

Step 1: Basic process for using Selenium Java

- i) Download and Install Java
 - a) <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
 - b) Download the latest JDK file based on your System Requirements. Accept License Agreement.
 - c) Once installation is complete, open command prompt and type java-version to see if the version exists
- ii) Download Eclipse IDE
<https://www.eclipse.org/downloads/packages/release/neon/2/eclipse-ide-java-developers>

Step 2: Download Selenium Webdriver

Follow the steps given below to download the latest version of Selenium WebDriver –

2.1 Open Selenium download section using this link –

<http://www.seleniumhq.org/download/>

2.2 Scroll down a bit. You will see a section called Selenium Client & WebDriver Language Bindings. Here you will see download links next to different languages such as Java, C#, Ruby etc. Since we will be using Java with Selenium, you will need to download Java specific drivers. To do so, click on Download from the Java section

Selenium Client & WebDriver Language Bindings

In order to create scripts that interact with the Selenium Server (Selenium RC, Selenium Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers. These languages include both 1.x and 2.x style clients.

While language bindings for [other languages exist](#), these are the core ones that are supported by the main project hosted on google code.

Language	Client Version	Release Date			
Java	3.12.0	2018-05-08	Download	Change log	Javadoc
C#	3.12.0	2018-05-08	Download	Change log	API docs
Ruby	3.12.0	2018-05-08	Download	Change log	API docs
Python	3.12.0	2018-05-08	Download	Change log	API docs
Javascript (Node)	4.0.0-alpha.1	2018-01-13	Download	Change log	API docs

C# NuGet

NuGet latest release is 3.12.0, Released on 2018-05-08

- [WebDriver](#)
- [WebDriverBackedSelenium](#)

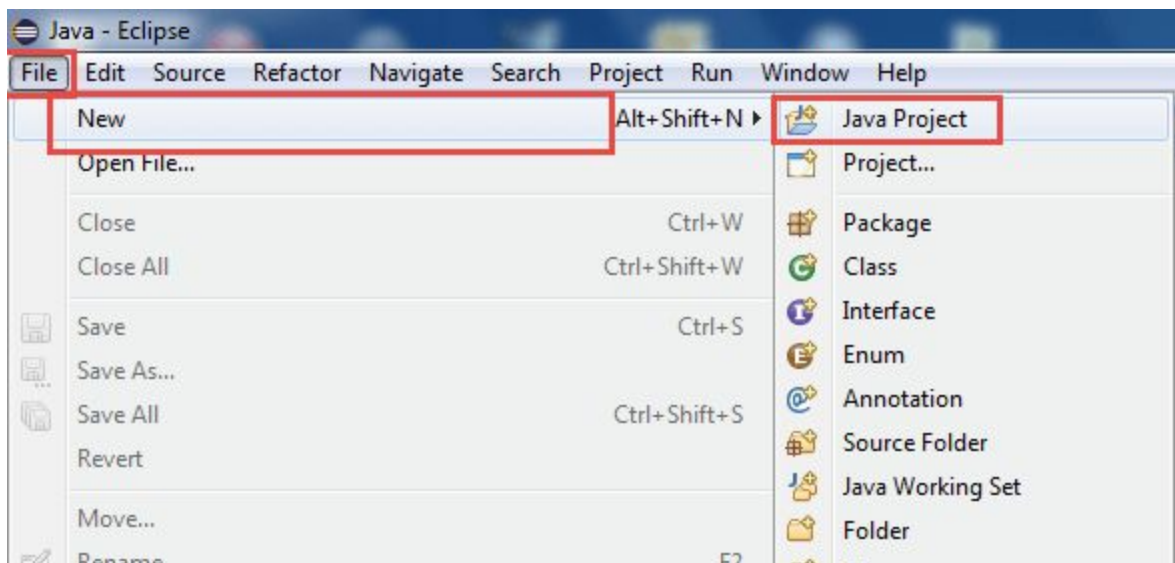
2.4 This download comes as a ZIP file. For simplicity, extract the contents of this ZIP file on your C drive so that you would have the directory "C:\selenium\".

