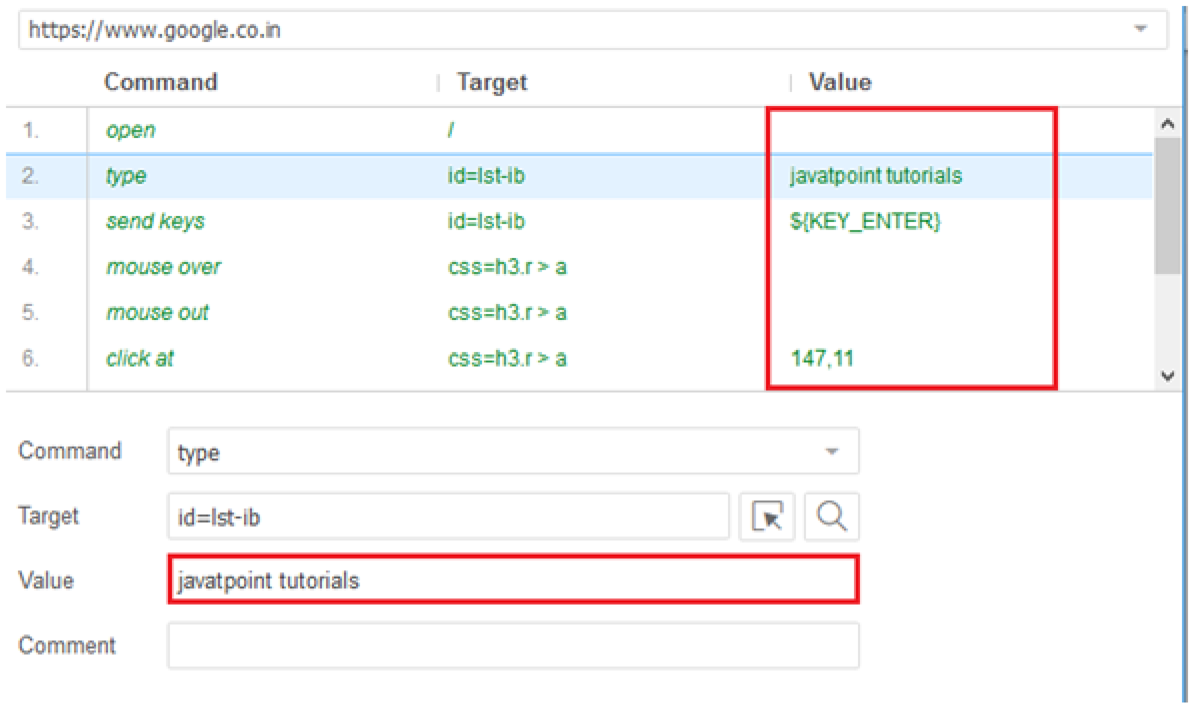
https://www.google.co.in

| | Command | Target | Value |
|---|---|---|---|
| 1. | open | / | |
| 2. | type | id=lst-ib | javatpoint tutorials |
| 3. | send keys | id=lst-ib | ${KEY_ENTER} |
| 4. | mouse over | css=h3.r > a | |
| 5. | mouse out | css=h3.r > a | |
| 6. | click at | css=h3.r > a | 147,11 |

| | |
|---|---|
| Command | type |
| Target | id=lst-ib |
| Value | javatpoint tutorials |
| Comment | |

- Value:
  Value is treated as an optional field and can be used when we need to send some

actual parameters. For instance, if you are entering the email address or password in a textbox, then the value will contain the actual credentials.



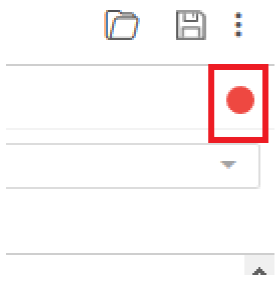
https://www.google.co.in

| | Command | Target | Value |
|---|---|---|---|
| 1. | open | / | |
| 2. | type | id=lst-ib | javatpoint tutorials |
| 3. | send keys | id=lst-ib | ${KEY_ENTER} |
| 4. | mouse over | css=h3.r > a | |
| 5. | mouse out | css=h3.r > a | |
| 6. | click at | css=h3.r > a | 147,11 |

| | |
|---|---|
| Command | type |
| Target | id=lst-ib |
| Value | javatpoint tutorials |
| Comment | |

## 6. Start/Stop Recording Button

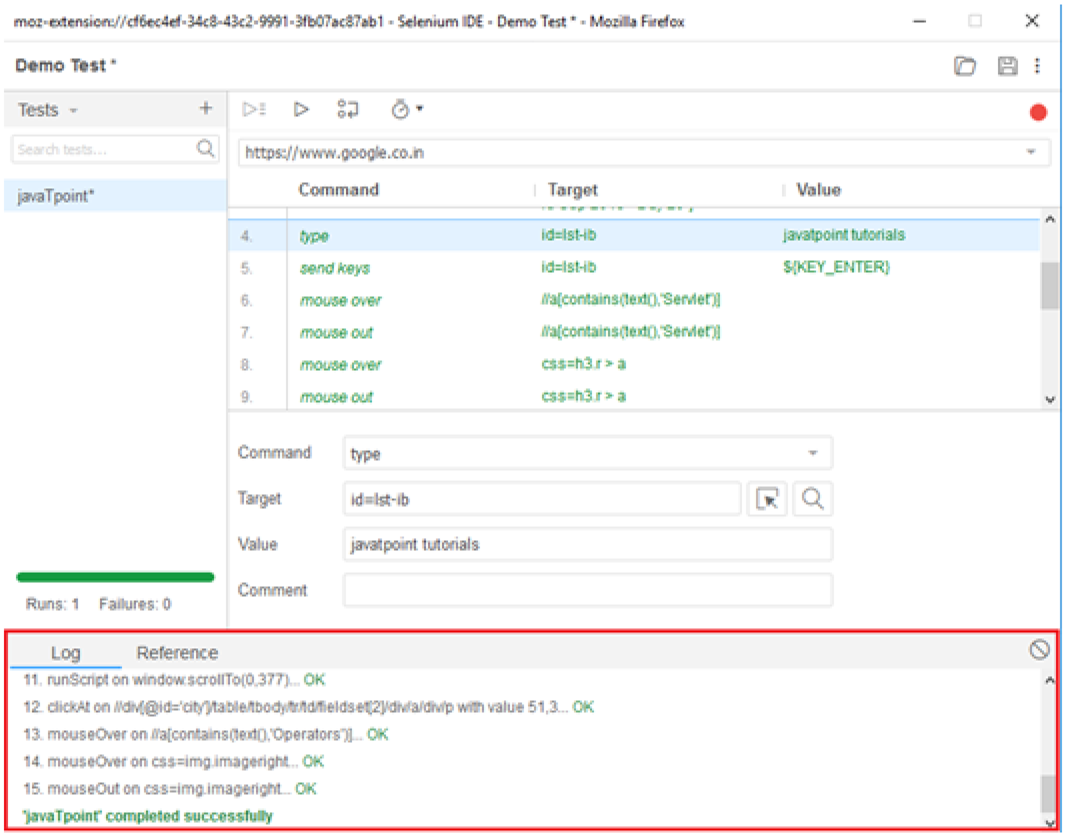
Record button records all of the user actions with the browser.



## 7. Log, Reference Pane

The Log Pane displays the runtime messages during execution. It provides real-time updates of the actions performed by the IDE. It can be categorized into four types: info, error, debug and warn.

The reference Pane displays the complete detail of the currently selected selenese command in the editor.



moz-extension://cf6ec4ef-34c8-43c2-9991-3fb07ac87ab1 - Selenium IDE - Demo Test * - Mozilla Firefox

Demo Test *

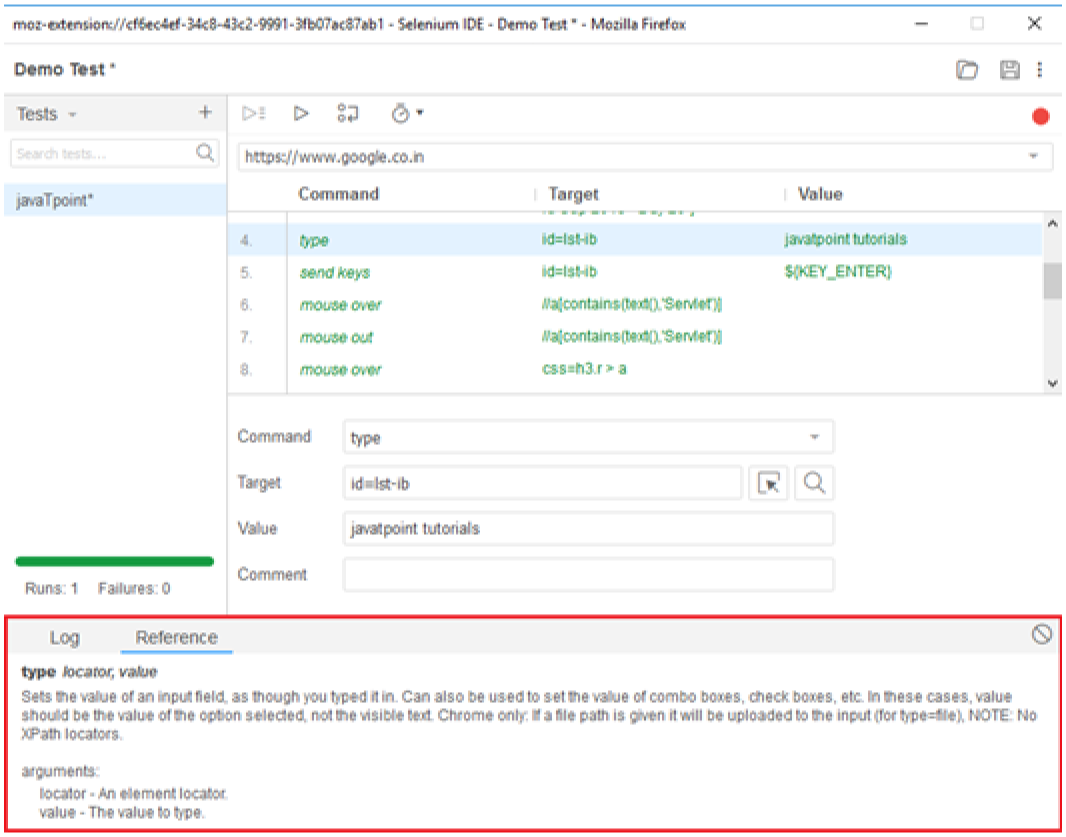
https://www.google.co.in

| | Command | Target | Value |
|---|---|---|---|
| 4. | type | id=lst-ib | javatpoint tutorials |
| 5. | send keys | id=lst-ib | ${KEY_ENTER} |
| 6. | mouse over | //a[contains(text(),'Servlet')] | |
| 7. | mouse out | //a[contains(text(),'Servlet')] | |
| 8. | mouse over | css=h3.r > a | |
| 9. | mouse out | css=h3.r > a | |

Command: type
Target: id=lst-ib
Value: javatpoint tutorials
Comment:

Runs: 1 Failures: 0

```
11. runScript on window.scrollTo(0,377)... OK
12. clickAt on //div[@id='city']/table/tbody/tr/td/fieldset[2]/div/a/div/p with value 51,3... OK
13. mouseOver on //a[contains(text(),'Operators')]... OK
14. mouseOver on css=img.imageright... OK
15. mouseOut on css=img.imageright... OK
'javaTpoint' completed successfully
```



moz-extension://cf6ec4ef-34c8-43c2-9991-3fb07ac87ab1 - Selenium IDE - Demo Test * - Mozilla Firefox

Demo Test *

https://www.google.co.in

| | Command | Target | Value |
|---|---|---|---|
| 4. | type | id=lst-ib | javatpoint tutorials |
| 5. | send keys | id=lst-ib | ${KEY_ENTER} |
| 6. | mouse over | //a[contains(text(),'Servlet')] | |
| 7. | mouse out | //a[contains(text(),'Servlet')] | |
| 8. | mouse over | css=h3.r > a | |

Command: type
Target: id=lst-ib
Value: javatpoint tutorials
Comment:

Runs: 1 Failures: 0

**type** *locator, value*

Sets the value of an input field, as though you typed it in. Can also be used to set the value of combo boxes, check boxes, etc. In these cases, value should be the value of the option selected, not the visible text. Chrome only: If a file path is given it will be uploaded to the input (for type=file), NOTE: No XPath locators.

arguments:
- locator - An element locator.
- value - The value to type.

## Selenium IDE- First Test Case

In this section, you will learn how to create a basic test case in Selenium ide.

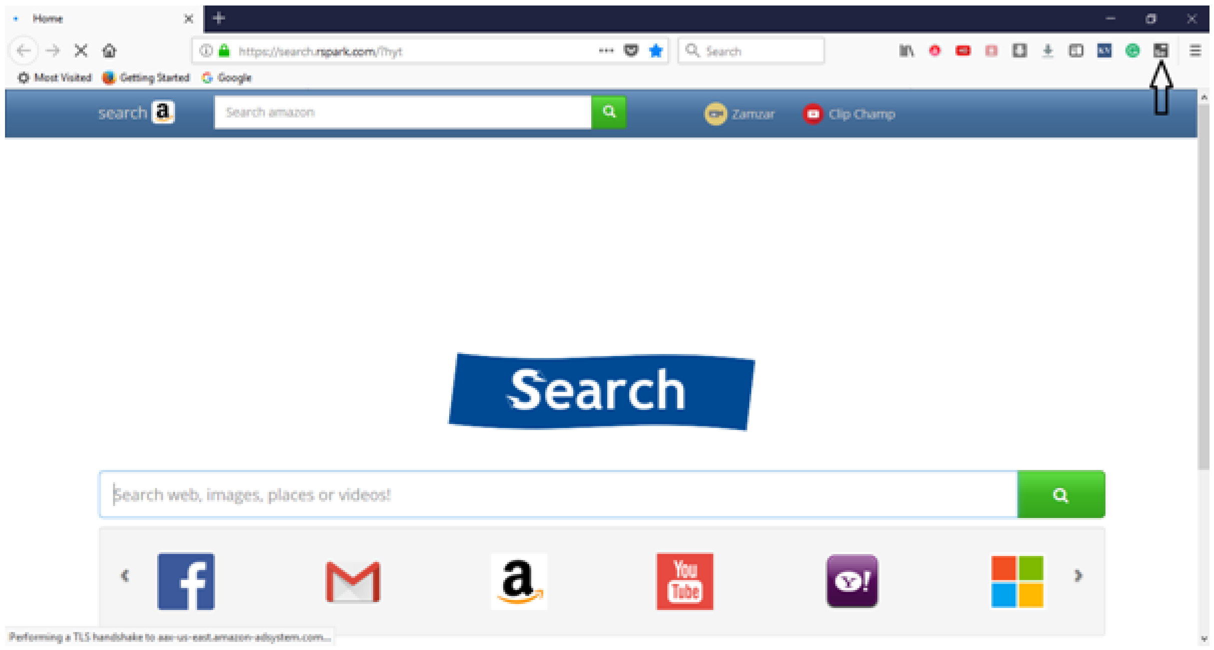
The entire test script creation process in Selenium IDE can be classified into three steps:

- Recording (recording user interactions with the browser)
- Playing back (executing the recorded script)
- Saving the test suite

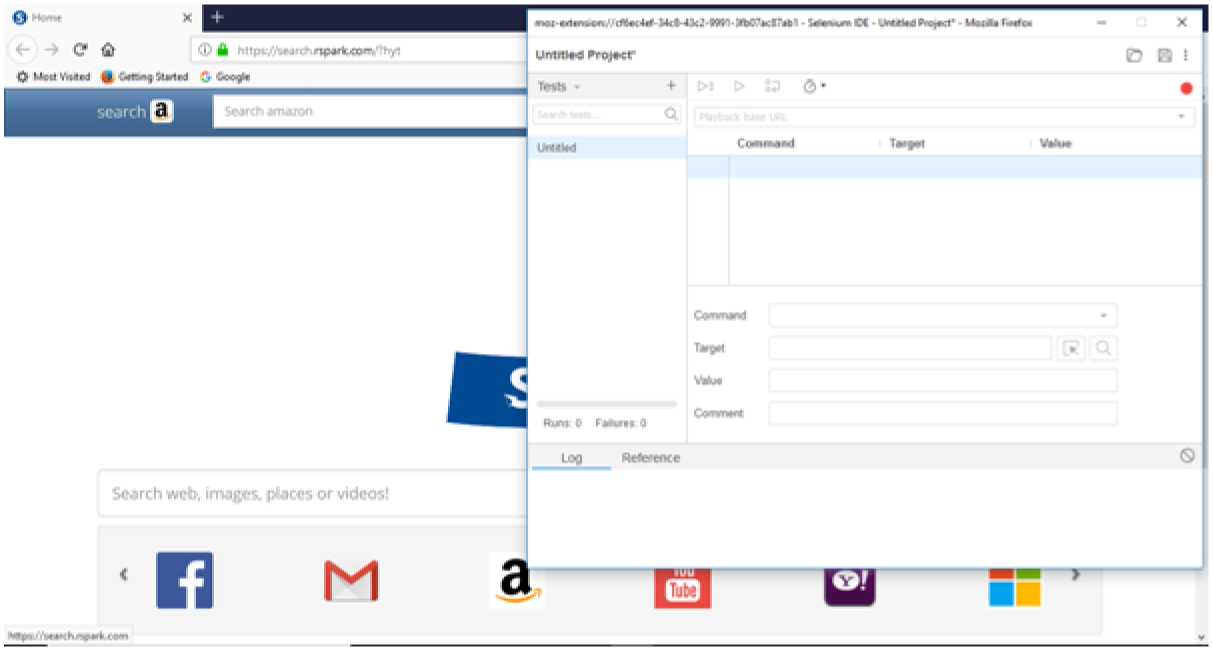
Now, we will see the implementation of the above three steps.

### 1. Recording

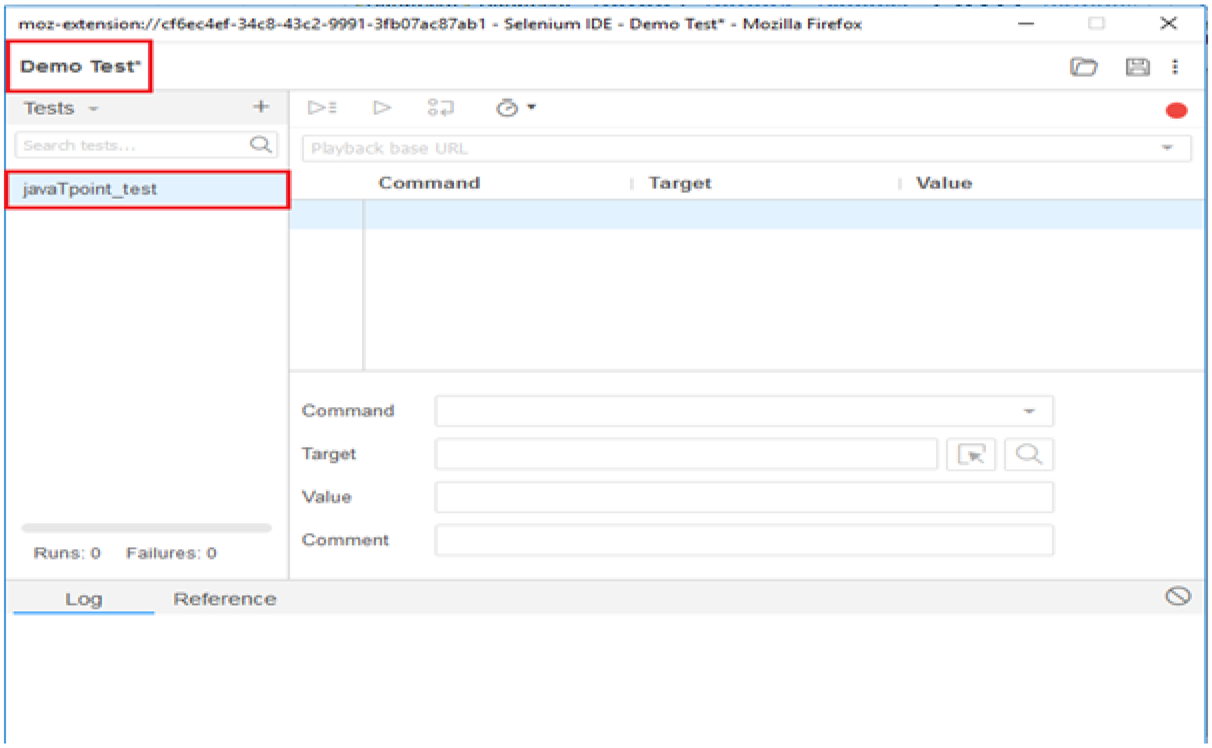
- Launch Firefox browser.
- Click on the Selenium icon present on the top right corner on your browser.


Home
https://search.rspark.com/?hyt
Search
Most Visited
Getting Started
Google
search a
Search amazon
Zamzar
Clip Champ
Search
Search web, images, places or videos!
Performing a TLS handshake to aax-us-east.amazon-adsystem.com...

