# **Selenium WebDriver- First Test Case**

In this section, you will learn how to create your First Selenium Automation Test Script.

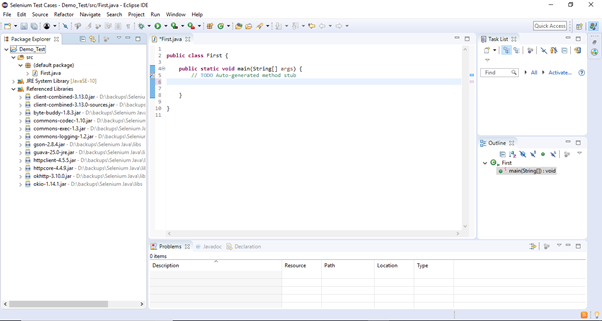
Under this test, we will automate the following scenarios:

* Invoke Google Chrome browser.
* Open URL: [www.google.com](https://www.google.com/)
* Click on the Google Search text box.
* Type the value "javatpoint tutorials"
* Click on the Search button.

We will create our test case step by step to give you a complete understanding of each component in detail.

Volume 0%

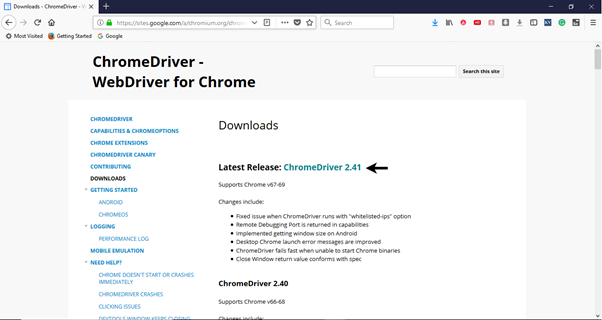
**Step1**. Launch Eclipse IDE and open project "Demo\_Test" which we have created in the previous section (Configure Selenium WebDriver) of this Tutorial. We will write our first Selenium test script in the "First.class" file under the "Demo\_Test" test suite.



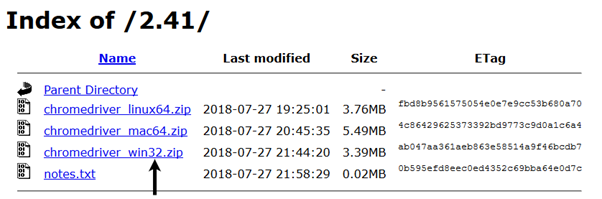
#### Note: To invoke a browser in Selenium, we have to download an executable file specific to that browser. For example, Chrome browser implements the WebDriver protocol using an executable called ChromeDriver.exe. These executable files start a server on your system which in turn is responsible for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

Step2. Open URL: <https://sites.google.com/a/chromium.org/chromedriver/downloads> in your browser.

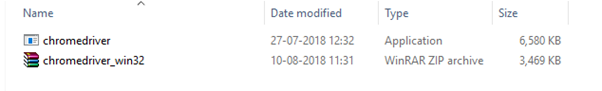
**Step3.** Click on the "ChromeDriver 2.41" link. It will redirect you to the directory of ChromeDriver executables files. Download as per the operating system you are currently on.



For windows, click on the "chromedriver\_win32.zip" download.



The downloaded file would be in zipped format. Unpack the contents in a convenient directory.



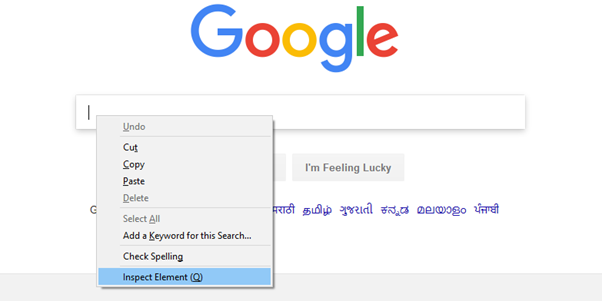
#### Note: Selenium developers have defined properties for each browser that needs the location of the respective executable files to be parsed in order to invoke a browser. For example, the property defined for Chrome browser - webdriver.chrome.driver, needs the path of its executable file - D:\ChromeDriver\chromedriver.exe in order to launch chrome browser.

Selenium WebDriver First Test Case

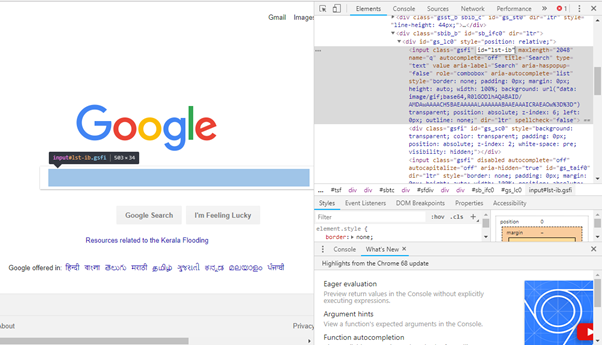
**Step4**. We would need a unique identification for the web elements like Google Search text box and Search button in order to automate them through our test script. These unique identifications are configured along with some Commands/Syntax to form Locators. Locators help us to locate and identify a particular web element in context of a web application.