Step 3: Configure Eclipse IDE for Selenium

3.1 Launch the "eclipse.exe"

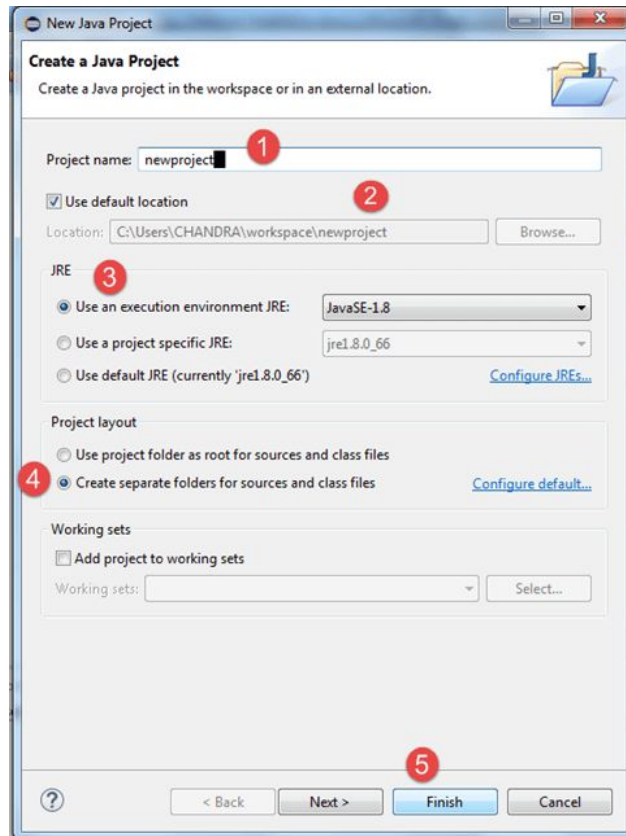


3.2 Create a new project through File > New > Java Project. Name the project as "newproject".



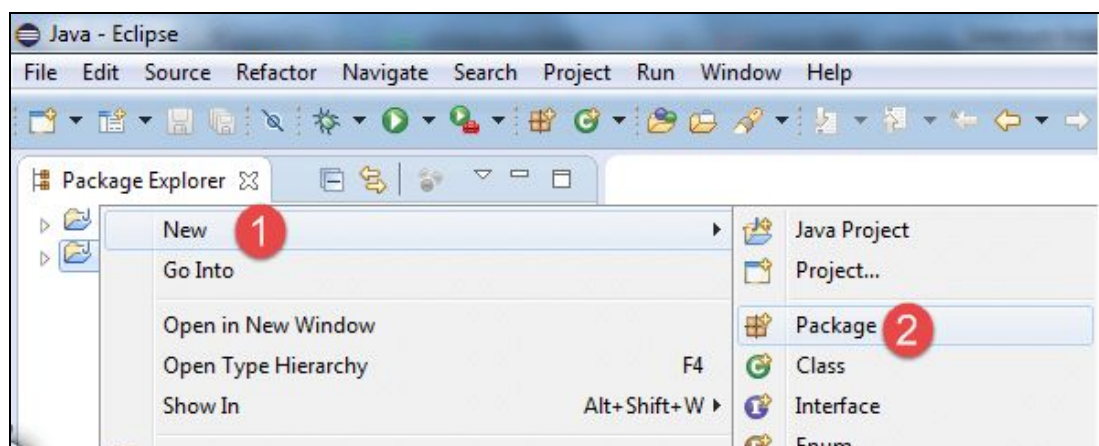
3.3 A new pop-up window will open enter details as follow

1. Project Name
2. Location to save project
3. Select an execution JRE
4. Select layout project option
5. Click on Finish button