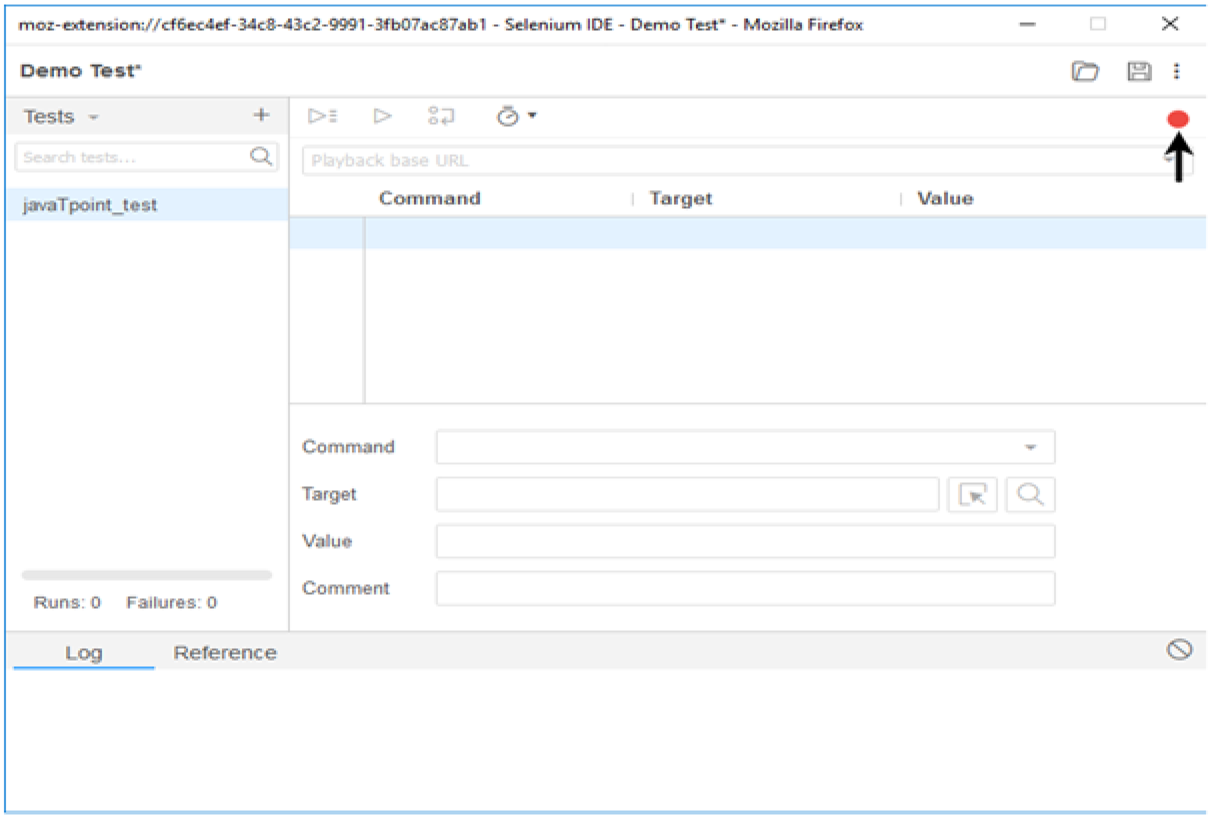
- It will launch the default interface of Selenium IDE.


moz-extension://cf6ec4ef-34c8-43c2-9991-3fb07ac87ab1 - Selenium IDE - Untitled Project* - Mozilla Firefox
Untitled Project*
Tests
Search tests...
Untitled
Playback base URL
Command
Target
Value
Comment
Runs: 0
Failures: 0
Log
Reference
Search web, images, places or videos!

- Rename the project as "Demo Test".
- Rename the test case as "javaTpoint_test".

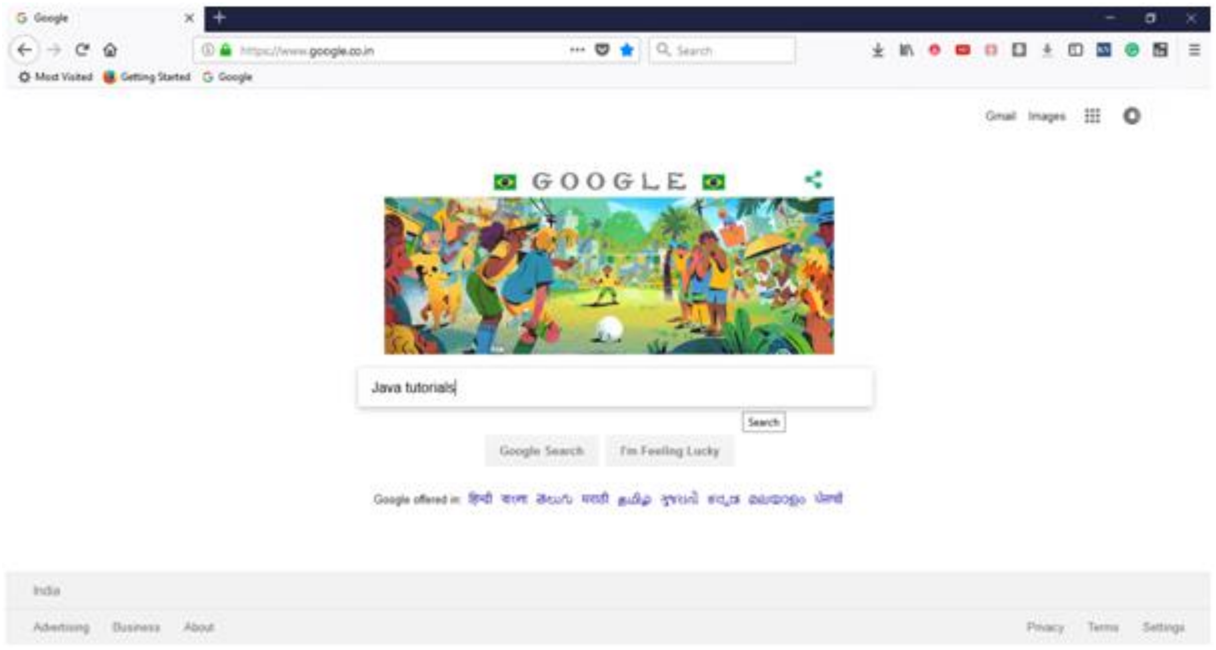


- Click on the "Start Recording" Button present on the top right corner on the IDE to start recording the test case.
-

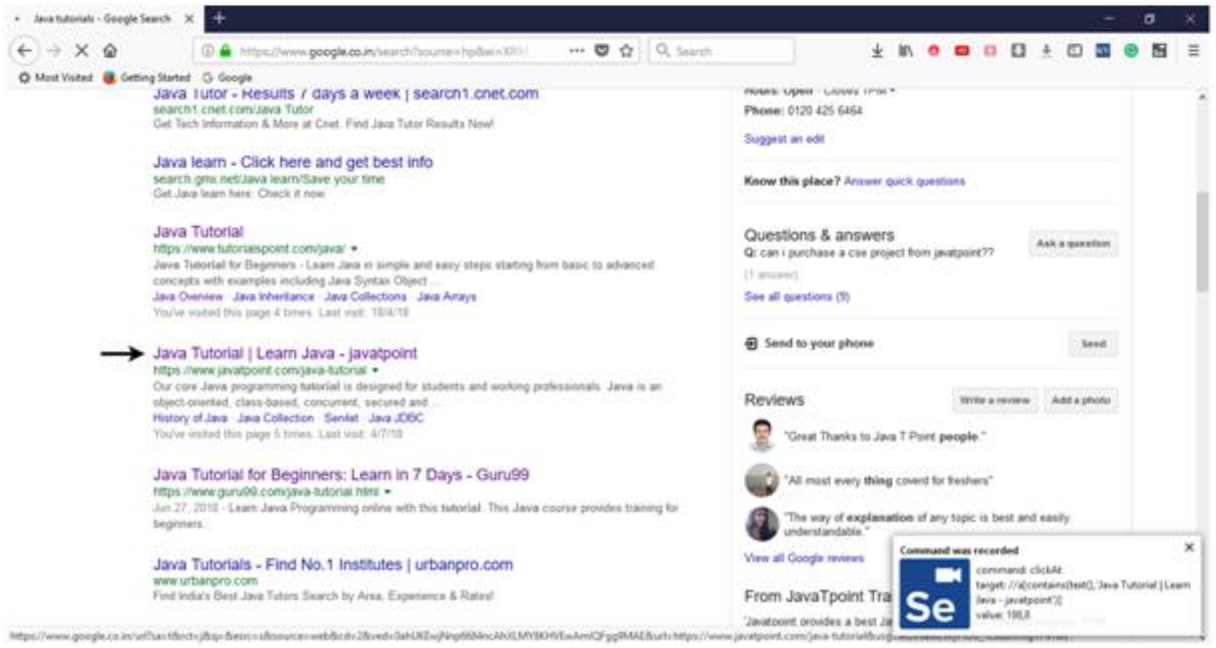


-

- Go to your Firefox browser and open URL:www.google.com
- It will redirect you to the Google search engine page.
- Type "Java Tutorials" in the Google search box.



- Hit enter to get the search results.
- Click on the link "Java Tutorial" provided under the URLhttps://www.javatpoint.com/java-tutorial


Java tutorials - Google Search
https://www.google.co.in/search?source=hp&ei=XRI
Search
Most Visited
Getting Started
Google

Java Tutor - Results 7 days a week | search1.cnet.com
search1.cnet.com/Java Tutor
Get Tech Information & More at Cnet. Find Java Tutor Results Now!

Java learn - Click here and get best info
search.gmx.net/Java learn/Save your time
Get Java learn here. Check it now.

Java Tutorial
https://www.tutorialspoint.com/java/
Java Tutorial for Beginners - Learn Java in simple and easy steps starting from basic to advanced concepts with examples including Java Syntax Object ...
Java Overview · Java Inheritance · Java Collections · Java Arrays
You've visited this page 4 times. Last visit: 18/4/18

Java Tutorial | Learn Java - javatpoint
https://www.javatpoint.com/java-tutorial
Our core Java programming tutorial is designed for students and working professionals. Java is an object-oriented, class-based, concurrent, secured and ...
History of Java · Java Collection · Servlet · Java JDBC
You've visited this page 5 times. Last visit: 4/7/18

Java Tutorial for Beginners: Learn in 7 Days - Guru99
https://www.guru99.com/java-tutorial.html
Jun 27, 2018 - Learn Java Programming online with this tutorial. This Java course provides training for beginners.

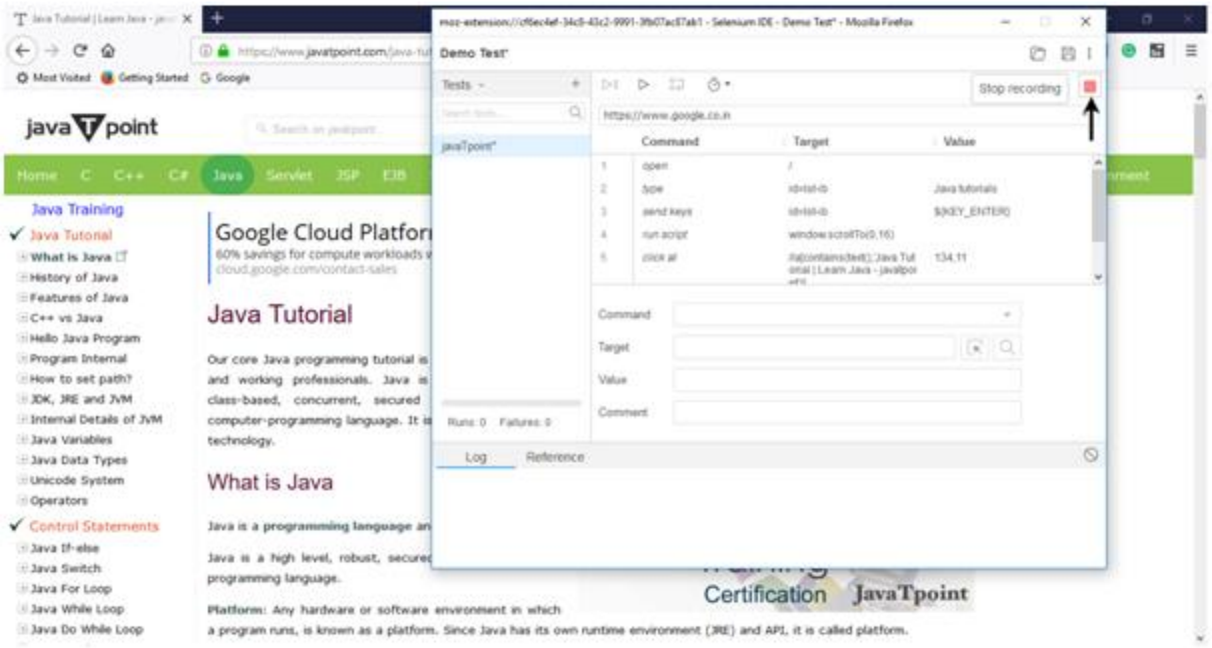
Java Tutorials - Find No.1 Institutes | urbanpro.com
www.urbanpro.com
Find India's Best Java Tutors Search by Area, Experience & Rates!

Phone: 0120 425 6464
Suggest an edit
Know this place? Answer quick questions
Questions & answers
Q: can i purchase a cse project from javatpoint??
(1 answer)
See all questions (9)
Ask a question
Send to your phone
Send
Reviews
Write a review
Add a photo
"Great Thanks to Java T Point people."
"All most every thing coverd for freshers"
"The way of explanation of any topic is best and easily understandable."
View all Google reviews
From JavaTpoint Tra
'Javatpoint provides a best Ja

Command was recorded
command: clickAt
target: //a[contains(text(),'Java Tutorial | Learn Java - javatpoint')]
value: 198,8

- It will redirect you to the javaTpoint's Java tutorial web page. Meanwhile, you will get the notifications of the actions performed by the IDE at the extreme right corner of your web browser.

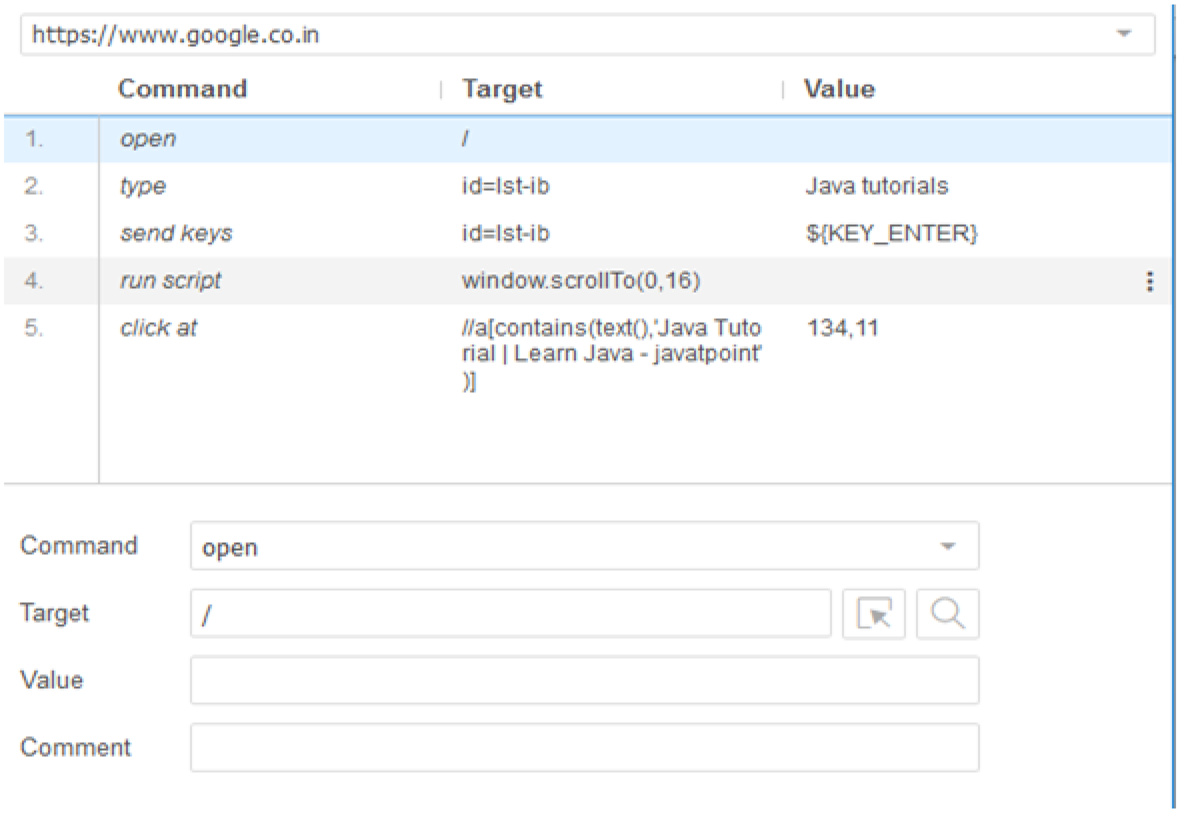
- Now, go the IDE and click on the "Stop Recording" button to stop recording your actions further.



https://www.google.co.in

| | Command | Target | Value |
|---|---|---|---|
| 1 | open | / | |
| 2 | type | id=lst-ib | Java tutorials |
| 3 | send keys | id=lst-ib | ${KEY_ENTER} |
| 4 | run script | window.scrollTo(0,16) | |
| 5 | click at | //a[contains(text(),'Java Tutorial \| Learn Java - javatpoint')] | 134,11 |

- The Test Editor box now contains the list of all of your interactions with the browser.



https://www.google.co.in

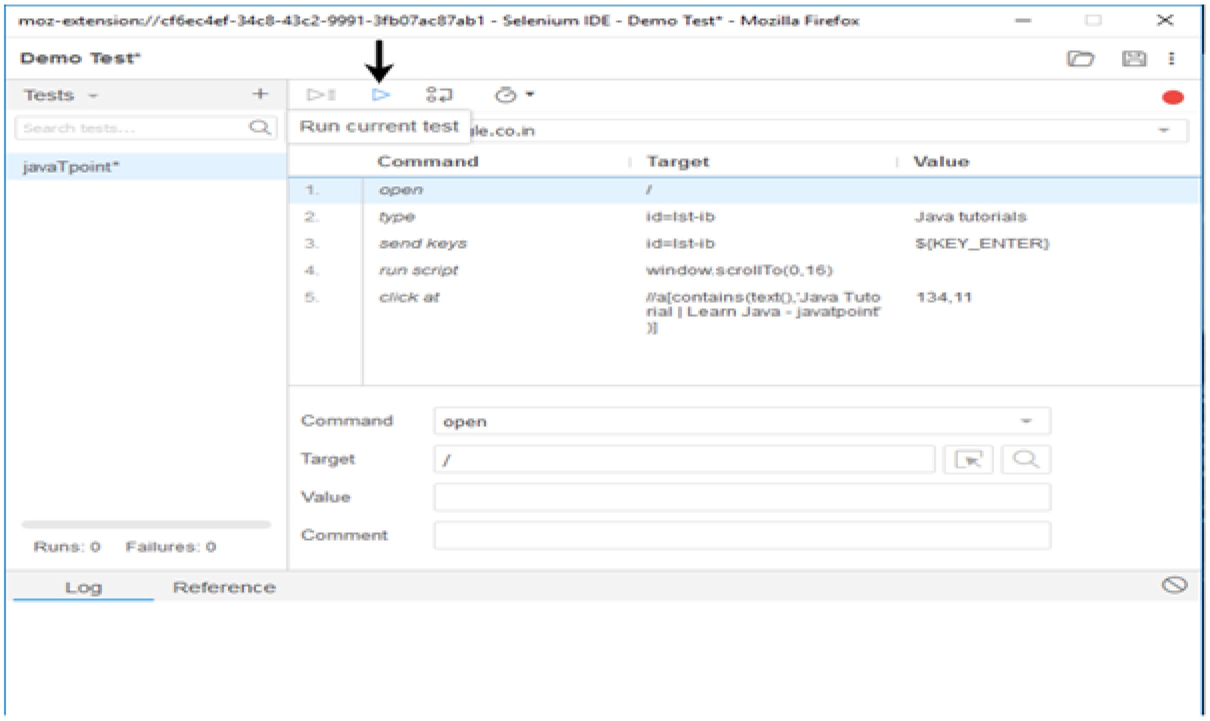
| | Command | Target | Value |
|---|---|---|---|
| 1. | *open* | / | |
| 2. | *type* | id=lst-ib | Java tutorials |
| 3. | *send keys* | id=lst-ib | ${KEY_ENTER} |
| 4. | *run script* | window.scrollTo(0,16) | |
| 5. | *click at* | //a[contains(text(),'Java Tutorial \| Learn Java - javatpoint')] | 134,11 |

| | |
|---|---|
| Command | open |
| Target | / |
| Value | |
| Comment | |

Now, we will proceed to the next step which includes executing the recorded script.

## 2. Playing Back

- Click on the "Run Current Test" button present on the tool bar menu of the IDE. It will execute all of your interactions with the browser and gives you an overall summary of the executed test script.