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

* Open URL: [https://www.google.com](https://www.google.com/) in your Chrome browser.
* Right click on the Google search text box and select Inspect Element.



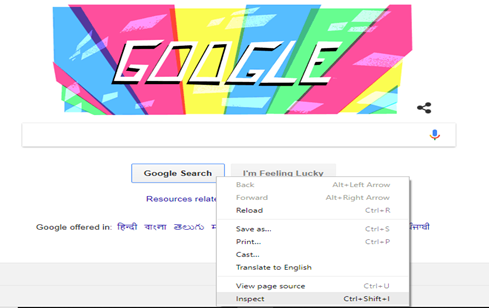
* It will launch a window containing all the specific codes involved in the development of the test box.



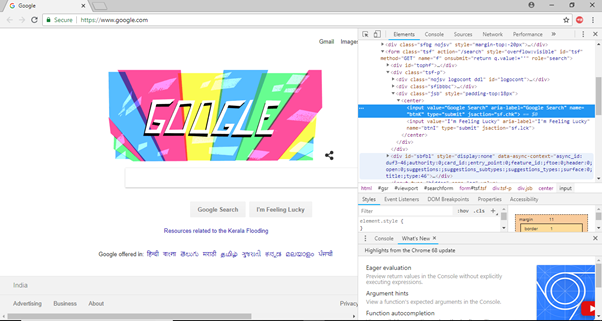
* Pick the value of id element i.e. "lst-ib".

Selenium WebDriver First Test Case

* Given below is the Java syntax for locating elements through "id" in Selenium WebDriver.
  1. driver.findElement(By.id (<element ID>))
* Here is the complete code for locating Google Search text box in our test script.
  1. driver.findElement(By.id ("lst-ib"))
* Now, 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 Google Search button.



* Pick the value of **name** element i.e. "btnK".

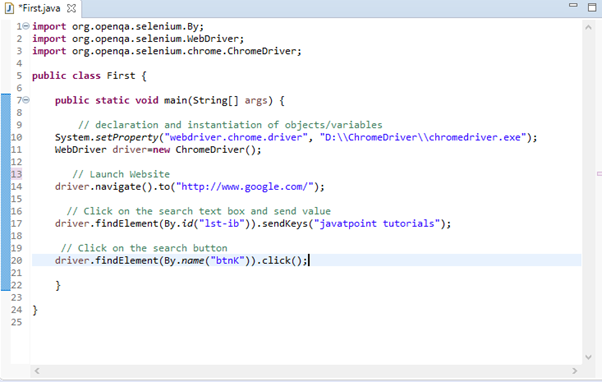
Selenium WebDriver First Test Case

* Given below is the Java syntax for locating elements through "name" in Selenium WebDriver.
  1. driver.findElement(By.name (<element name>))
* Here is the complete code for locating Google Search button in our test script.
  1. driver.findElement(By.name ("btnK"))

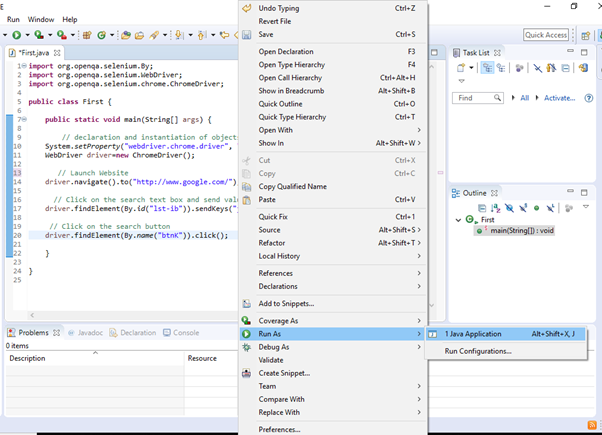
**Step5**. Now it is time to code. We have embedded comments for each block of code to explain the steps clearly.

1. **import** org.openqa.selenium.By;
2. **import** org.openqa.selenium.WebDriver;
3. **import** org.openqa.selenium.chrome.ChromeDriver;
5. **public** **class** First {
7. **public** **static** **void** main(String[] args) {
9. // declaration and instantiation of objects/variables
10. System.setProperty("webdriver.chrome.driver", "D:\\ChromeDriver\\chromedriver.exe");
11. WebDriver driver=**new** ChromeDriver();
13. // Launch website
14. driver.navigate().to("http://www.google.com/");
16. // Click on the search text box and send value
17. driver.findElement(By.id("lst-ib")).sendKeys("javatpoint tutorials");
19. // Click on the search button
20. driver.findElement(By.name("btnK")).click();
22. }
24. }

The Eclipse code window will look like this:



**Step6**. Right click on the Eclipse code and select **Run As > Java Application**.



**Step7**. The output of above test script would be displayed in Google Chrome browser.