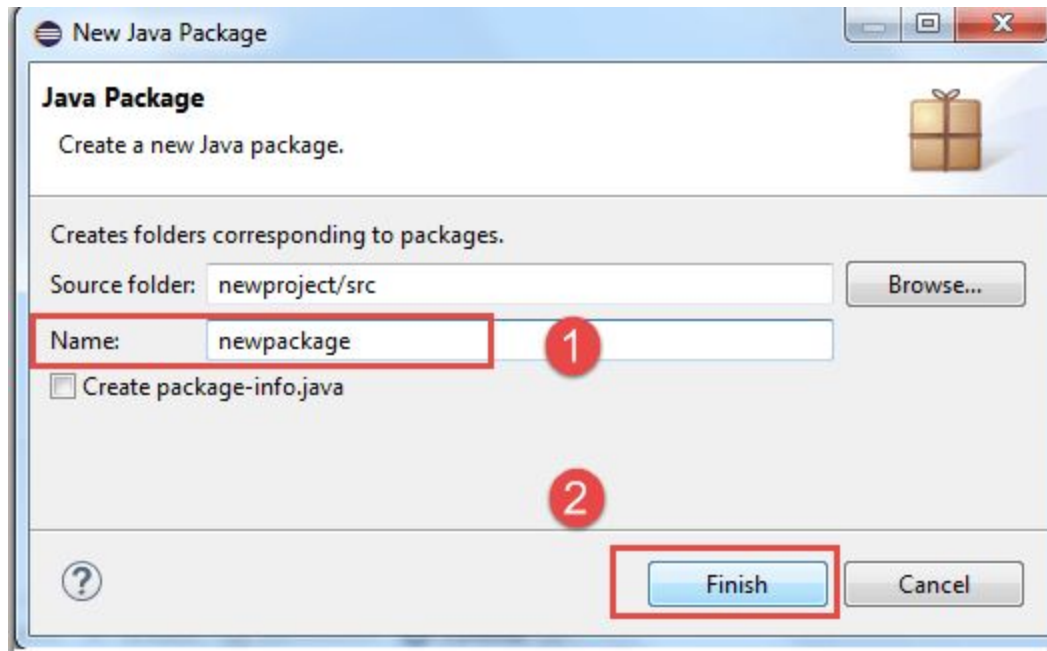
3.4 In this step,

1. Right-click on the newly created project and
2. Select New > Package, and name that package as "newpackage".



3.5 A pop-up window will open to name the package

1. Enter the name of the package
2. Click on Finish button

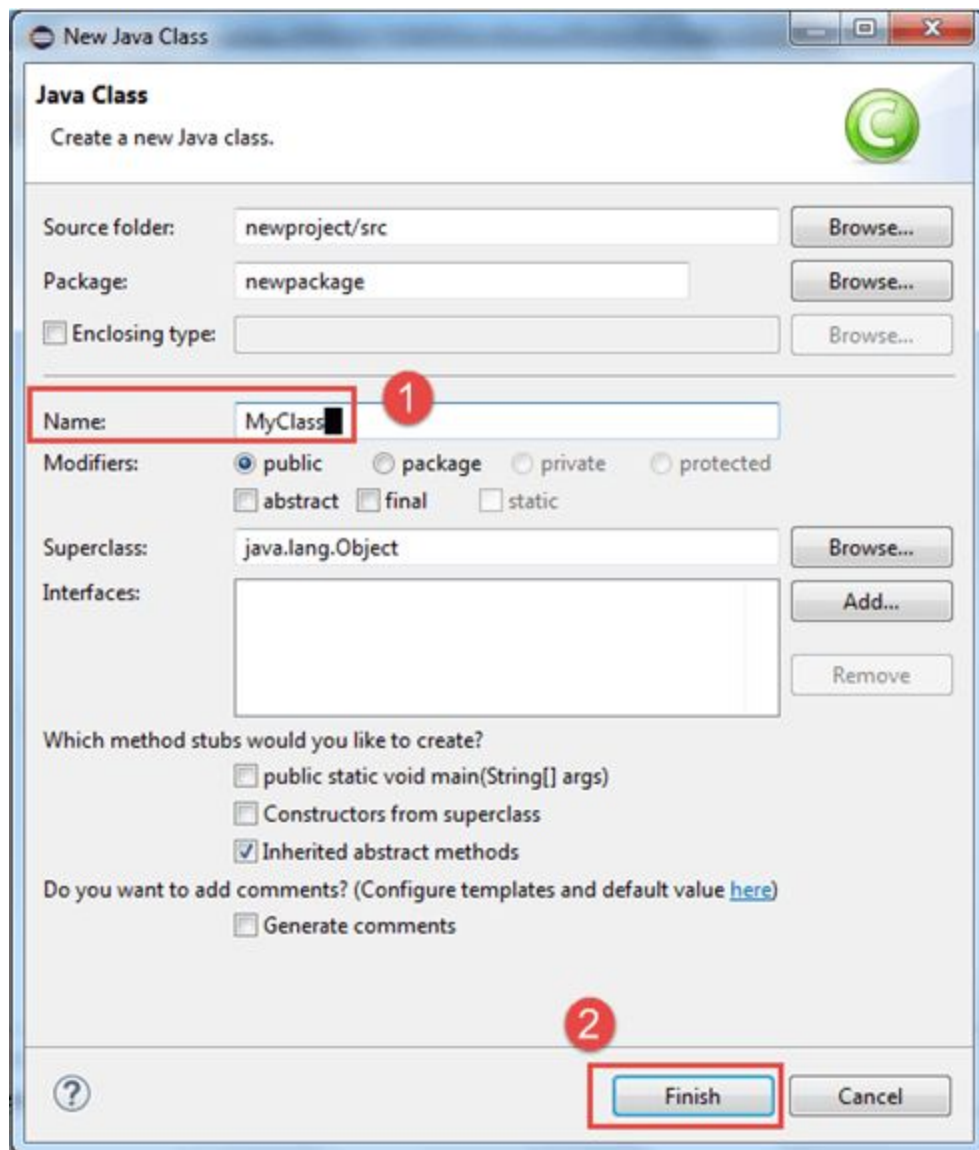