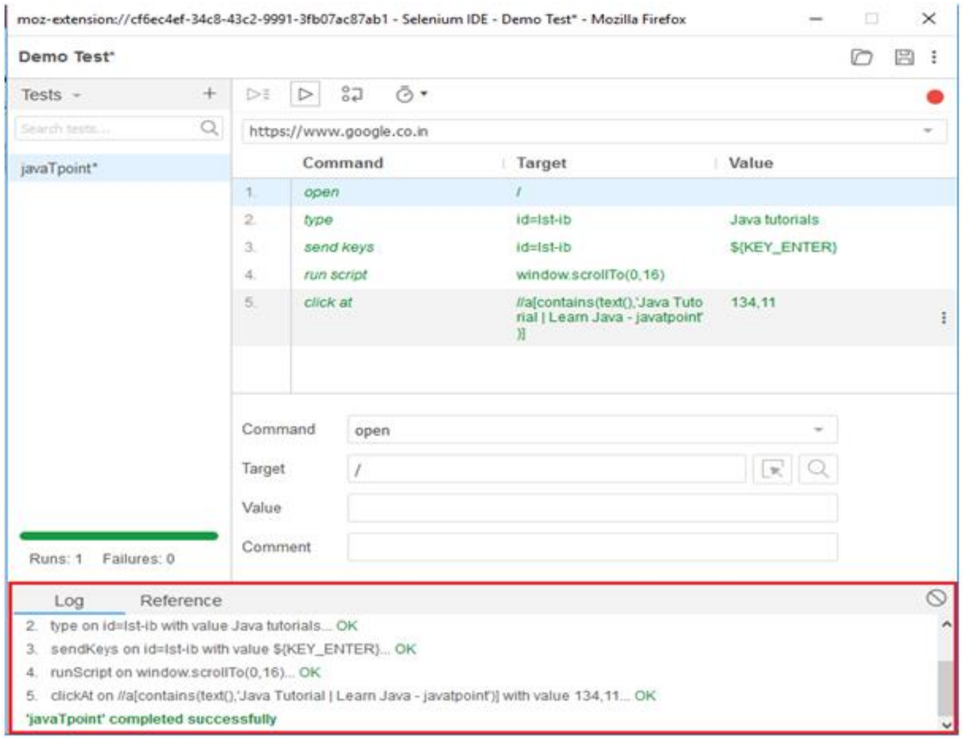

moz-extension://cf6ec4ef-34c8-43c2-9991-3fb07ac87ab1 - Selenium IDE - Demo Test* - Mozilla Firefox

Demo Test*

| | Command | Target | Value |
|---|---|---|---|
| 1. | open | / | |
| 2. | type | id=lst-ib | Java tutorials |
| 3. | send keys | id=lst-ib | ${KEY_ENTER} |
| 4. | run script | window.scrollTo(0,16) | |
| 5. | click at | //a[contains(text(),'Java Tutorial \| Learn Java - javatpoint')] | 134,11 |

| | |
|---|---|
| Command | open |
| Target | / |
| Value | |
| Comment | |

- The Log pane displays the overall summary of the executed test scripts.


moz-extension://cf6ec4ef-34c8-43c2-9991-3fb07ac87ab1 - Selenium IDE - Demo Test* - Mozilla Firefox

Demo Test*

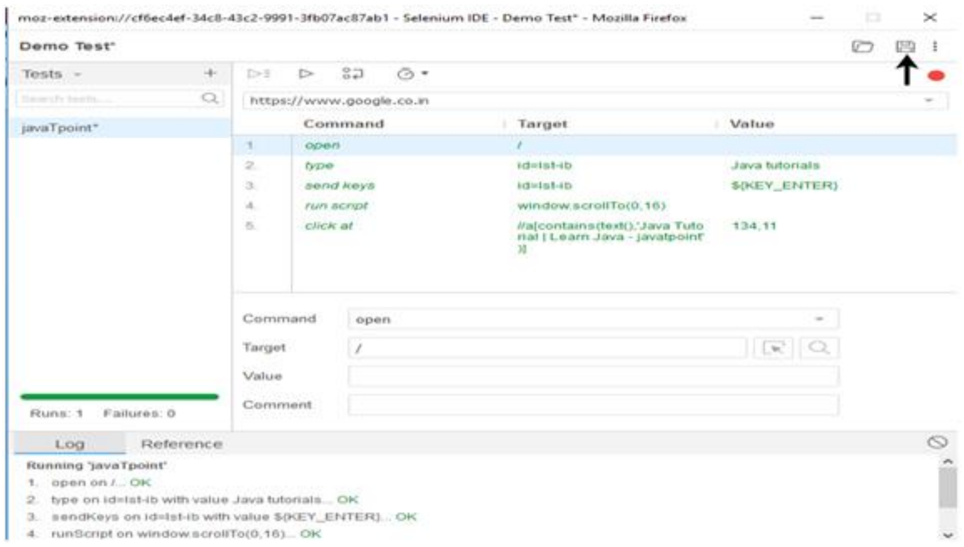
| | Command | Target | Value |
|---|---|---|---|
| 1. | open | / | |
| 2. | type | id=lst-ib | Java tutorials |
| 3. | send keys | id=lst-ib | ${KEY_ENTER} |
| 4. | run script | window.scrollTo(0,16) | |
| 5. | click at | //a[contains(text(),'Java Tutorial \| Learn Java - javatpoint')] | 134,11 |

| | |
|---|---|
| Command | open |
| Target | / |
| Value | |
| Comment | |

```
2. type on id=lst-ib with value Java tutorials... OK
3. sendKeys on id=lst-ib with value ${KEY_ENTER}... OK
4. runScript on window.scrollTo(0,16)... OK
5. clickAt on //a[contains(text(),'Java Tutorial | Learn Java - javatpoint')] with value 134,11... OK
'javaTpoint' completed successfully
```

## 3. Saving the test suite

- Click on the save button present on the extreme right corner of the menu bar.

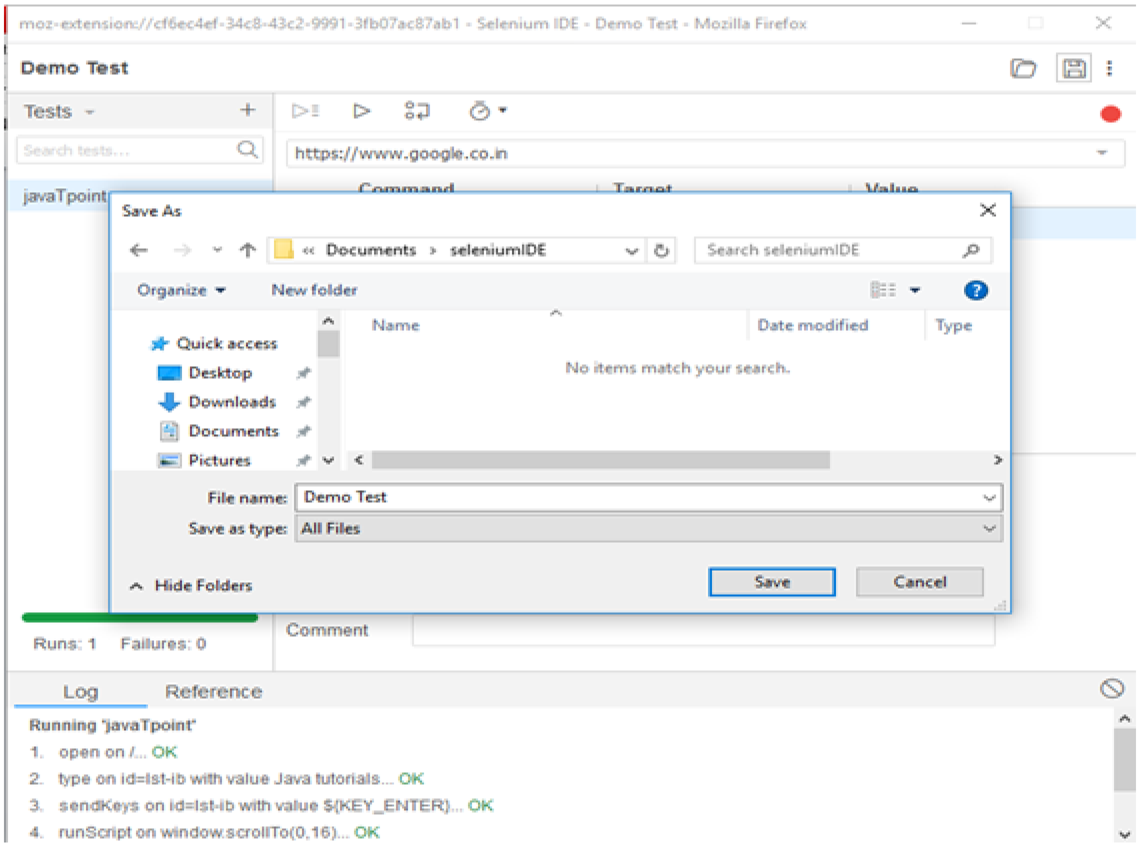

moz-extension://cf6ec4ef-34c8-43c2-9991-3fb07ac87ab1 - Selenium IDE - Demo Test* - Mozilla Firefox
Demo Test*
Tests
javaTpoint*
https://www.google.co.in
Runs: 1 Failures: 0
Log
Reference

| | Command | Target | Value |
|---|---|---|---|
| 1. | open | / | |
| 2. | type | id=lst-ib | Java tutorials |
| 3. | send keys | id=lst-ib | ${KEY_ENTER} |
| 4. | run script | window.scrollTo(0,16) | |
| 5. | click at | //a[contains(text(),'Java Tutorial \| Learn Java - javatpoint')] | 134,11 |

| | |
|---|---|
| Command | open |
| Target | / |
| Value | |
| Comment | |

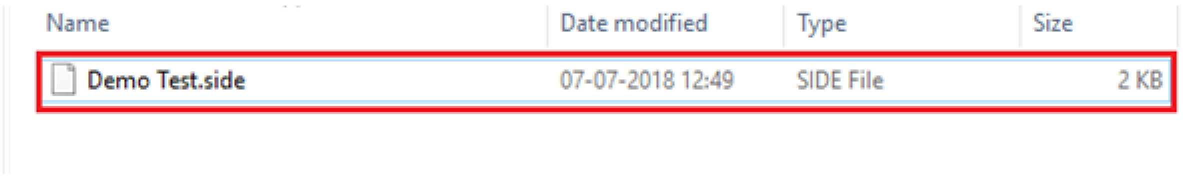
```
Running 'javaTpoint'
1.  open on /... OK
2.  type on id=lst-ib with value Java tutorials... OK
3.  sendKeys on id=lst-ib with value ${KEY_ENTER}... OK
4.  runScript on window.scrollTo(0,16)... OK
```

- Save the entire test suite as "Demo Test".



```
Running 'javaTpoint'
1.  open on /... OK
2.  type on id=lst-ib with value Java tutorials... OK
3.  sendKeys on id=lst-ib with value ${KEY_ENTER}... OK
4.  runScript on window.scrollTo(0,16)... OK
```

- The test suite can be found at the location provided in the above steps. Notice that the test script is saved in .side format.



| Name | Date modified | Type | Size |
|---|---|---|---|
| Demo Test.side | 07-07-2018 12:49 | SIDE File | 2 KB |

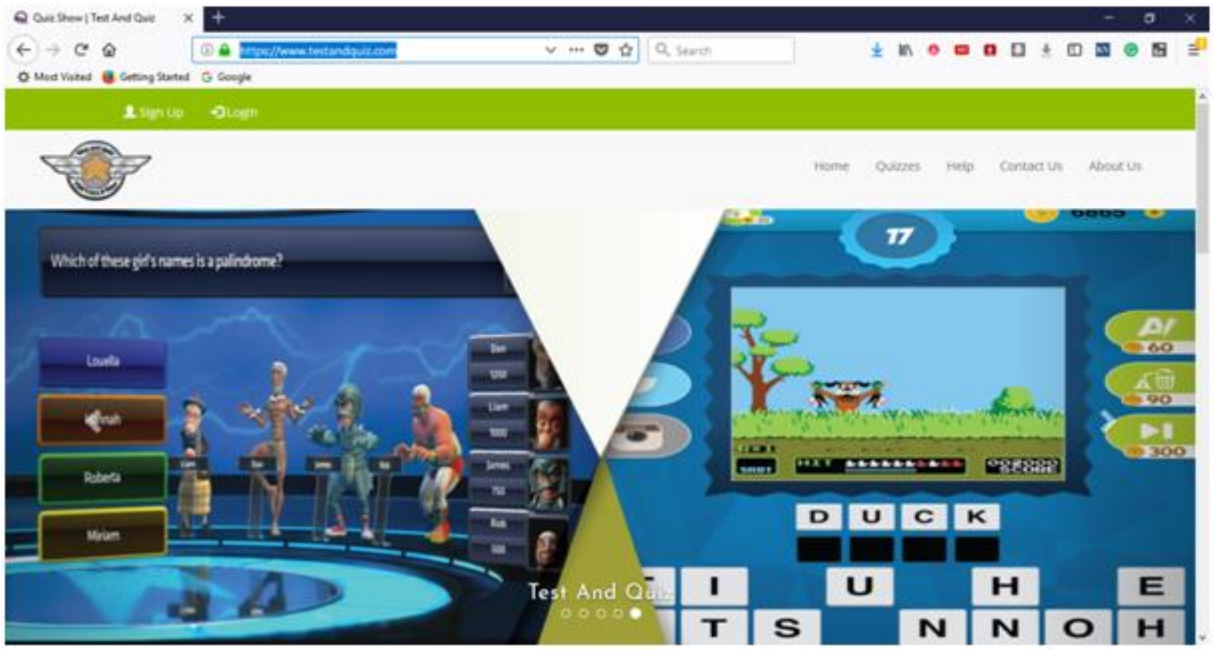
## Selenium IDE- Login Test

In this section, you will learn how to create a Login Test case in Selenium IDE.

For our test purpose, we will be testing the login page provided by the Test and Quiz website, present under the URL:https://www.testandquiz.com/

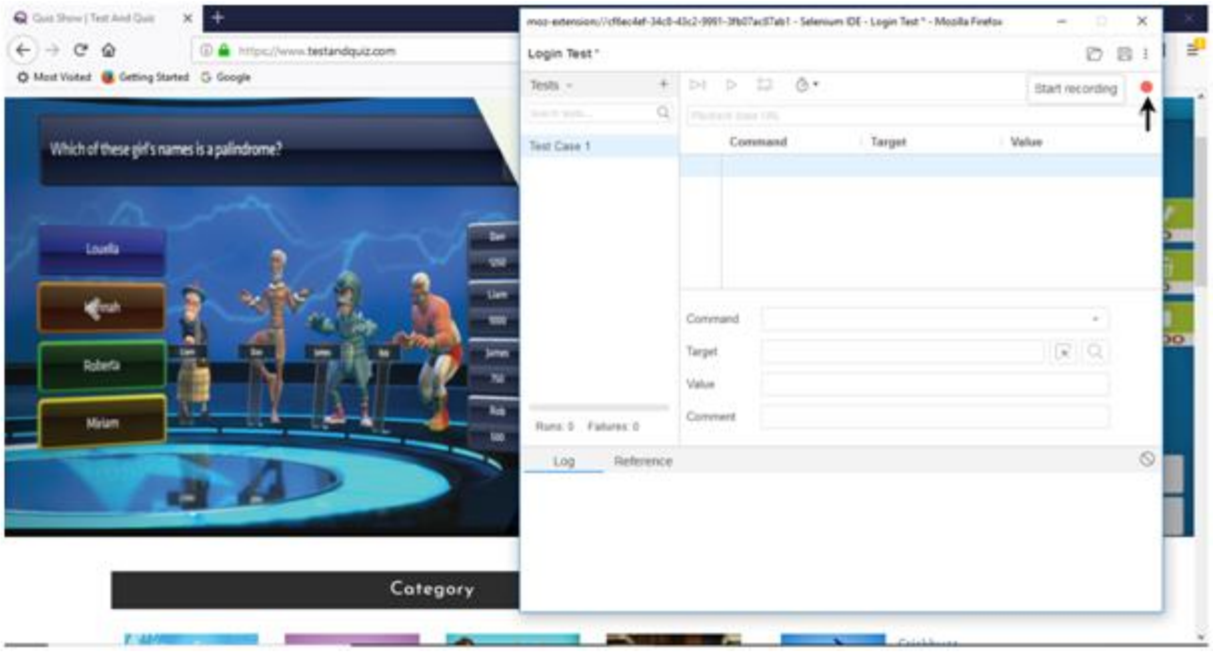
Note:you can sign-up and create a login test on any publically available website.

The following image shows the snapshot of the home page that will appear when we hit the above mentioned URL.

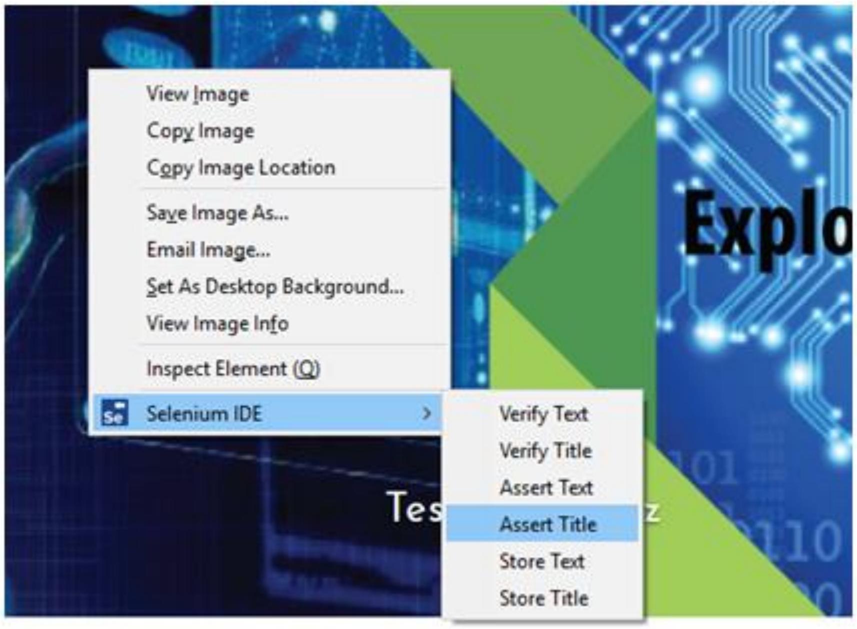

Quiz Show | Test And Quiz
https://www.testandquiz.com
Sign Up
Login
Home
Quizzes
Help
Contact Us
About Us
Which of these girl's names is a palindrome?
Louella
Roberta
Miriam
Test And Quiz
DUCK

First you have sign-up to get the login credentials. For this test, we have already generated our login credentials. Now, we will generate a test script to create a Login Test in Selenium IDE.

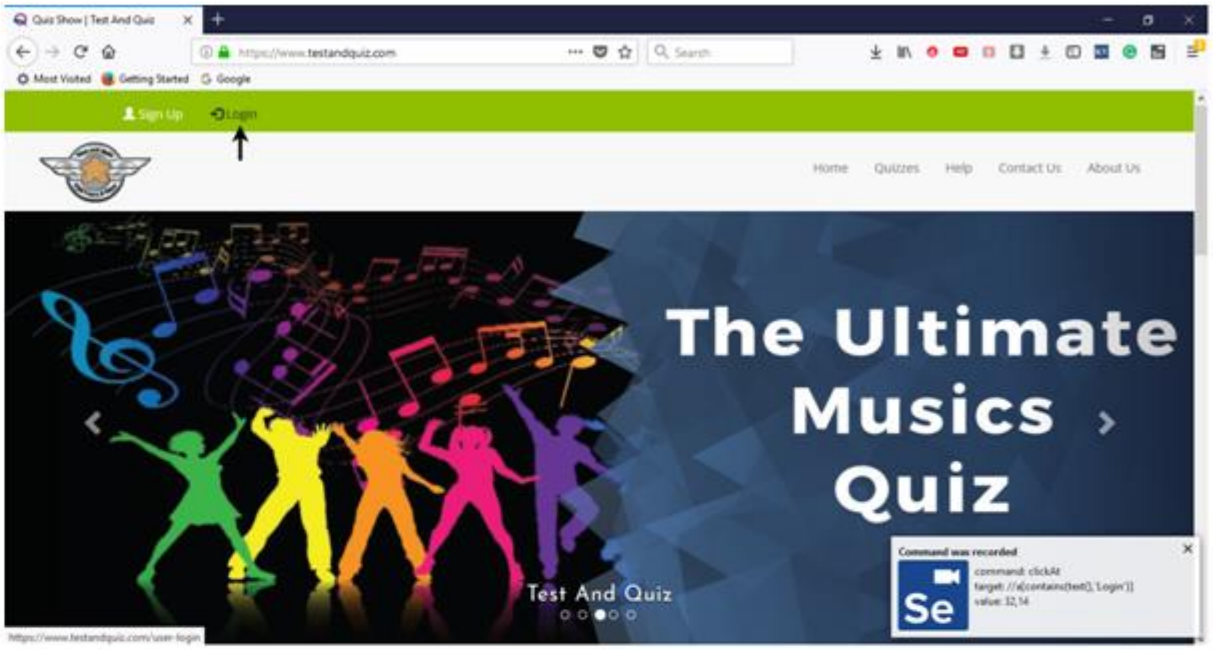
- Launch Firefox browser.
- Click on the Selenium icon present on the top right corner on your browser.
- It will launch the default interface of Selenium IDE.
- Go to your Firefox browser and open URL:https://www.testandquiz.com/
- Enter the project name as "Login Test".
- Enter the test case name as "Test Case 1".
- Click on the "Start Recording" button to start the recording of the test case.



