# **Selenium WebDriver - Browser Commands**

The very basic browser operations of WebDriver include **opening a browser**; perform few tasks and then **closing the browser**.

Given are some of the most commonly used Browser commands for Selenium WebDriver.

### **1. Get Command**

**Method:**

Volume 0%

1. get(String arg0) : **void**

In WebDriver, this method loads a new web page in the existing browser window. It accepts String as parameter and returns void.

**The respective command to load a new web page can be written as:**

1. driver.get(URL);
3. // Or can be written as
5. String URL = "URL";
6. driver.get(URL);

**Example:** For instance, the command to load the official website of javaTpointcan be written as:

1. driver.get("www.javatpoint.com")

### **2. Get Title Command**

**Method:**

1. getTitle(): String

In WebDriver, this method fetches the title of the current web page. It accepts no parameter and returns a String.

The respective command to fetch the title of the current page can be written as:

1. driver.getTitle();
3. // Or can be written as
5. String Title = driver.getTitle();

### **3. Get Current URL Command**

**Method:**

1. getCurrentUrl(): String

In WebDriver, this method fetches the string representing the Current URL of the current web page. It accepts nothing as parameter and returns a String value.

The respective command to fetch the string representing the current URL can be written as:

1. driver.getCurrentUrl();
3. //Or can be written as
5. String CurrentUrl = driver.getCurrentUrl();

### **4. Get Page Source Command**

**Method:**

1. getPageSource(): String

In WebDriver, this method returns the source code of the current web page loaded on the current browser. It accepts nothing as parameter and returns a String value.

The respective command to get the source code of the current web page can be written as:

1. driver.getPageSource();
3. //Or can be written as
4. String PageSource = driver.getPageSource();

### **5. Close Command**

**Method:**

1. close(): **void**

This method terminates the current browser window operating by WebDriver at the current time. If the current window is the only window operating by WebDriver, it terminates the browser as well. This method accepts nothing as parameter and returns void.

The respective command to terminate the browser window can be written as:

1. driver.close();

### **6. Quit Command**

**Method:**

1. quit(): **void**

This method terminates all windows operating by WebDriver. It terminates all tabs as well as the browser itself. It accepts nothing as parameter and returns void.

The respective command to terminate all windows can be written as:

1. driver.quit();

Let us consider a sample test script in which will cover most of the Browser Commands provided by WebDriver.

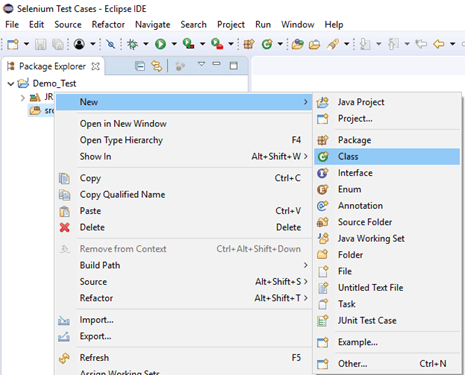
In this sample test, we will automate the following test scenarios:

* Invoke Chrome Browser
* Open URL: <https://www.google.co.in/>
* Get Page Title name and Title length
* Print Page Title and Title length on the Eclipse Console
* Get page URL and verify whether it is the desired page or not
* Get page Source and Page Source length
* Print page Length on Eclipse Console.
* Close the Browser

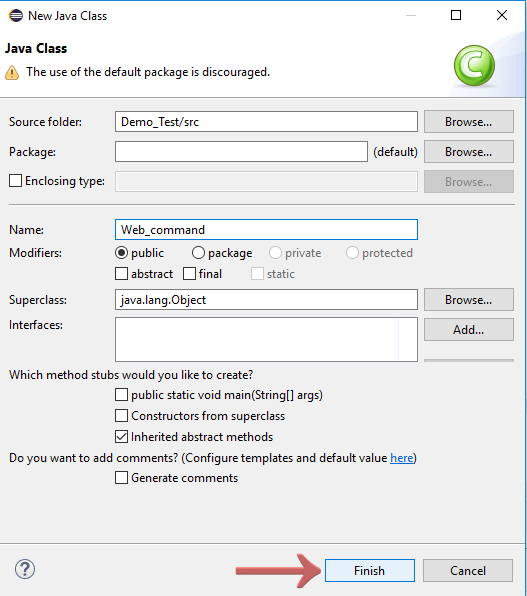
For our test purpose, we are using the home page of "Google" search engine.