- Go to your Firefox browser and right click on any blank space within the page and select the Selenium IDE option.
- Click on Selenium IDE > Assert Title. The Assert Title command makes sure that the page title is correct.

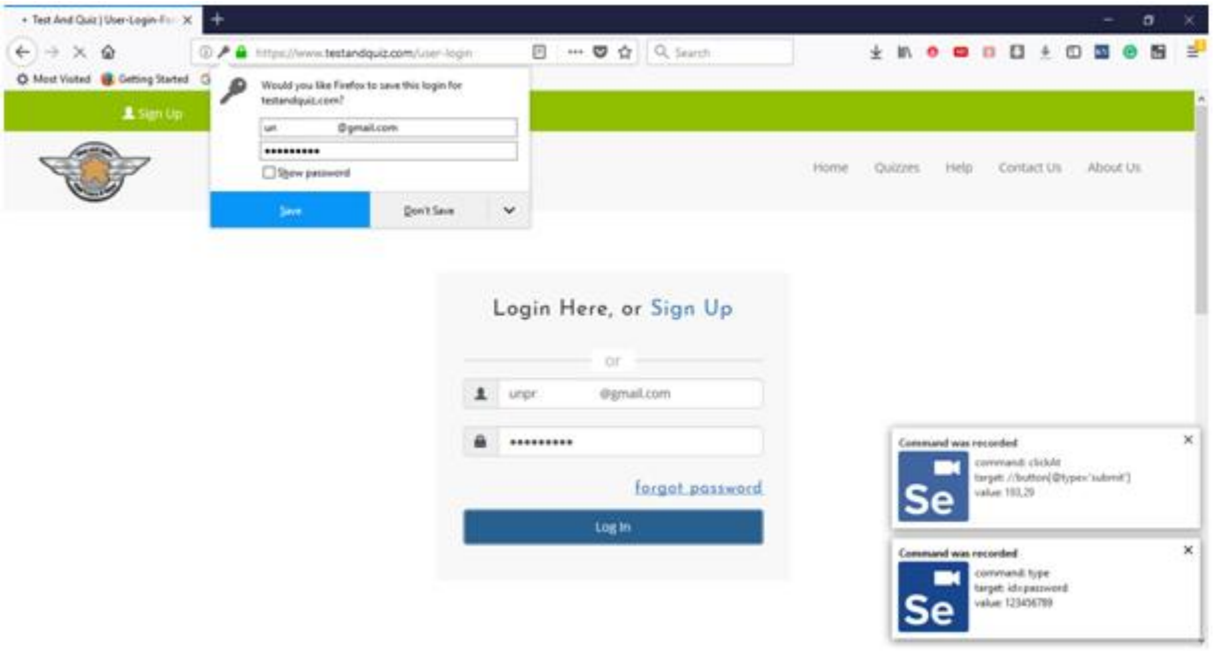


- Now, click on the login button located at the top corner of the website.



Command was recorded
command: clickAt
target: //a[contains(text(),'Login')]
value: 32,14

- Fill-in the login credentials and click on the login button. Meanwhile you will get the notifications of the actions performed by IDE at the extreme right corner of your browser.

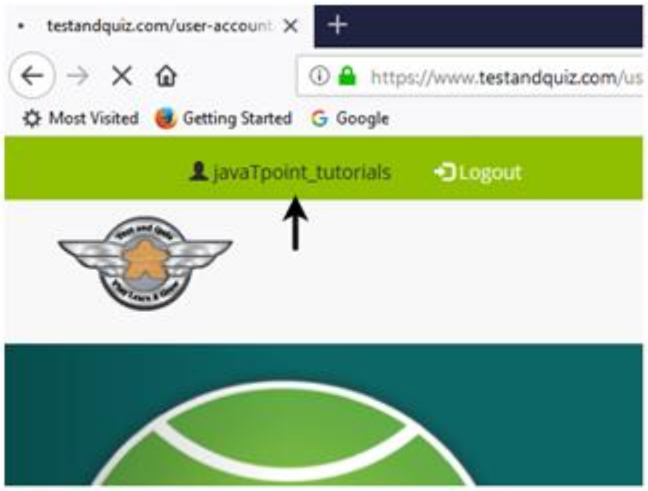


Command was recorded
command: clickAt
target: //button[@type='submit']
value: 193,29

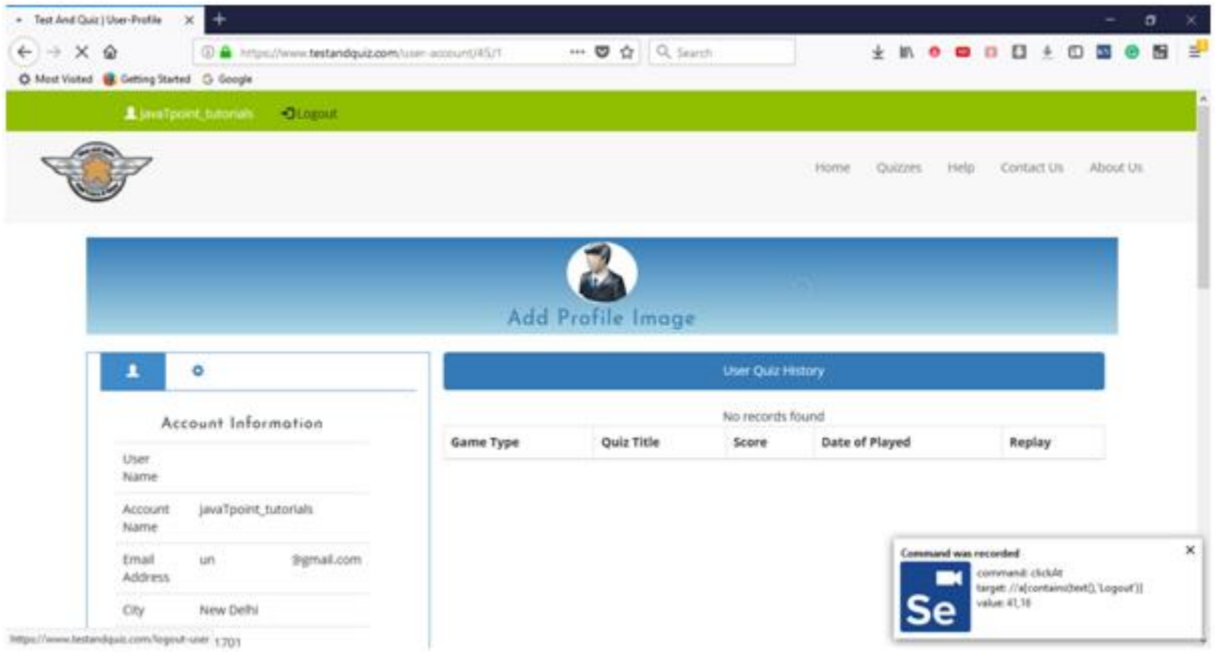
Command was recorded
command: type
target: id=password
value: 123456789

-

- Once you get logged-in, click on the user name section to view your account details.



- It will redirect you to your account settings page, where you can edit your personal details.



Account Information

| | |
|---|---|
| User Name | |
| Account Name | javaTpoint_tutorials |
| Email Address | un @gmail.com |
| City | New Delhi |

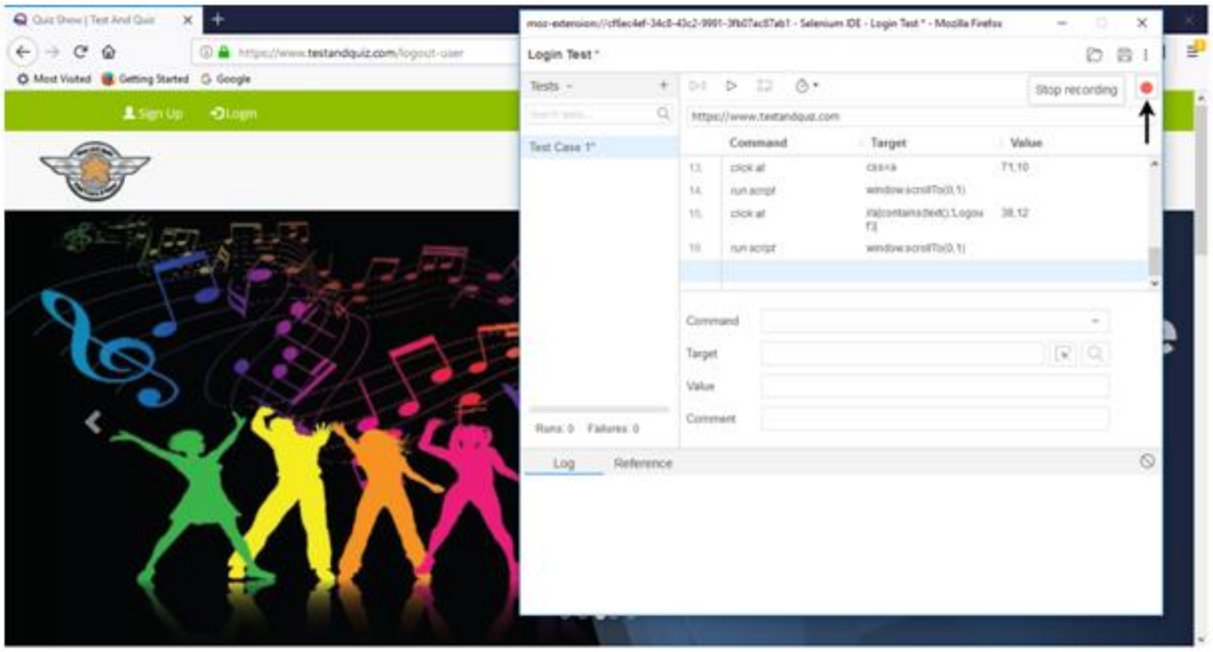
No records found

| Game Type | Quiz Title | Score | Date of Played | Replay |
|---|---|---|---|---|

Command was recorded

```
command: clickAt
target: //a[contains(text(),'Logout')]
value: 41,18
```

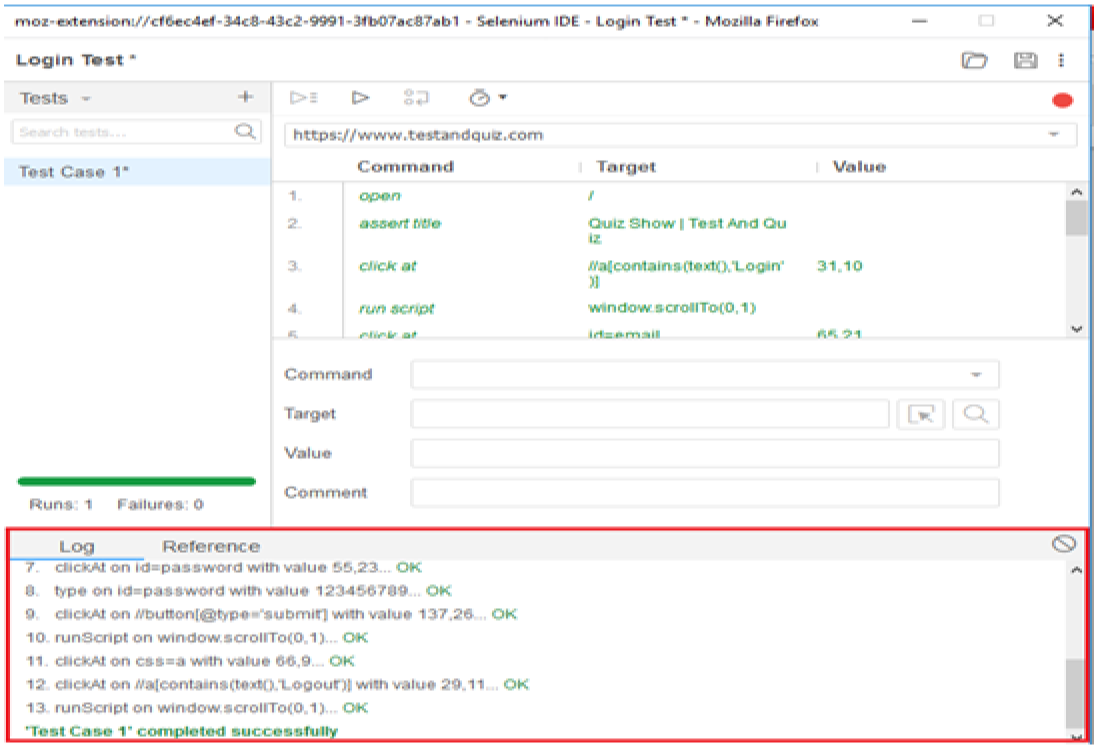
- Click on the Logout button.
- Now, go to the IDE and click on Stop Recording button to stop recording the test case.



| | Command | Target | Value |
|---|---|---|---|
| 13. | click at | css=a | 71,10 |
| 14. | run script | window.scrollTo(0,1) | |
| 15. | click at | //a[contains(text(),'Logout')] | 38,12 |
| 16. | run script | window.scrollTo(0,1) | |

## 2. Playing Back

- Click on the "Run Current Test" button present on the tool bar menu of the IDE. It will execute all of your interactions with the browser and gives you an overall summary of the executed test script.
- The Log pane displays the overall summary of the executed test scripts.



moz-extension://cf6ec4ef-34c8-43c2-9991-3fb07ac87ab1 - Selenium IDE - Login Test * - Mozilla Firefox

Login Test *

https://www.testandquiz.com

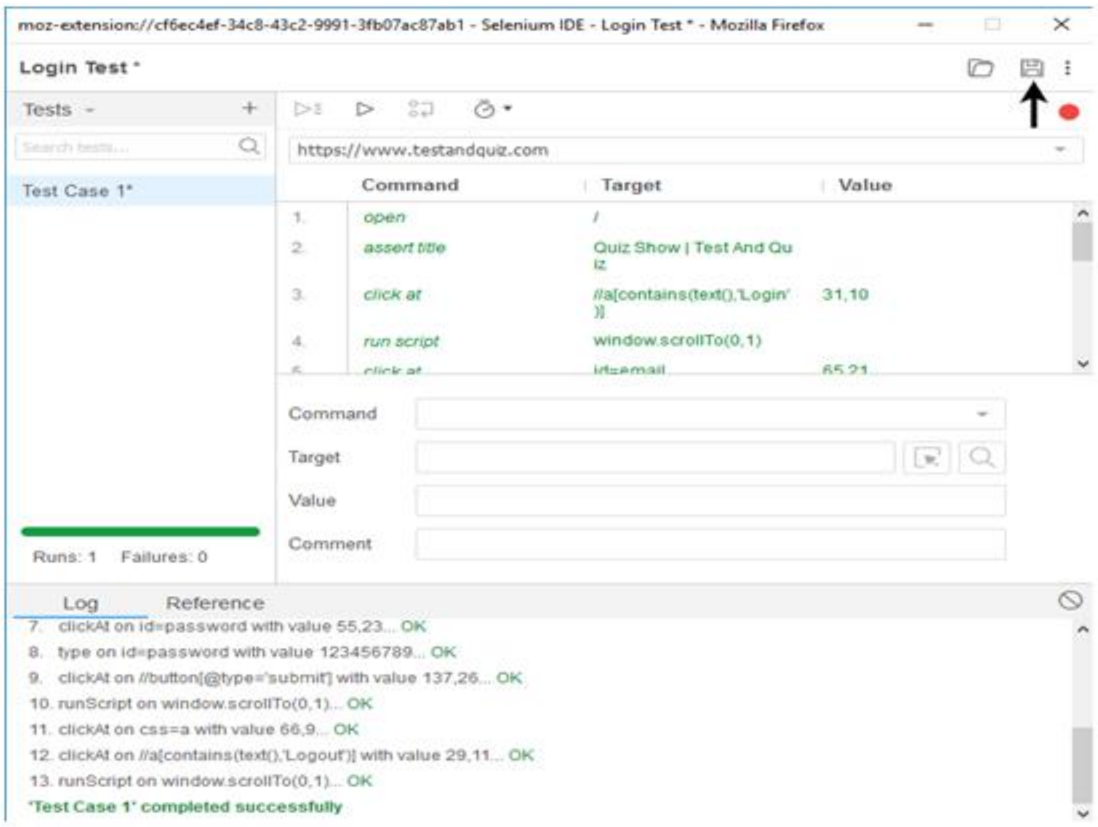
| | Command | Target | Value |
|---|---|---|---|
| 1. | open | / | |
| 2. | assert title | Quiz Show \| Test And Quiz | |
| 3. | click at | //a[contains(text(),'Login')] | 31,10 |
| 4. | run script | window.scrollTo(0,1) | |
| 5. | click at | id=email | 65,21 |

Runs: 1 Failures: 0

```
7.  clickAt on id=password with value 55,23... OK
8.  type on id=password with value 123456789... OK
9.  clickAt on //button[@type='submit'] with value 137,26... OK
10. runScript on window.scrollTo(0,1)... OK
11. clickAt on css=a with value 66,9... OK
12. clickAt on //a[contains(text(),'Logout')] with value 29,11... OK
13. runScript on window.scrollTo(0,1)... OK
'Test Case 1' completed successfully
```

## 3. Saving the test suite

- Click on the save button present on the extreme right corner of the menu bar.


moz-extension://cf6ec4ef-34c8-43c2-9991-3fb07ac87ab1 - Selenium IDE - Login Test * - Mozilla Firefox

Login Test *

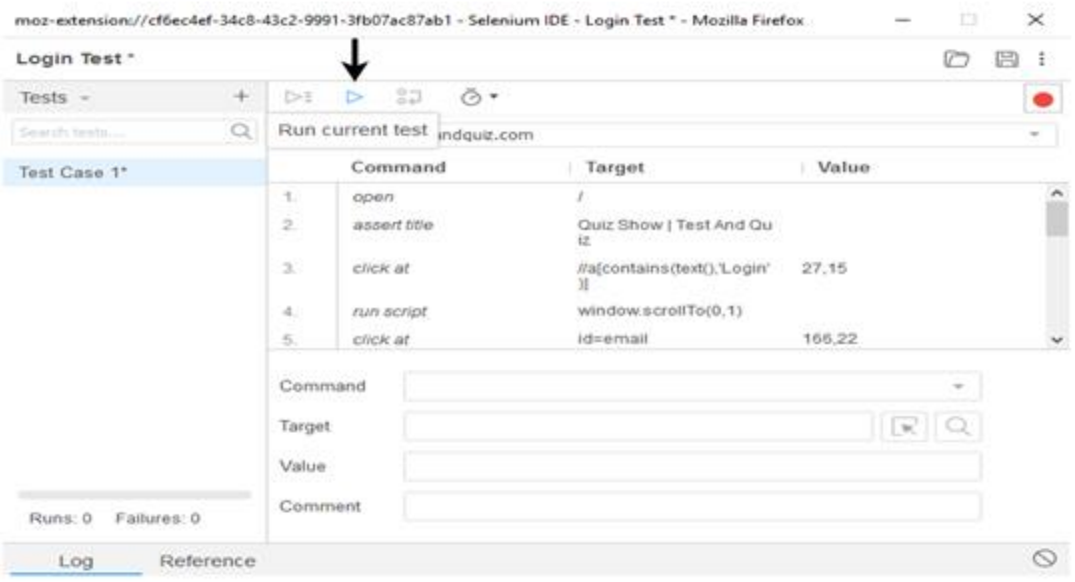
https://www.testandquiz.com

| | Command | Target | Value |
|---|---|---|---|
| 1. | open | / | |
| 2. | assert title | Quiz Show \| Test And Quiz | |
| 3. | click at | //a[contains(text(),'Login')] | 31,10 |
| 4. | run script | window.scrollTo(0,1) | |
| 5. | click at | id=email | 65,21 |

Test Case 1*

Runs: 1 Failures: 0

```
7. clickAt on id=password with value 55,23... OK
8. type on id=password with value 123456789... OK
9. clickAt on //button[@type='submit'] with value 137,26... OK
10. runScript on window.scrollTo(0,1)... OK
11. clickAt on css=a with value 66,9... OK
12. clickAt on //a[contains(text(),'Logout')] with value 29,11... OK
13. runScript on window.scrollTo(0,1)... OK
'Test Case 1' completed successfully
```


moz-extension://cf6ec4ef-34c8-43c2-9991-3fb07ac87ab1 - Selenium IDE - Login Test * - Mozilla Firefox

Login Test *

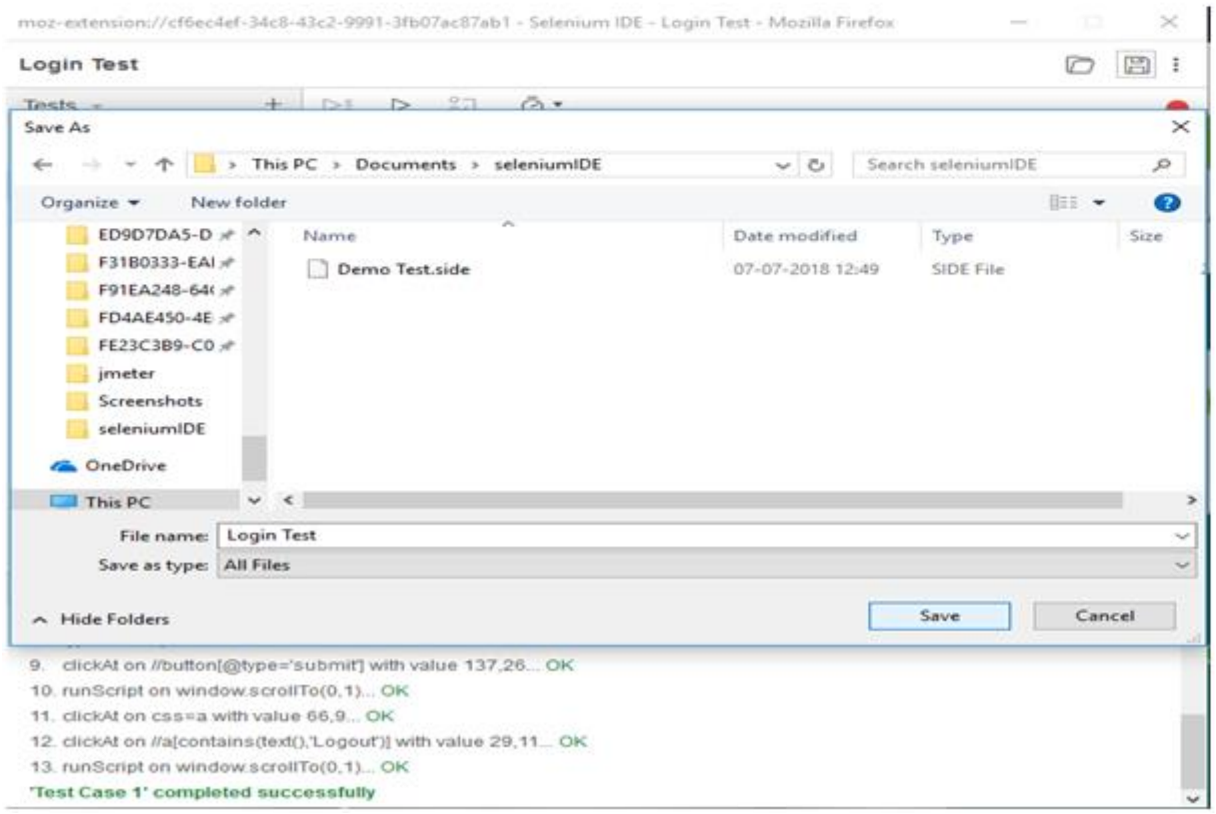
Run current test

| | Command | Target | Value |
|---|---|---|---|
| 1. | open | / | |
| 2. | assert title | Quiz Show \| Test And Quiz | |
| 3. | click at | //a[contains(text(),'Login')] | 27,15 |
| 4. | run script | window.scrollTo(0,1) | |
| 5. | click at | id=email | 166,22 |

Test Case 1*

Runs: 0 Failures: 0

- Save the entire test suite as "Login Test".



File name: Login Test
Save as type: All Files

```
9.  clickAt on //button[@type='submit'] with value 137,26... OK
10. runScript on window.scrollTo(0,1)... OK
11. clickAt on css=a with value 66,9... OK
12. clickAt on //a[contains(text(),'Logout')] with value 29,11... OK
13. runScript on window.scrollTo(0,1)... OK
'Test Case 1' completed successfully
```

- The test suite can be found at the location provided in the above steps.

## Selenium IDE- Commands (Selenese)

Selenium commands, also known as "Selenese" are the set of commands used in Selenium IDE that run your tests. Using selenese, one can perform activities like:

- Testing the existence of UI elements based on their HTML tags.
- Test for specific content.
- Test for broken links.
- Testing input fields, selection list options, submitting forms and table data among other things.
- Testing of window size, mouse options, alerts, Ajax functionality, pop-up windows, event handling and many other web application features.

A sequence of Selenium commands (Selenese) together is known as test script.

## Types of Selenium Commands

Selenium commands are basically classified in three categories:

1. Actions
2. Accessors
3. Assertions

## 1. Actions

Actions are the selenium commands that generally manipulate the state of the application. Execution of Actions generates events like click this link, select that option, type this box, etc. If an Action fails, or has a bug, the execution of current test is stopped.

Some of the most commonly used Actions commands include:

| **Command/Syntax** | **Description** |
|---|---|
| open (url) | It launches the desired URL in the specified browser and it accepts both relative and absolute URLs. |
| type (locator,value) | It sets the value of an input field, similar to user typing action. |
| typeKeys (locator,value) | This command simulates keystroke events on the specified element. |
| click (locator) | This command enables clicks on a link, button, checkbox or radio button. |
| clickAt (locator,coordString) | This command enables clicks on an element with the help of locator and co-ordinates |
| doubleClick (locator) | This command enables double clicks on a webelement based on the specified element. |
| focus (locator) | It moves the focus to the specified element |
| highlight (locator) | It changes the background color of the specified element to yellow to highlight is useful for debugging purposes. |
| close() | This command simulates the user clicking the "close" button in the title bar of a popup window or tab. |
| store (expression,variableName) | This command specifies the name of a variable in which the result is to be stored and expression is the value to store |

| | |
|---|---|
| waitForCondition (script,timeout) | This command executes the specified JavaScript snippet repeatedly until it evaluates to "true". |

## . Accessors

Accessors are the selenium commands that examine the state of the application and store the results in variables. They are also used to automatically generate Assertions.

Some of the most commonly used Accessors commands include:

| Command/Syntax | Description |
|---|---|
| storeTitle (variableName) | This command gets the title of the current page. |
| storeText (locator, variableName) | This command gets the text of an element.. |
| storeValue (locator,variableName) | This command gets the (whitespace-trimmed) value of an input field. |
| storeTable (tableCellAddress, variableName) | This command gets the text from a cell of a table. |
| storeLocation (variableName) | This command gets the absolute URL of the current page. |
| storeElementIndex (locator, variableName) | This command gets the relative index of an element to its parent (starting from 0). |
| storeBodyText (variableName) | This command gets the entire text of the page. |
| storeAllButtons (variableName) | It returns the IDs of all buttons on the page. |
| storeAllFields (variableName) | It returns the IDs of all input fields on the page. |
| storeAllLinks (variableName) | It returns the IDs of all links on the page. |

### 2. Assertions

Assertions are the commands that enable testers to verify the state of the application. Assertions are generally used in three modes assert, verify and waitfor.

Some of the most commonly used Assertions commands are:

| Command/Syntax | Description |
|---|---|
| verifySelected(selectLocator, optionLocator) | This command verifies that the selected option of a drop-down satisfies the optionSpecifier. |
| verifyAlert (pattern) | This command verifies the alert text; used with accessorstoreAlert. |
| verifyAllButtons (pattern) | This command verifies the button which is used withaccessorstoreAllButtons. |
| verifyAllLinks (pattern) | This command verifies all links; used with the accessorstoreAllLinks. |
| verifyBodyText(pattern) | This command verifies the body text; used with the accessorstoreBodyText. |
| verifyAttribute(attributeLocator, pattern) | This command verifies an attribute of an element; used with the accessorstoreAttribute. |
| waitForErrorOnNext (message) | This command enables Waits for error; used with the accessorassertErrorOnNext. |
| waitForAlert (pattern) | This command enables waits for the alert; used with the accessorstoreAlert. |
| verifyAllWindowIds (pattern) | This command verifies the window id; used with the accessorstoreAllWindowIds. |

## Selenium IDE- Creating Test Cases Manually

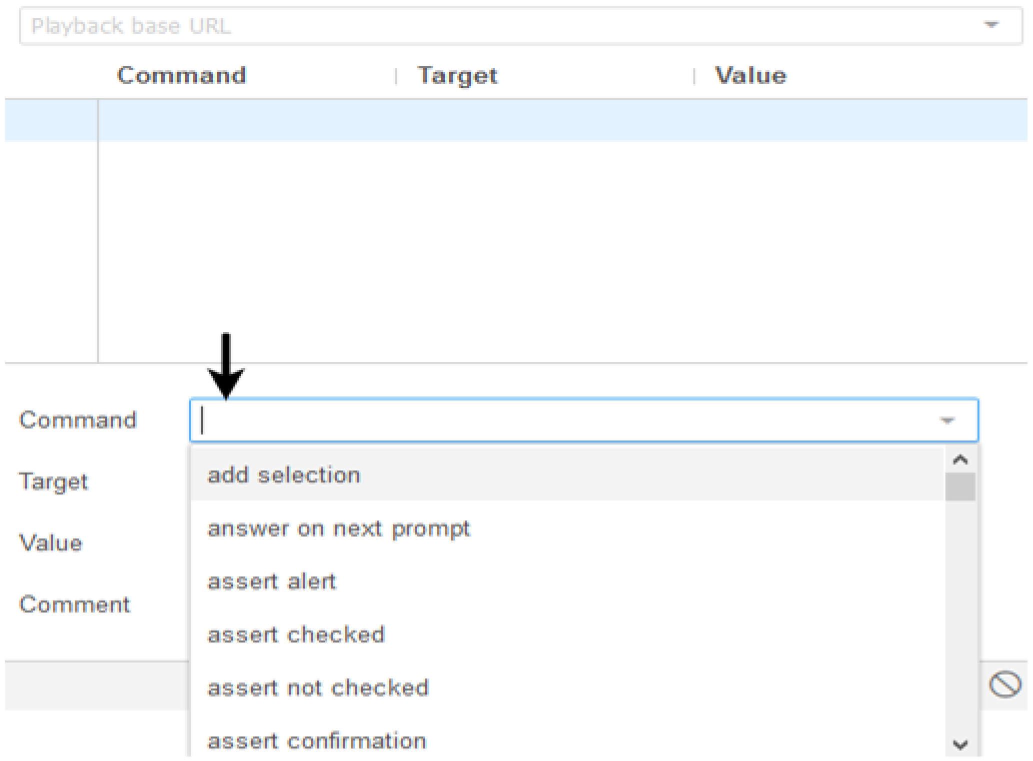
In this section, you will learn how to create a test case manually using Selenium commands in Selenium IDE. In simple words, we will create test cases by inserting selenium commands instead of recording option.

For this test, we will search a text operation on any publically available search engine (say "Google"). Subsequently, we will create a Login test case in the same test suite.

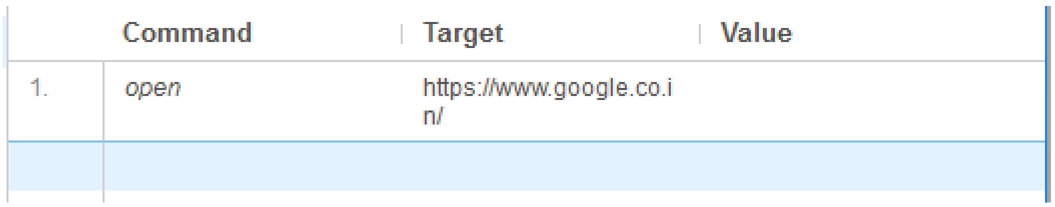
To create a test case manually, first you have to go through the most commonly used selenium commands which we have discussed in the previous section. Now, we will create our first test case on search operation. We will search our text on Google search engine.

## 1. Insert Commands

- Launch Firefox browser.
- Click on the Selenium icon present on the top right corner on your browser.
- It will launch the default interface of Selenium IDE.
- Enter the project name as "Manual Test".
- Enter the test case name as "Search Test".
- Click on the command text box present on the Test Script Editor Box.



- Modify the properties of First command as:
- Command : open
- Target : https://www.google.co.in
- During execution of the test case, this command will load the Google search engine web page on your Firefox browser.

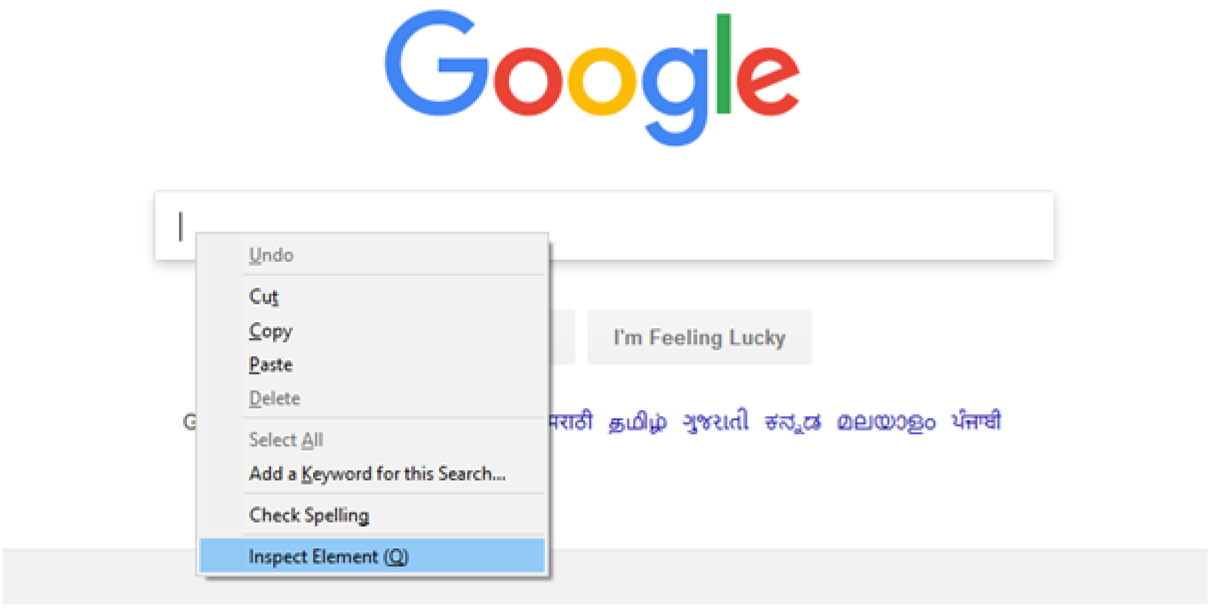


| | Command | Target | Value |
|---|---|---|---|
| 1. | open | https://www.google.co.in/ | |

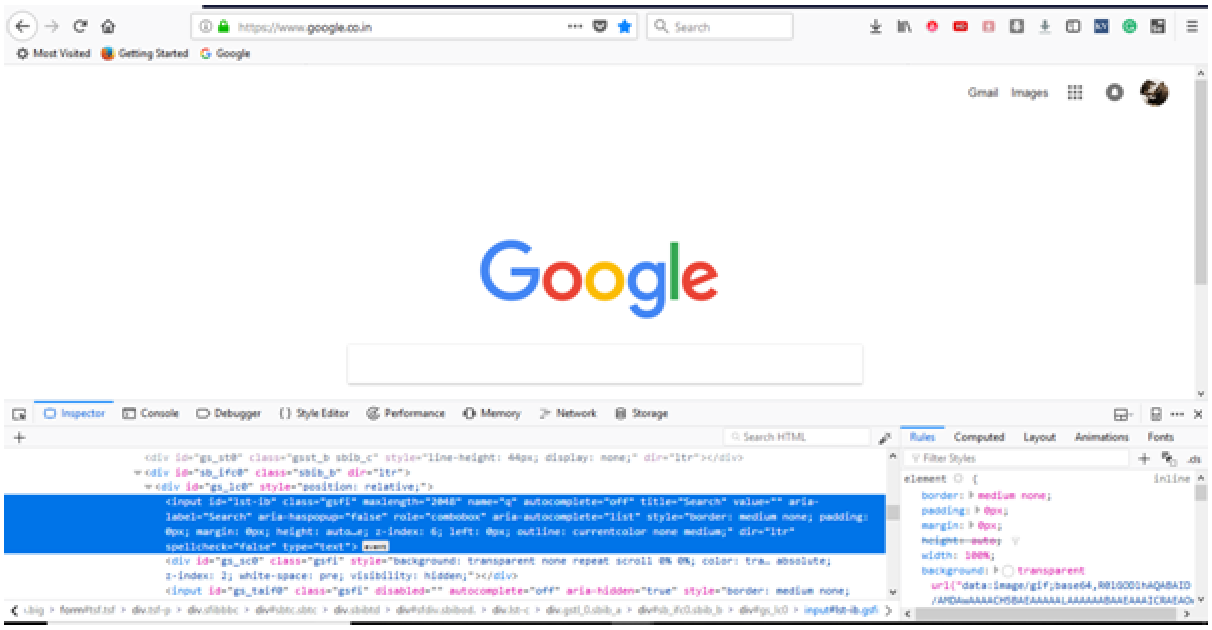
Now, we have to add a command that will click on the Google search engine text box. For this, we need a unique identification element of the text box which will help the IDE to identifythe target location.

The method for finding a unique identification element involves inspection of HTML codes.

- Open URL: https://www.google.co.inyour Firefox browser.
- Right click on the Google search text box and select Inspect Element.

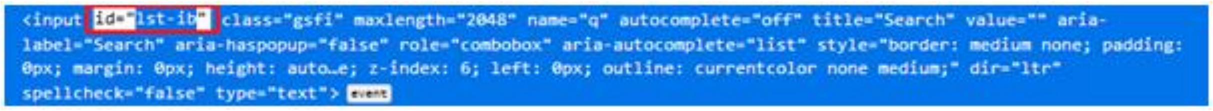


- It will launch a window containing all the specific codes involved in the development of the test box.



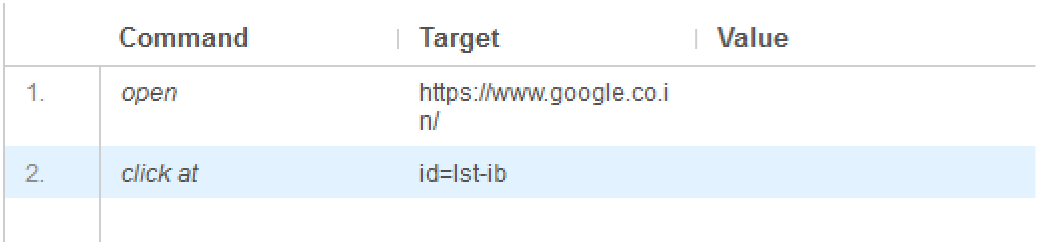
```
<div id="gs_st0" class="gsst_b sbib_c" style="line-height: 44px; display: none;" dir="ltr"></div>
▾<div id="sb_ifc0" class="sbib_b" dir="ltr">
  ▾<div id="gs_lc0" style="position: relative;">
    <input id="lst-ib" class="gsfi" maxlength="2048" name="q" autocomplete="off" title="Search" value="" aria-
    label="Search" aria-haspopup="false" role="combobox" aria-autocomplete="list" style="border: medium none; padding:
    0px; margin: 0px; height: auto…e; z-index: 6; left: 0px; outline: currentcolor none medium;" dir="ltr"
    spellcheck="false" type="text"> event
    <div id="gs_sc0" class="gsfi" style="background: transparent none repeat scroll 0% 0%; color: tra… absolute;
    z-index: 2; white-space: pre; visibility: hidden;"></div>
    <input id="gs_taif0" class="gsfi" disabled="" autocomplete="off" aria-hidden="true" style="border: medium none;
```

- Pick the input tag element that contains an ID name for the text box.



```
<input id="lst-ib" class="gsfi" maxlength="2048" name="q" autocomplete="off" title="Search" value="" aria-
label="Search" aria-haspopup="false" role="combobox" aria-autocomplete="list" style="border: medium none; padding:
0px; margin: 0px; height: auto…e; z-index: 6; left: 0px; outline: currentcolor none medium;" dir="ltr"
spellcheck="false" type="text"> event
```

- Modify the properties of second command as:
- Command :click at
- Target : id=lst-ib
- During execution of the test case, this command will click on the search text box present on the Google search engine web page.

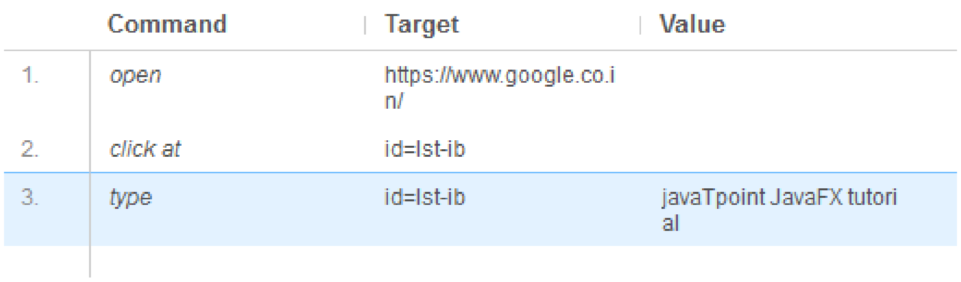


| | Command | Target | Value |
|---|---|---|---|
| 1. | open | https://www.google.co.in/ | |
| 2. | click at | id=lst-ib | |

Note: The "Value" portion of the Test Script Editor box is optional for most of the commands.

We will use the same identification ID for our third command. The third command will type the specified text in to the Google search text box.