We will create our test case step by step to give you a complete understanding on how to use Browser Commands in WebDriver.

* **Step1.** Launch Eclipse IDE and open the existing test suite "Demo\_Test" which we have created in [WebDriver Installation](https://www.javatpoint.com/selenium-webdriver-installation) section of WebDriver tutorial.
* **Step2.** Right click on the "src" folder and create a new Class File from **New > Class**.



Give your Class name as "Navigation\_command" and click on "Finish" button.



**Step3.** Let's get to the coding ground.

To automate our test scenarios, first you need to know "How to invoke/launch web browsers in WebDriver?"

#### Note: To invoke a browser in Selenium, we have to download an executable file specific to that browser. For example, Chrome browser implements the WebDriver protocol using an executable called ChromeDriver.exe. These executable files start a server on your system which in turn is responsible for running your test scripts in Selenium.

We have stated the procedures and methods to run our tests on different browser in later sections of this tutorial. For reference you can go through each one of them before proceeding with the actual coding.

1. [Running test on Firefox](https://www.javatpoint.com/selenium-webdriver-running-test-on-firefox-browser-gecko-driver)
2. [Running test on Chrome](https://www.javatpoint.com/selenium-webdriver-running-test-on-chrome-browser)
3. [Running test on Internet Explorer](https://www.javatpoint.com/selenium-webdriver-running-test-on-ie-browser)
4. [Running test on Safari](https://www.javatpoint.com/selenium-webdriver-running-test-on-safari-browser)

* To invoke Google Chrome browser, we need to download the ChromeDriver.exe file and set the system property to the path of your ChromeDriver.exe file. We have already discussed this in earlier sessions of this tutorial. You can also refer to "[Running test on Chrome Browser](https://www.javatpoint.com/selenium-webdriver-running-test-on-chrome-browser)" to learn how to download and set System property for Chrome driver.

Here is the sample code to set system property for Chrome driver:

1. // System Property for Chrome Driver
2. System.setProperty("webdriver.chrome.driver","D:\\ChromeDriver\\chromedriver.exe");

After that we have to initialize Chrome driver using ChromeDriver class.

Here is the sample code to initialize Chrome driver using ChromeDriver class:

1. // Instantiate a ChromeDriver class.
2. WebDriver driver=**new** ChromeDriver();

Combining both of the above code blocks, we will get the code snippet to launch Google Chrome browser.

1. // System Property for Chrome Driver
2. System.setProperty("webdriver.chrome.driver","D:\\ChromeDriver\\chromedriver.exe");
4. // Instantiate a ChromeDriver class.
5. WebDriver driver=**new** ChromeDriver();

* To automate our second test scenario i.e. "Get Page Title name and Title length", we have to store the title name and length in string and int variable respectively.

Here is the sample code to do that:

1. // Storing Title name in the String variable
2. String title = driver.getTitle();
4. // Storing Title length in the Int variable
5. **int** titleLength = driver.getTitle().length();

To print page Title name and Title length in the Console window, follow the given code snippet:

1. // Printing Title & Title length in the Console window
2. System.out.println("Title of the page is : " + title);
3. System.out.println("Length of the title is : "+ titleLength);

* The next test scenario requires fetching the URL and verifying it against the actual URL.

First, we will store the current URL in a String variable:

1. // Storing URL in String variable
2. String actualUrl = driver.getCurrentUrl();

Verifying the current URL as the actual URL:

1. **if** (actualUrl.equals("https://www.google.co.in"))
2. {
3. System.out.println("Verification Successful - The correct Url is opened.");
4. }
5. Else
6. {
7. System.out.println("Verification Failed - An incorrect Url is opened.");
8. }

* To automate our 6th test scenario (Get page Source and Page Source length), we will store the page source and page source length in the string and int variable respectively.

1. // Storing Page Source in String variable
2. String pageSource = driver.getPageSource();
4. // Storing Page Source length in Int variable
5. **int** pageSourceLength = pageSource.length();

To print the length of the Page source on console window, follow the given code snippet:

1. // Printing length of the Page Source on console
2. System.out.println("Total length of the Page Source is : " + pageSourceLength);

* Finally, the given code snippet will terminate the process and close the browser.

1. driver.close();

Combining all of the above code blocks together, we will get the required source code to execute our test script "Web\_command".