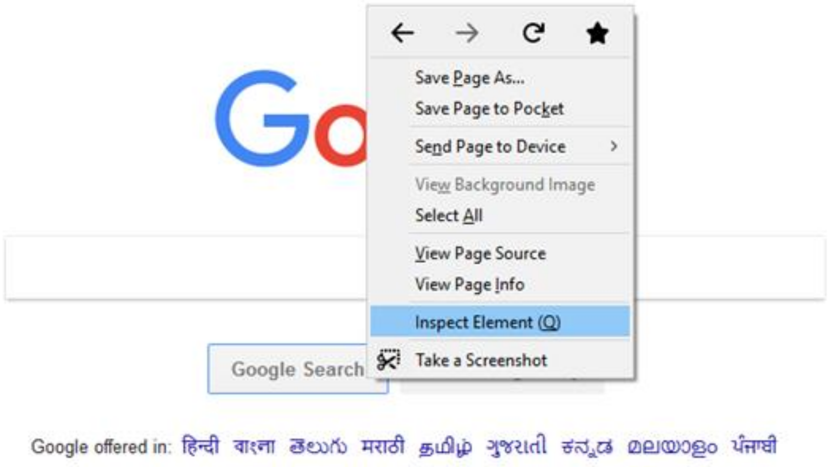
- Modify the properties of third command as:
- Command : type
- Target : id=lst-ib
- Value: javaTpointJavaFX tutorial
- During execution of the test case, this command will type the specified text on the Google search text box.



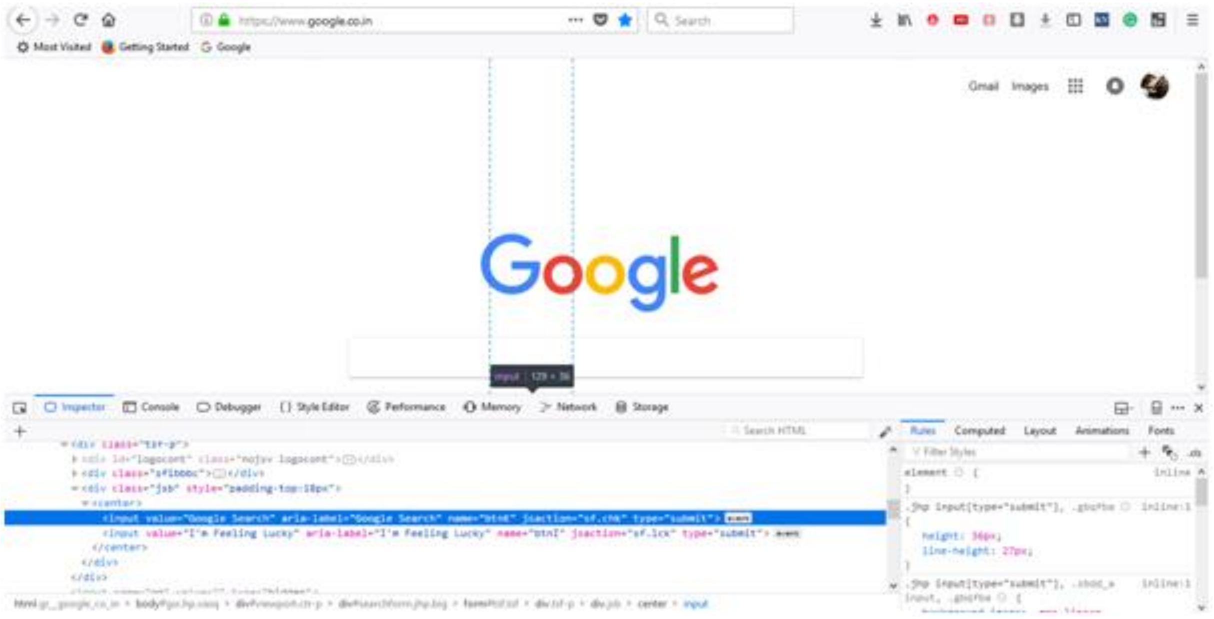
| | Command | Target | Value |
|---|---|---|---|
| 1. | open | https://www.google.co.in/ | |
| 2. | click at | id=lst-ib | |
| 3. | type | id=lst-ib | javaTpoint JavaFX tutorial |

We will now add a command which will generate a button click event on our web page. For this event to be generated, we need a unique identification element for the Google search button.

- Right click on the Google search button and select Inspect Element.

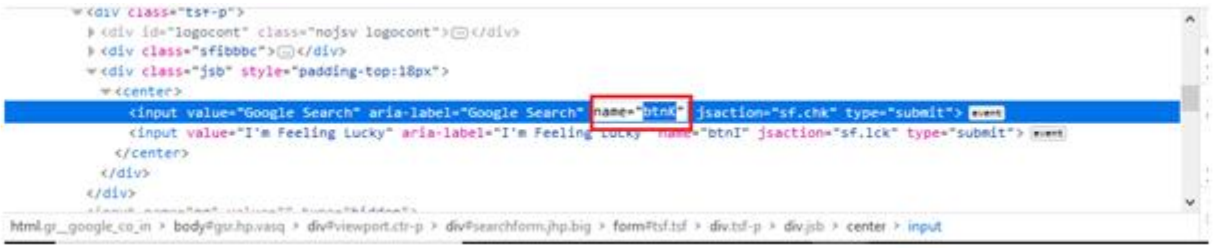


- It will launch a window containing all the specific codes involved in the development of the search button.



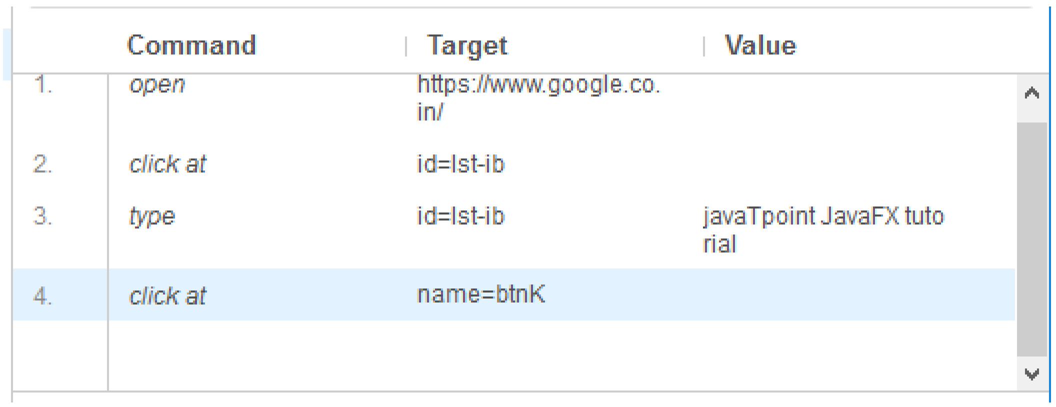
```
▾ <div class="tsf-p">
  ▸ <div id="logocont" class="nojsv logocont">⋯</div>
  ▸ <div class="sfibbbc">⋯</div>
  ▾ <div class="jsb" style="padding-top:18px">
    ▾ <center>
        <input value="Google Search" aria-label="Google Search" name="btnK" jsaction="sf.chk" type="submit"> event
        <input value="I'm Feeling Lucky" aria-label="I'm Feeling Lucky" name="btnI" jsaction="sf.lck" type="submit"> event
      </center>
    </div>
  </div>
html.gr__google_co_in › body#gsr.hp.vasq › div#viewport.ctr-p › div#searchform.jhp.big › form#tsf.tsf › div.tsf-p › div.jsb › center › input
```

- Pick the name element that contains the specified name for the Google search button.



```
▾ <div class="tsf-p">
  ▸ <div id="logocont" class="nojsv logocont">⋯</div>
  ▸ <div class="sfibbbc">⋯</div>
  ▾ <div class="jsb" style="padding-top:18px">
    ▾ <center>
        <input value="Google Search" aria-label="Google Search" name="btnK" jsaction="sf.chk" type="submit"> event
        <input value="I'm Feeling Lucky" aria-label="I'm Feeling Lucky" name="btnI" jsaction="sf.lck" type="submit"> event
      </center>
    </div>
  </div>
html.gr__google_co_in › body#gsr.hp.vasq › div#viewport.ctr-p › div#searchform.jhp.big › form#tsf.tsf › div.tsf-p › div.jsb › center › input
```

- Modify the properties of fourth command as:
- Command : click at
- Target : name=btnK
- During execution of the test case, this command will click on the search button present on the Google search engine web page.

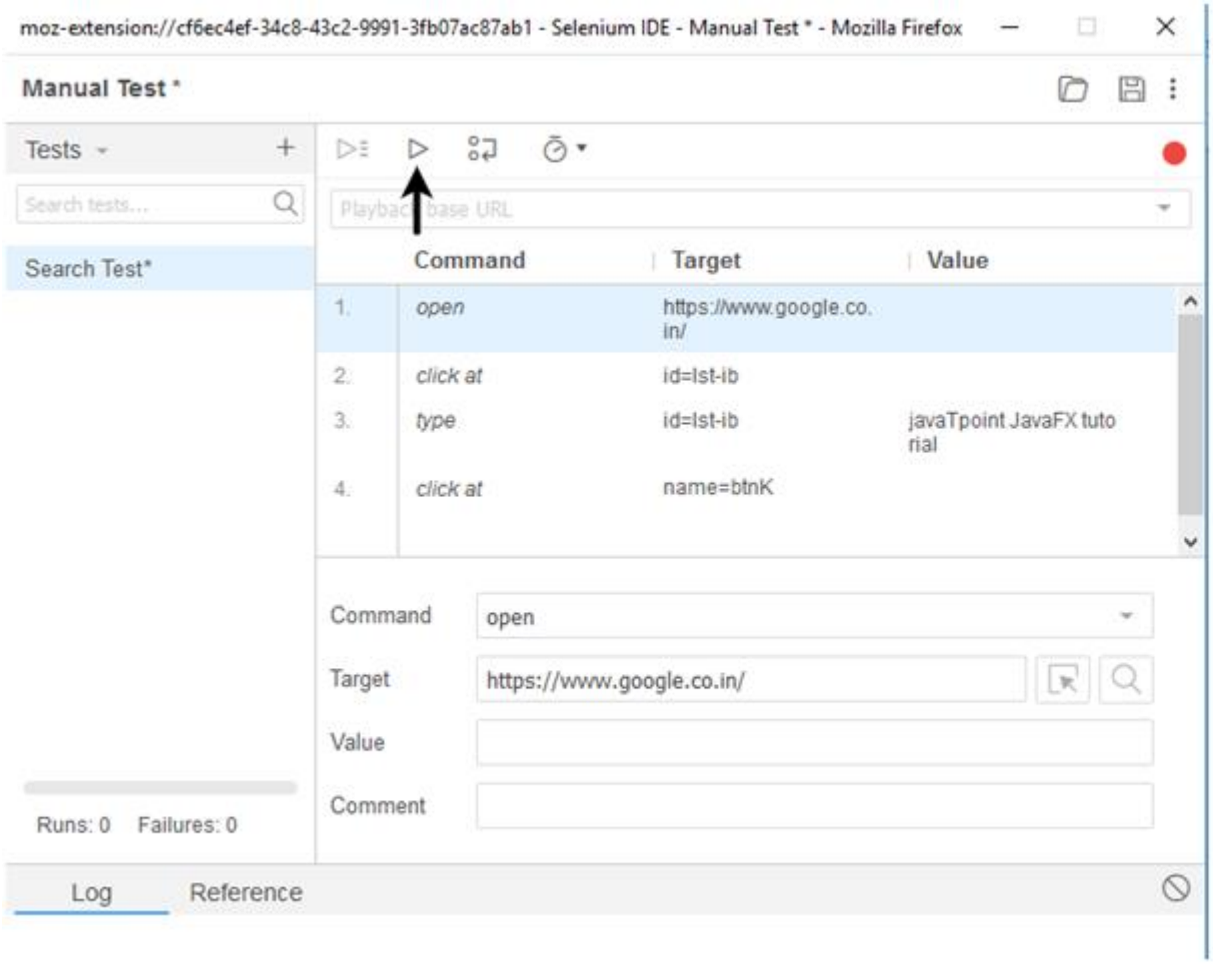


| | Command | Target | Value |
|---|---|---|---|
| 1. | open | https://www.google.co.in/ | |
| 2. | click at | id=lst-ib | |
| 3. | type | id=lst-ib | javaTpoint JavaFX tutorial |
| 4. | click at | name=btnK | |

We are now ready to execute our first test script.

## Executing the Test Script

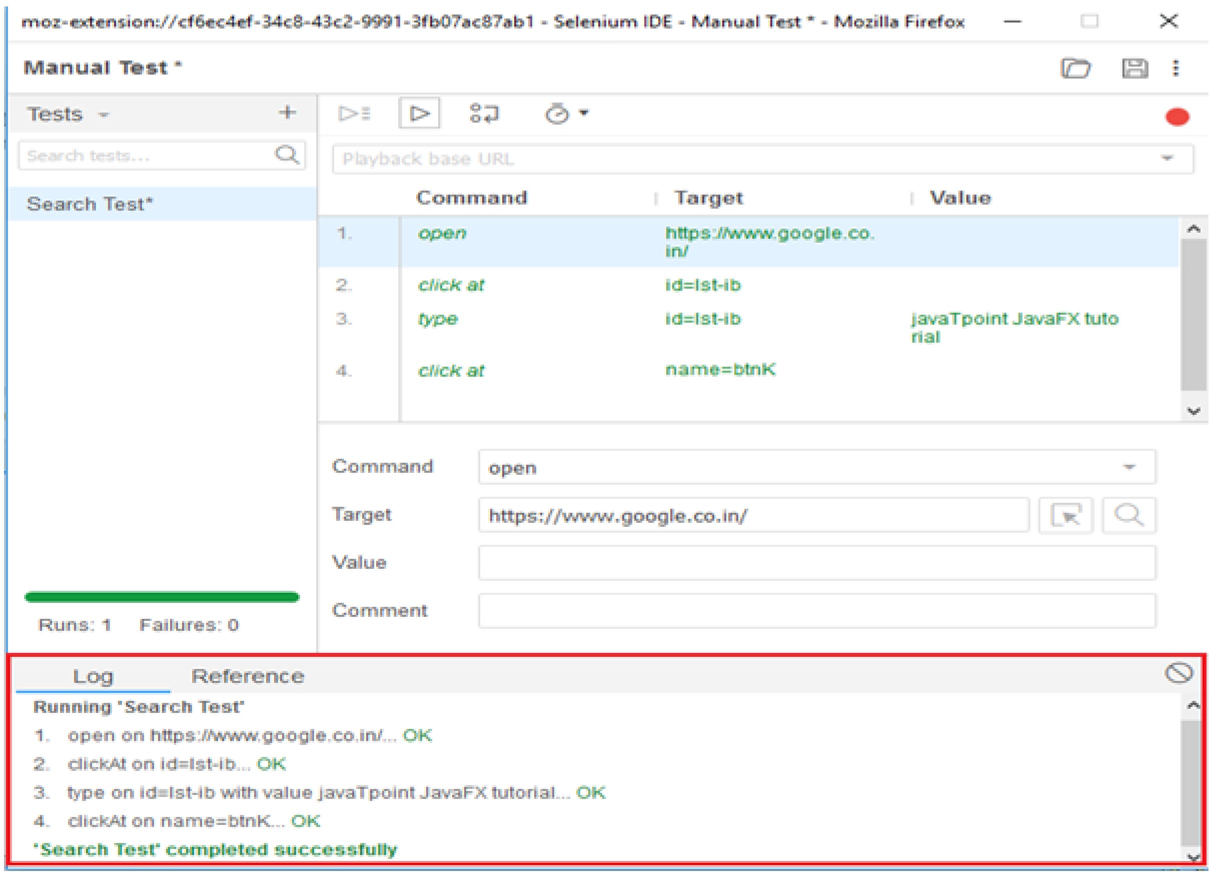
- Click on the "Run Current Test" button present on the tool bar menu of the IDE. It will execute all of your inserted commands on the browser and gives you an overall summary of the executed test script.


moz-extension://cf6ec4ef-34c8-43c2-9991-3fb07ac87ab1 - Selenium IDE - Manual Test * - Mozilla Firefox
Manual Test *
Tests
Search tests...
Search Test*
Playback base URL
Runs: 0 Failures: 0
Log
Reference

| | Command | Target | Value |
|---|---|---|---|
| 1. | open | https://www.google.co.in/ | |
| 2. | click at | id=lst-ib | |
| 3. | type | id=lst-ib | javaTpoint JavaFX tutorial |
| 4. | click at | name=btnK | |

Command: open
Target: https://www.google.co.in/
Value:
Comment:

- The Log pane displays the overall summary of the executed test scripts.


moz-extension://cf6ec4ef-34c8-43c2-9991-3fb07ac87ab1 - Selenium IDE - Manual Test * - Mozilla Firefox
Manual Test *
Tests
Search tests...
Search Test*
Playback base URL
Runs: 1 Failures: 0
Log
Reference

| | Command | Target | Value |
|---|---|---|---|
| 1. | open | https://www.google.co.in/ | |
| 2. | click at | id=lst-ib | |
| 3. | type | id=lst-ib | javaTpoint JavaFX tutorial |
| 4. | click at | name=btnK | |

| | |
|---|---|
| Command | open |
| Target | https://www.google.co.in/ |
| Value | |
| Comment | |

```
Running 'Search Test'
1.  open on https://www.google.co.in/... OK
2.  clickAt on id=lst-ib... OK
3.  type on id=lst-ib with value javaTpoint JavaFX tutorial... OK
4.  clickAt on name=btnK... OK
'Search Test' completed successfully
```

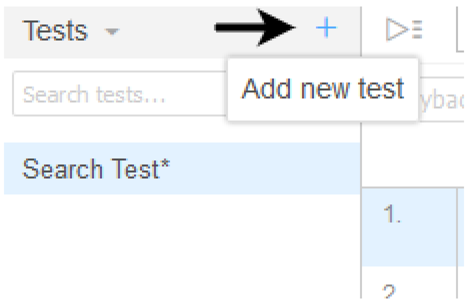
Now, we will create our second test case within the same test suite.

We will generate a test case based on login feature provided by one of the most popular website "Rediffmail".

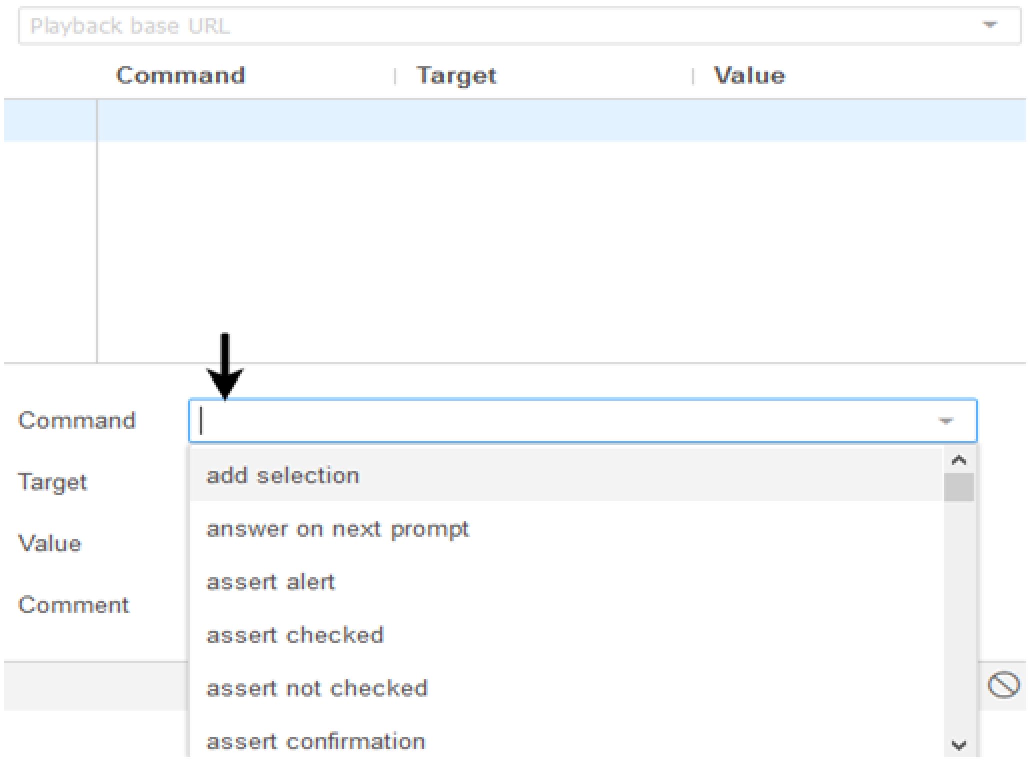
First you have sign-up to get the login credentials. For this test, we have already generated our login credentials.

## 1. Insert Commands

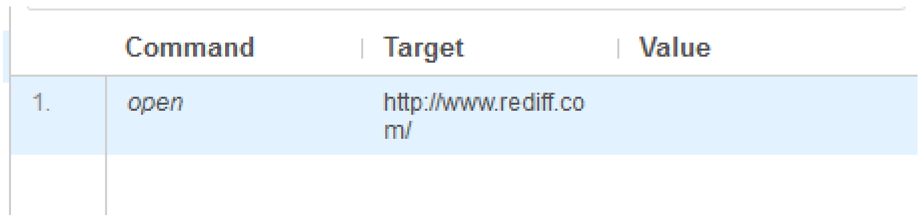
Click on the "Add new test" button at the top of the test case pane.



- Rename the test case as "Login Test".
- Click on the command text box present on the Test Script Editor Box.



- Modify the properties of First command as:
- Command : open
- Target : http://www.rediff.com/
- During execution of the test case, this command will load the Rediff home page on your Firefox browser.

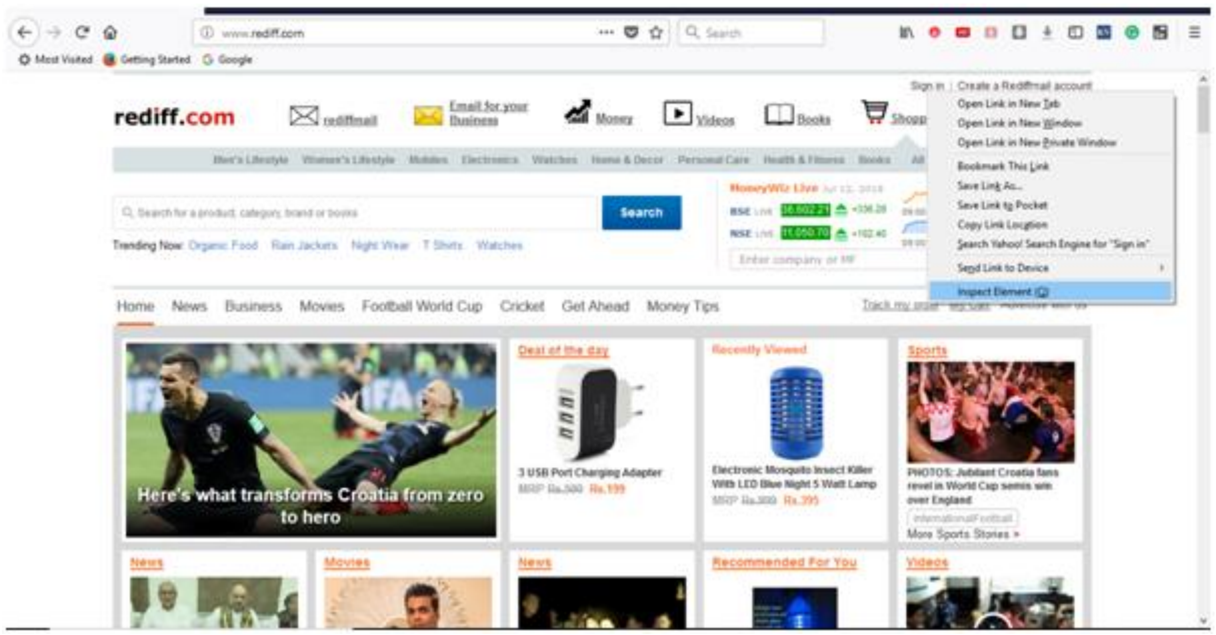


| | Command | Target | Value |
|---|---|---|---|
| 1. | *open* | http://www.rediff.com/ | |

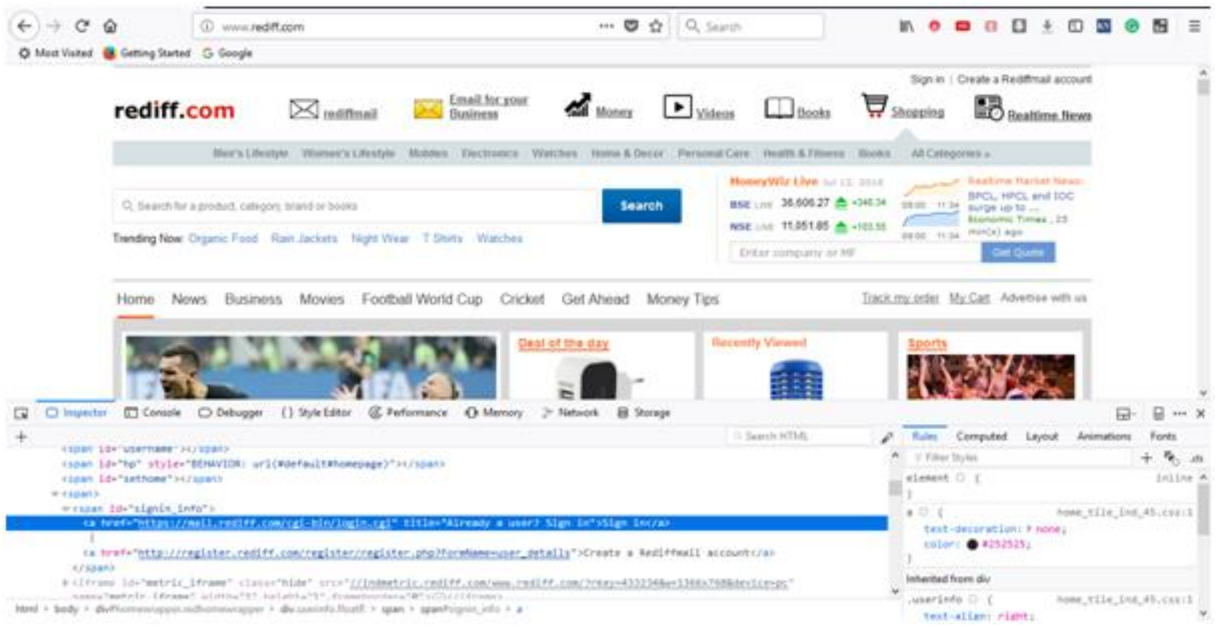
Now, we have to add a command that will click on the "Sign-in" link present at the top right corner of Rediff website. For this, we need a unique identification element of the "Sign-in" link which will help the IDE to identify the target location.

The method for finding a unique identification element involves inspection of HTML codes.

- Open URL:http://www.rediff.com/ on your Firefox browser.
- Right click on the "Sign-in" and select Inspect Element.

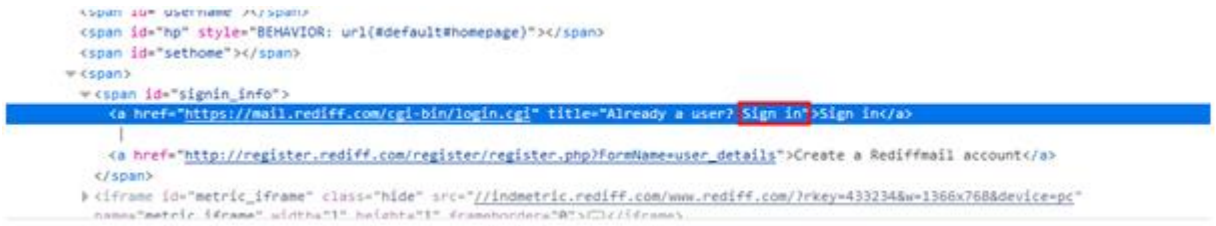


- It will launch a window containing all the specific codes involved in the development of the "Sign-in" link.



```
<span id="username"></span>
<span id="hp" style="BEHAVIOR: url(#default#homepage)"></span>
<span id="sethome"></span>
<span>
<span id="signin_info">
<a href="https://mail.rediff.com/cgi-bin/login.cgi" title="Already a user? Sign in">Sign in</a>
<a href="http://register.rediff.com/register/register.php?FormName=user_details">Create a Rediffmail account</a>
</span>
<iframe id="metric_iframe" class="hide" src="//indmetric.rediff.com/www.rediff.com/?rkey=433234&w=1366x768&device=pc"
```

- Pick the link element that contains the specified name for the "Sign-in" link.



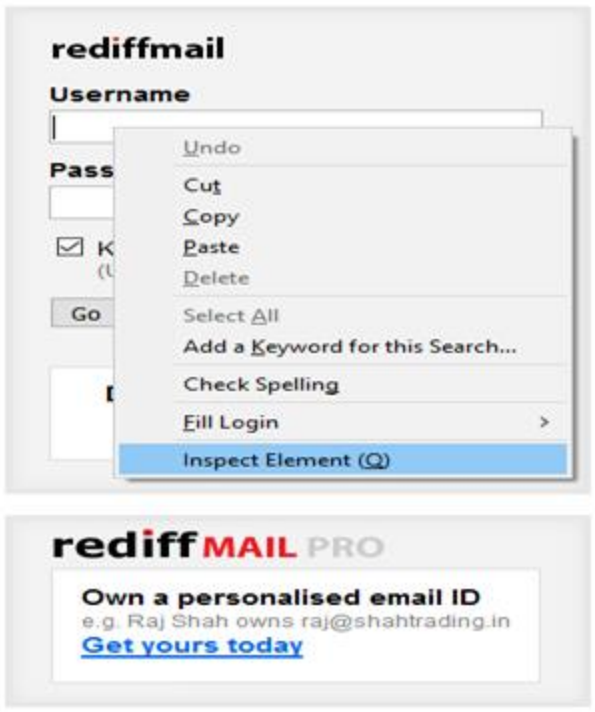
```
<span id="username"></span>
<span id="hp" style="BEHAVIOR: url(#default#homepage)"></span>
<span id="sethome"></span>
▾<span>
  ▾<span id="signin_info">
     <a href="https://mail.rediff.com/cgi-bin/login.cgi" title="Already a user? Sign in">Sign in</a>
     |
     <a href="http://register.rediff.com/register/register.php?FormName=user_details">Create a Rediffmail account</a>
   </span>
 ▸<iframe id="metric_iframe" class="hide" src="//indmetric.rediff.com/www.rediff.com/?rkey=433234&w=1366x768&device=pc"
   name="metric_iframe" width="1" height="1" frameborder="0"></iframe>
```

- Modify the properties of second command as:
- Command : click at
- Target : link=Sign in
- During execution of the test case, this command will click on the "Sign in" link.

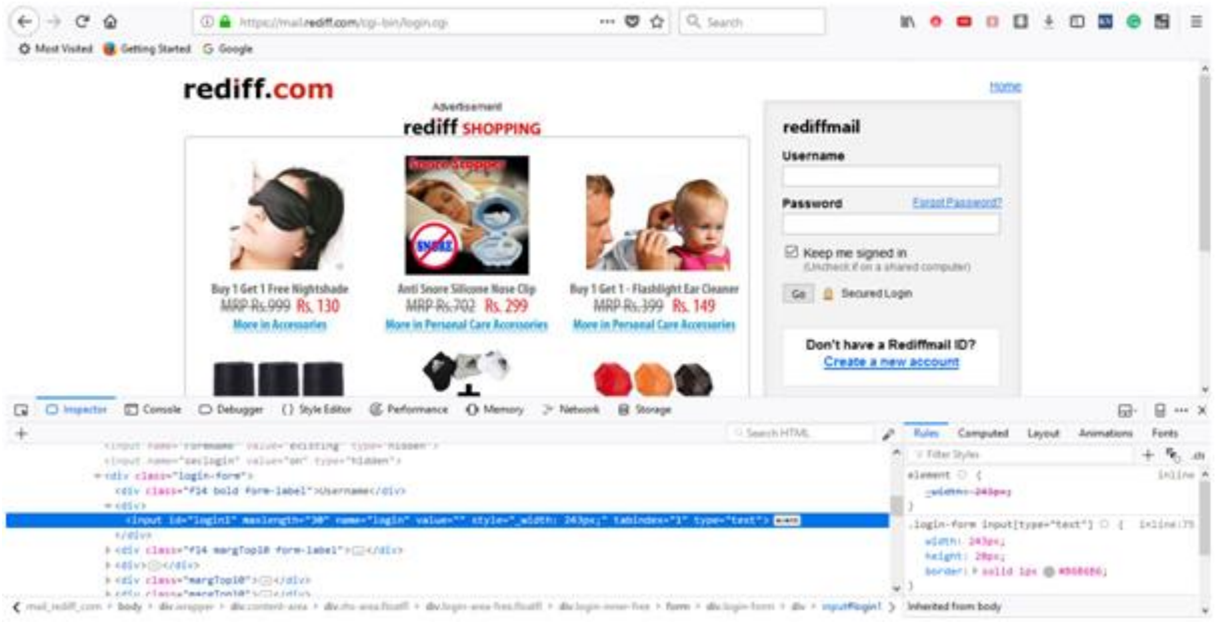
| | Command | Target | Value |
|---|---|---|---|
| 1. | *open* | http://www.rediff.com/ | |
| 2. | *click at* | link=Sign in | |

The "Sign in" link will redirect you to the login page. Therefore, for the third command to be entered we need a unique identification for the "Username" text box which will help the IDE to identify the target location.

- Right click on the "Username" text box and select Inspect Element.

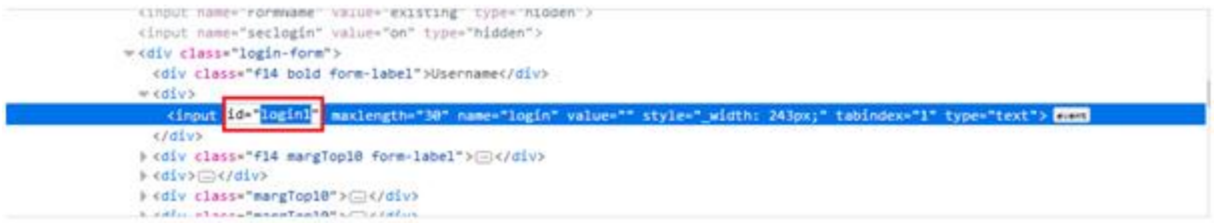


- It will launch a window containing all the specific codes involved in the development of the "Username" text box.



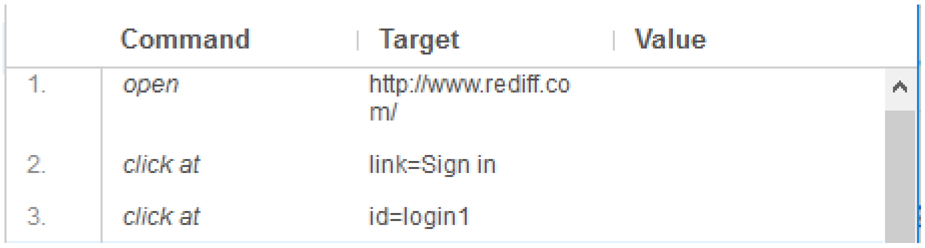
```
<input name="fromname" value="existing" type="hidden">
<input name="seclogin" value="on" type="hidden">
<div class="login-form">
  <div class="f14 bold form-label">Username</div>
  <div>
    <input id="login1" maxlength="30" name="login" value="" style="_width: 263px;" tabindex="1" type="text">
  </div>
  <div class="f14 margTop10 form-label"></div>
  <div></div>
  <div class="margTop10"></div>
```

- 
- Pick the ID element that contains the specific ID for the "Username" text box.



```
<input name="seclogin" value="on" type="hidden">
▾<div class="login-form">
   <div class="f14 bold form-label">Username</div>
  ▾<div>
     <input id="login1" maxlength="30" name="login" value="" style="_width: 243px;" tabindex="1" type="text">
   </div>
  ▸<div class="f14 margTop10 form-label">…</div>
  ▸<div>…</div>
  ▸<div class="margTop10">…</div>
```

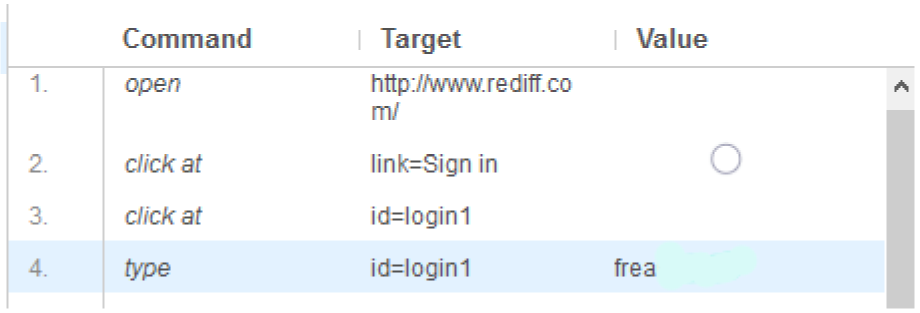
- Modify the properties of third command as:
- Command : click at
- Target : id=login1
- During execution of the test case, this command will click on the "Username" text box.



| | Command | Target | Value |
|---|---|---|---|
| 1. | open | http://www.rediff.com/ | |
| 2. | click at | link=Sign in | |
| 3. | click at | id=login1 | |

We will use the same ID element for our next command which involves typing the user-id as login credential.

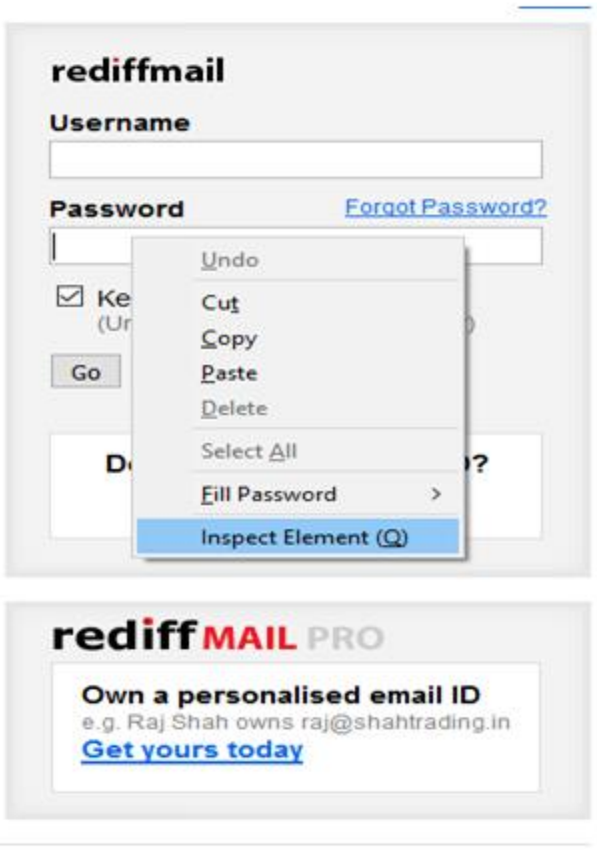
- Modify the properties of fourth command as:
- Command : type.
- Target : id=login1
- Value : frea********* (User Login ID)
- During execution of the test case, this command will type the user id in the "Usename" text box.



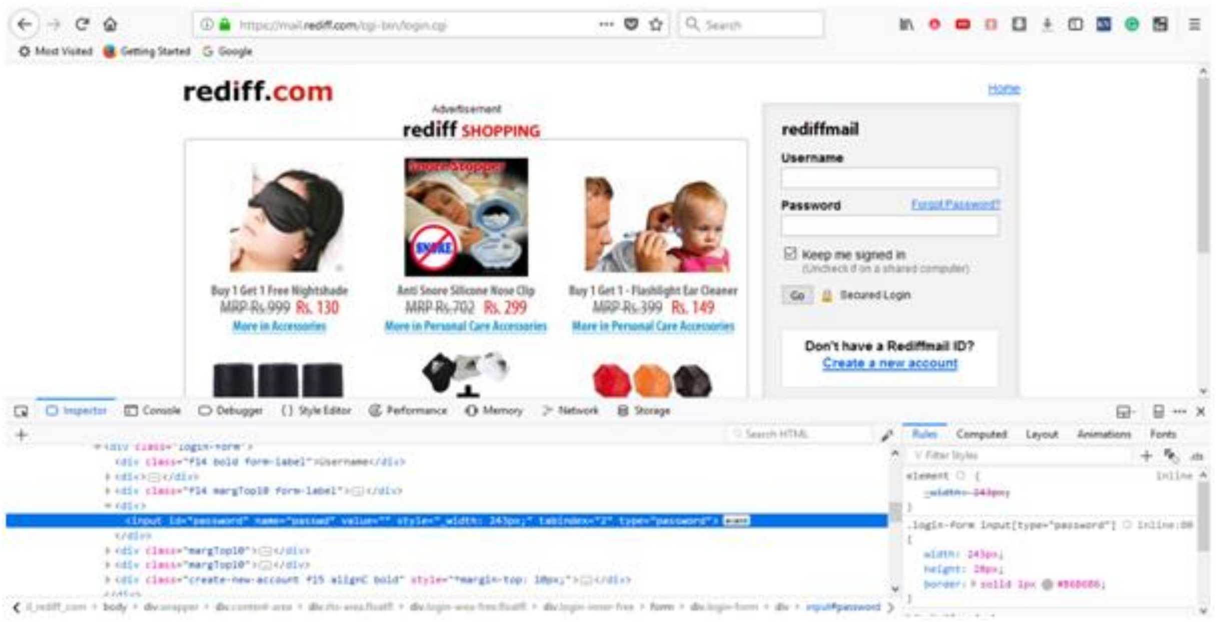
| | Command | Target | Value |
|---|---|---|---|
| 1. | open | http://www.rediff.com/ | |
| 2. | click at | link=Sign in | |
| 3. | click at | id=login1 | |
| 4. | type | id=login1 | frea |

We will now add a command that will click on the "Password" text box. For this command, we need a unique identification for the "Password" field which will help the IDE to identify the target location.