The final test script will appear something like this:

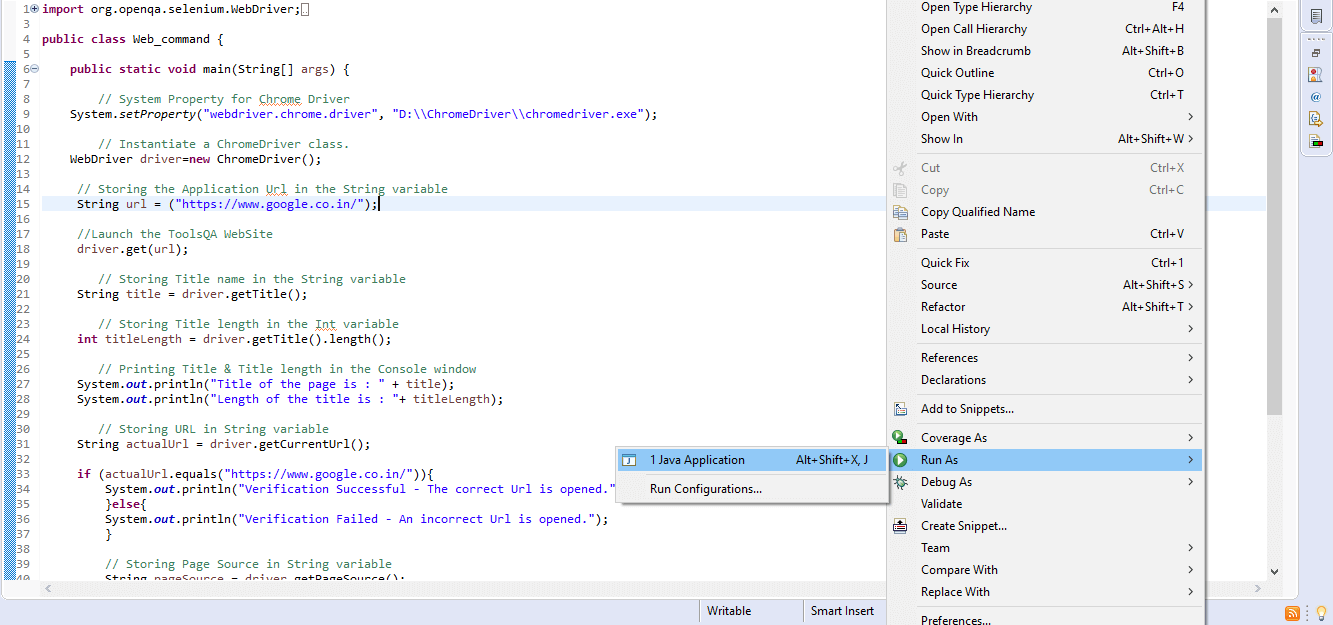
(We have embedded comment in each section to explain the steps clearly)

1. **import** org.openqa.selenium.WebDriver;
2. **import** org.openqa.selenium.chrome.ChromeDriver;
4. **public** **class** Web\_command {
6. **public** **static** **void** main(String[] args) {
8. // System Property for Chrome Driver
9. System.setProperty("webdriver.chrome.driver","D:\\ChromeDriver\\chromedriver.exe");
11. // Instantiate a ChromeDriver class.
12. WebDriver driver=**new** ChromeDriver();
14. // Storing the Application Url in the String variable
15. String url = ("https://www.google.co.in/");
17. //Launch the ToolsQA WebSite
18. driver.get(url);
20. // Storing Title name in the String variable
21. String title = driver.getTitle();
23. // Storing Title length in the Int variable
24. **int** titleLength = driver.getTitle().length();
26. // Printing Title & Title length in the Console window
27. System.out.println("Title of the page is : " + title);
28. System.out.println("Length of the title is : "+ titleLength);
30. // Storing URL in String variable
31. String actualUrl = driver.getCurrentUrl();
33. **if** (actualUrl.equals("https://www.google.co.in/")){
34. System.out.println("Verification Successful - The correct Url is opened.");
35. }
36. **else**{
38. System.out.println("Verification Failed - An incorrect Url is opened.");
39. }
41. // Storing Page Source in String variable
42. String pageSource = driver.getPageSource();
44. // Storing Page Source length in Int variable
45. **int** pageSourceLength = pageSource.length();
47. // Printing length of the Page Source on console
48. System.out.println("Total length of the Pgae Source is : " + pageSourceLength);

51. //Closing browser
52. driver.close();
53. }
54. }

To run the test script on Eclipse window, right click on the screen and click

**Run as → Java application**



After execution, the test script will launch the chrome browser and automate all the test scenarios. The console window will show the results for print commands.