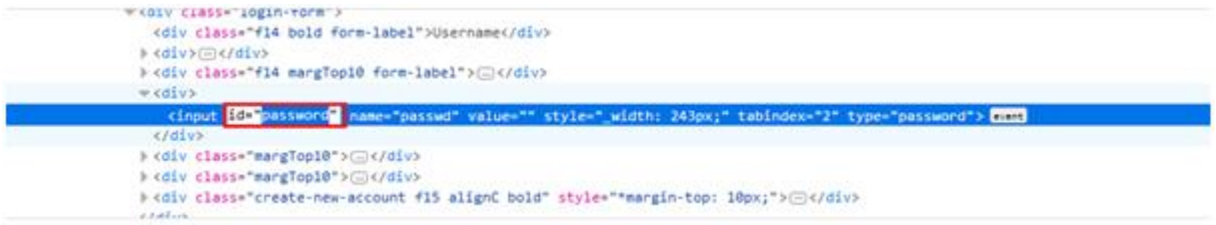
- Right click on the "Password" text box and select Inspect Element.



- It will launch a window containing all the specific codes involved in the development of the "Password" text box.

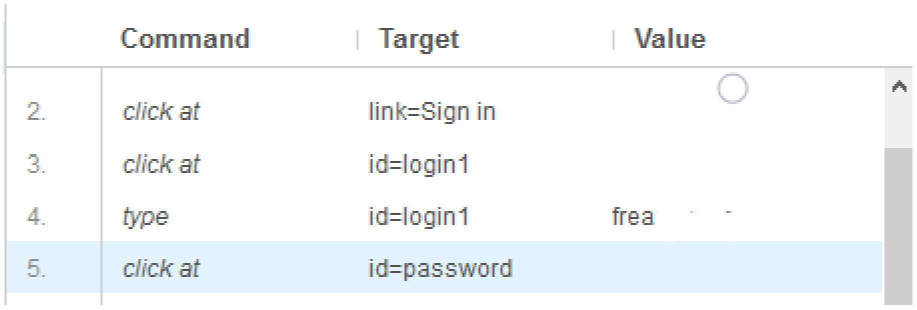


- Pick the ID element that contains the specific ID for the "Password" text box.



```
▾<div class="login-form">
   <div class="f14 bold form-label">Username</div>
  ▸<div>⊡</div>
  ▸<div class="f14 margTop10 form-label">⊡</div>
  ▾<div>
     <input id="password" name="passwd" value="" style="_width: 243px;" tabindex="2" type="password"> event
   </div>
  ▸<div class="margTop10">⊡</div>
  ▸<div class="margTop10">⊡</div>
  ▸<div class="create-new-account f15 alignC bold" style="*margin-top: 10px;">⊡</div>
```

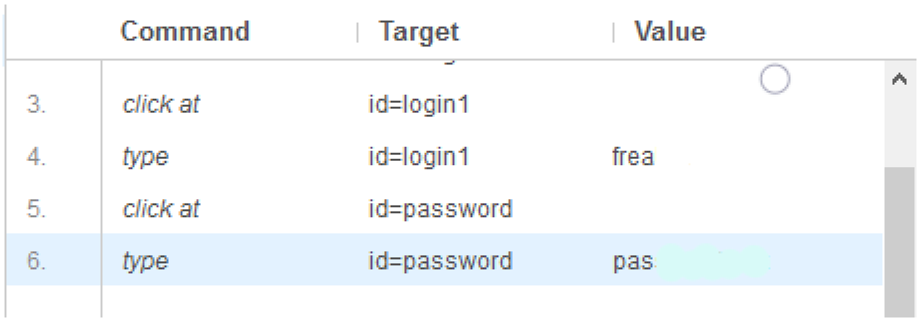
- Modify the properties of fifth command as:
- Command : click at
- Target : id=password
- During execution of the test case, this command will click on the "Username" text box.



| | Command | Target | Value |
|---|---|---|---|
| 2. | *click at* | link=Sign in | |
| 3. | *click at* | id=login1 | |
| 4. | *type* | id=login1 | frea |
| 5. | *click at* | id=password | |

We will use the same ID element for our next command which involves typing the password as login credential.

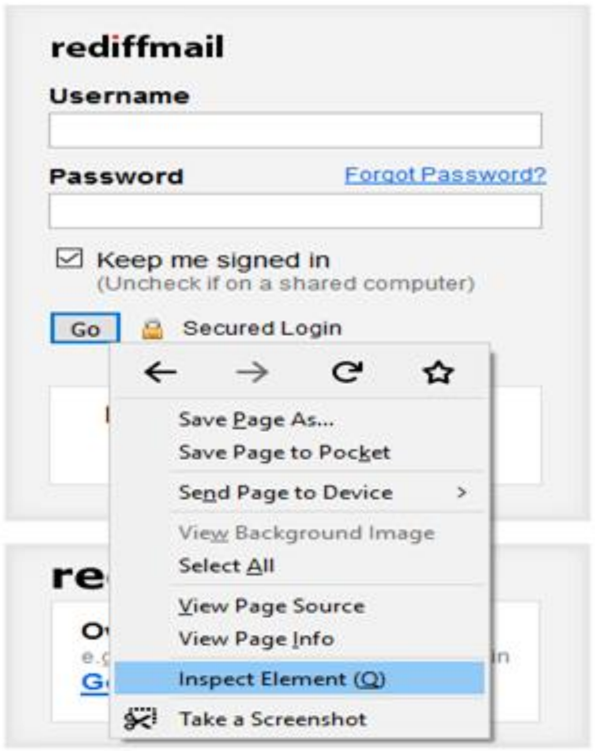
- Modify the properties of sixth command as:
- Command : type
- arget : id=password
- Value : **********(User Login Password)
- During execution of the test case, this command will type the user login password in the "Password" text box.



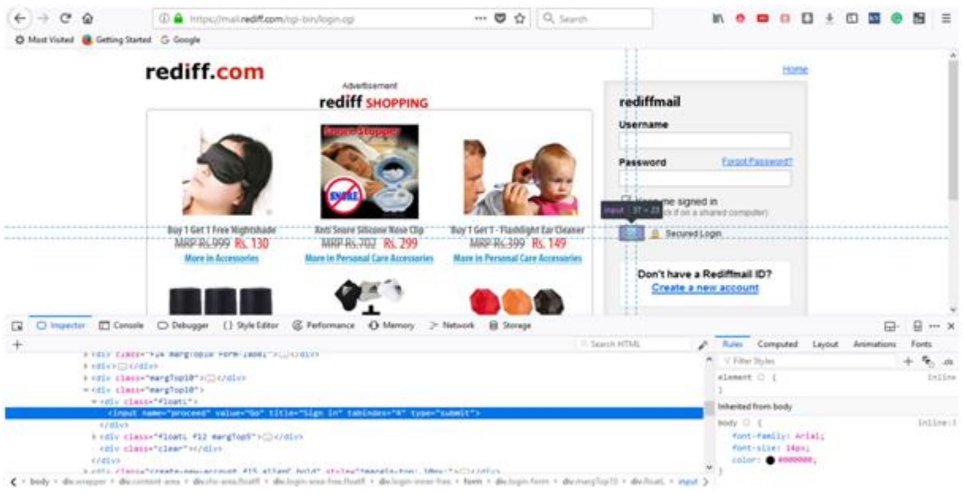
|  | Command | Target | Value |
|---|---|---|---|
| 3. | click at | id=login1 |  |
| 4. | type | id=login1 | frea |
| 5. | click at | id=password |  |
| 6. | type | id=password | pas |

At last, we need a unique identification element for the Login Submit button which will help the IDE to identify the target location.

- Right click on the "Go" button and select Inspect Element.

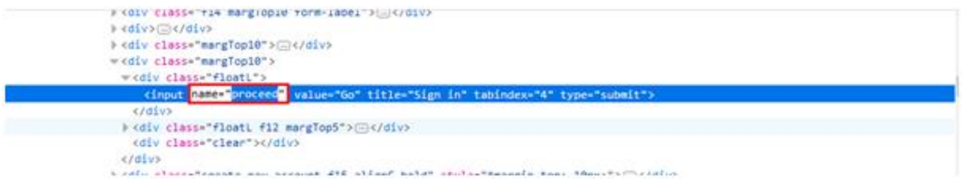


- It will launch a window containing all the specific codes involved in the development of the "Go" submit button.



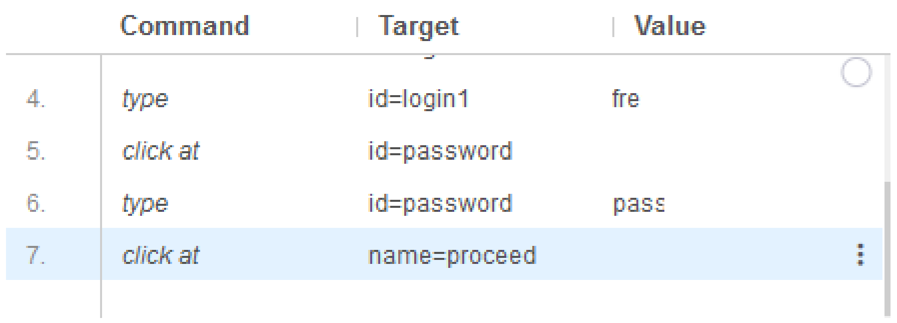
```
<div>⋯</div>
<div class="margTop10">⋯</div>
<div class="margTop10">
  <div class="floatL">
    <input name="proceed" value="Go" title="Sign in" tabindex="4" type="submit">
  </div>
  <div class="floatL f12 margTop5">⋯</div>
  <div class="clear"></div>
</div>
```

- Pick the name element that contains the specific name for the "Go" submit button.



```
▸ <div class="f14 margTop10 form-label">⋯</div>
▸ <div>⋯</div>
▸ <div class="margTop10">⋯</div>
▾ <div class="margTop10">
  ▾ <div class="floatL">
      <input name="proceed" value="Go" title="Sign in" tabindex="4" type="submit">
    </div>
  ▸ <div class="floatL f12 margTop5">⋯</div>
    <div class="clear"></div>
  </div>
```

- Modify the properties of seventh command as:
- Command : click at
- Target : name=proceed
- During execution of the test case, this command will click on the "Go" submit button.

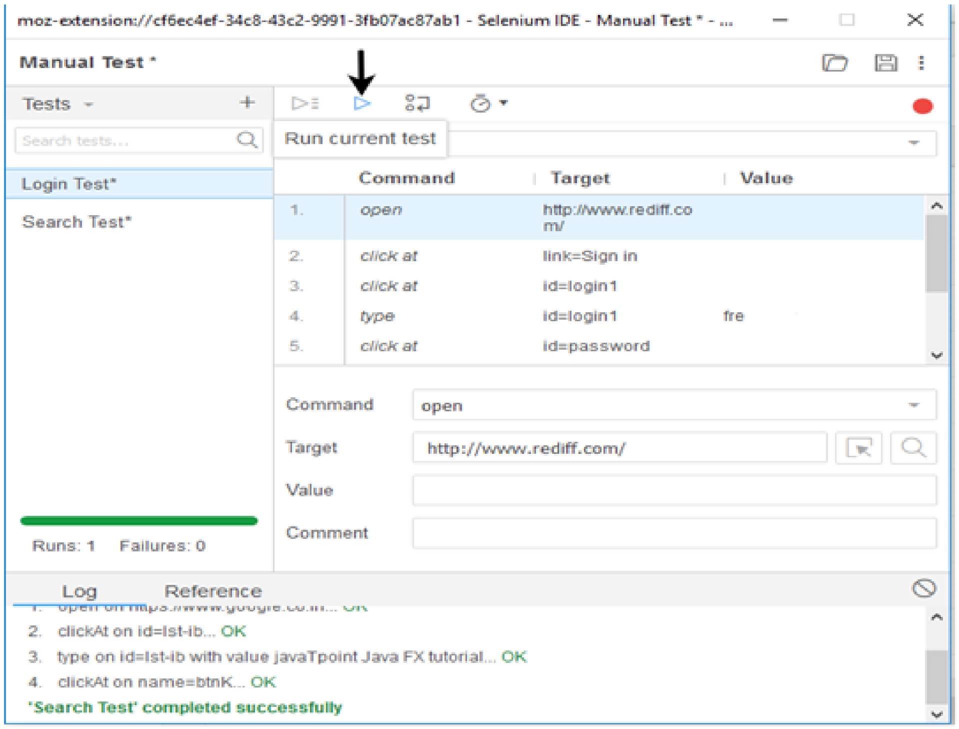


| | Command | Target | Value |
|---|---|---|---|
| 4. | type | id=login1 | fre |
| 5. | click at | id=password | |
| 6. | type | id=password | pass |
| 7. | click at | name=proceed | |

We are now ready to execute our second test script.

## Executing the Test Script

- Click on the "Run Current Test" button present on the tool bar menu of the IDE. It will execute all of your inserted commands on the browser and gives you an overall summary of the executed test script.

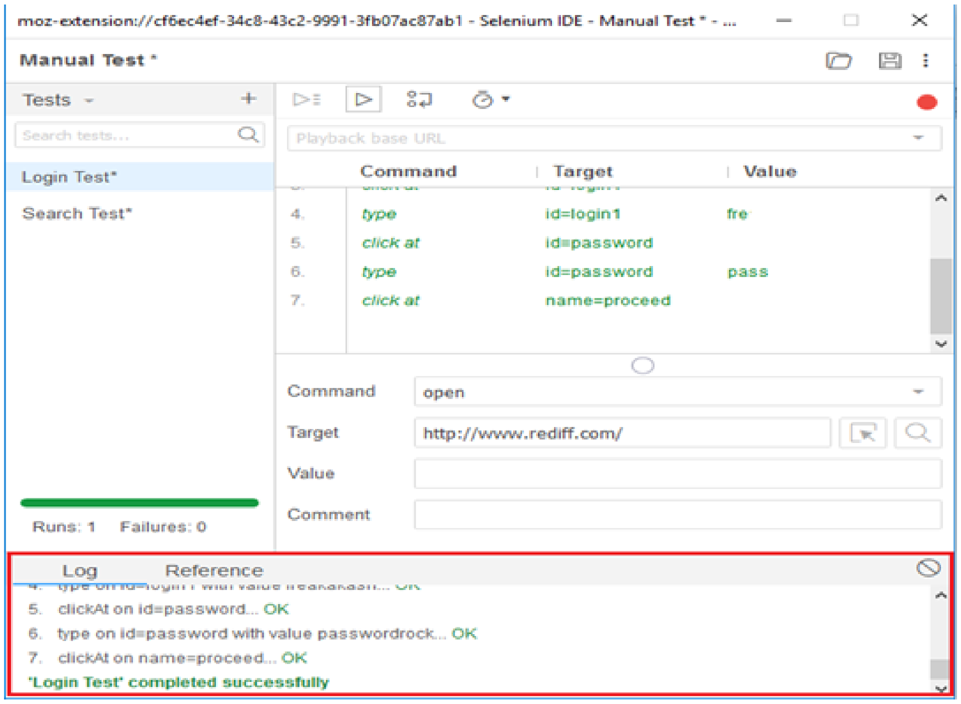


| Command | Target | Value |
|---|---|---|
| 1. open | http://www.rediff.com/ | |
| 2. click at | link=Sign in | |
| 3. click at | id=login1 | |
| 4. type | id=login1 | fre |
| 5. click at | id=password | |

Command: open
Target: http://www.rediff.com/
Value:
Comment:

```
1. open on https://www.google.co.in... OK
2. clickAt on id=lst-ib... OK
3. type on id=lst-ib with value javaTpoint Java FX tutorial... OK
4. clickAt on name=btnK... OK
'Search Test' completed successfully
```

- The Log pane displays the overall summary of the executed test scripts.



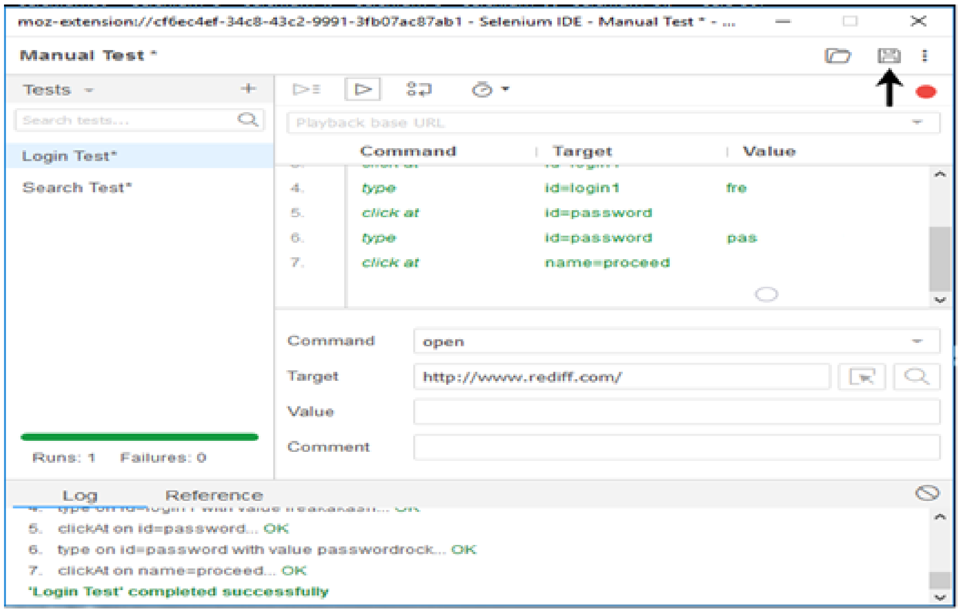
| | Command | Target | Value |
|---|---|---|---|
| 4. | type | id=login1 | fre |
| 5. | click at | id=password | |
| 6. | type | id=password | pass |
| 7. | click at | name=proceed | |

| | |
|---|---|
| Command | open |
| Target | http://www.rediff.com/ |
| Value | |
| Comment | |

```
5.  clickAt on id=password... OK
6.  type on id=password with value passwordrock... OK
7.  clickAt on name=proceed... OK
'Login Test' completed successfully
```

## Saving the test suite

- Click on the save button present on the extreme right corner of the menu bar.

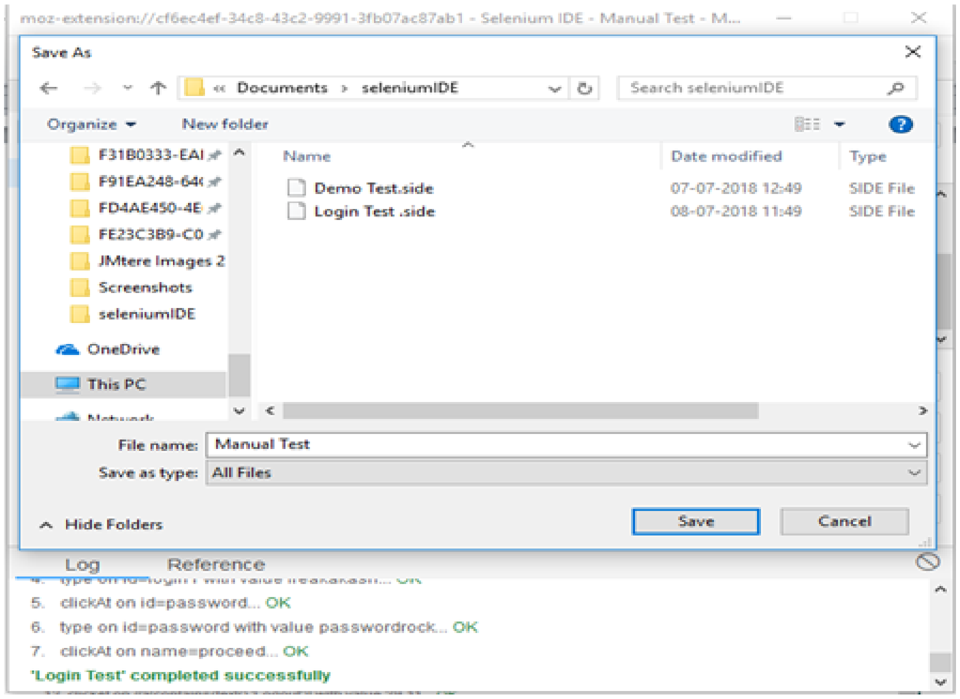


| | Command | Target | Value |
|---|---|---|---|
| 4. | type | id=login1 | fre |
| 5. | click at | id=password | |
| 6. | type | id=password | pas |
| 7. | click at | name=proceed | |

| | |
|---|---|
| Command | open |
| Target | http://www.rediff.com/ |
| Value | |
| Comment | |

```
5.  clickAt on id=password... OK
6.  type on id=password with value passwordrock... OK
7.  clickAt on name=proceed... OK
'Login Test' completed successfully
```

- Save the entire test suite as "Manual Test".



| Name | Date modified | Type |
|---|---|---|
| Demo Test.side | 07-07-2018 12:49 | SIDE File |
| Login Test .side | 08-07-2018 11:49 | SIDE File |

File name: Manual Test
Save as type: All Files

```
5.  clickAt on id=password... OK
6.  type on id=password with value passwordrock... OK
7.  clickAt on name=proceed... OK
'Login Test' completed successfully
```

- The test suite can be found at the location provided in the above steps.