3.6 Create a new Java class **under** newpackage by right-clicking on it and then selecting- New > Class, and then name it as "MyClass".



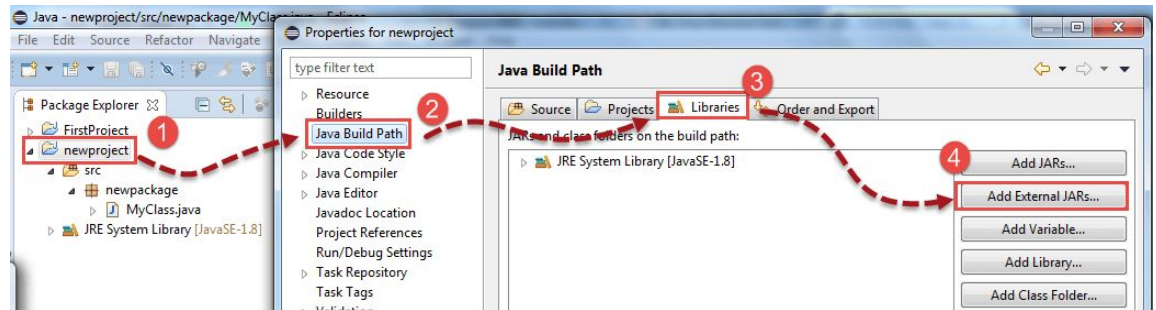
3.7 When you click on Class, a pop-up window will open, enter details as

1. Name of the class
2. Click on Finish button

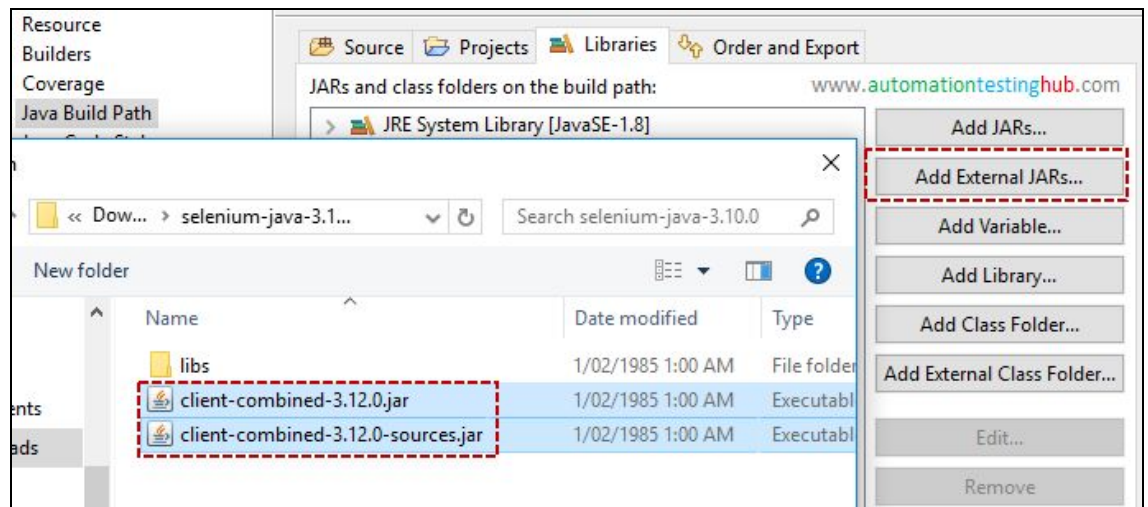


Step 4: Add Selenium JAR files in Eclipse project

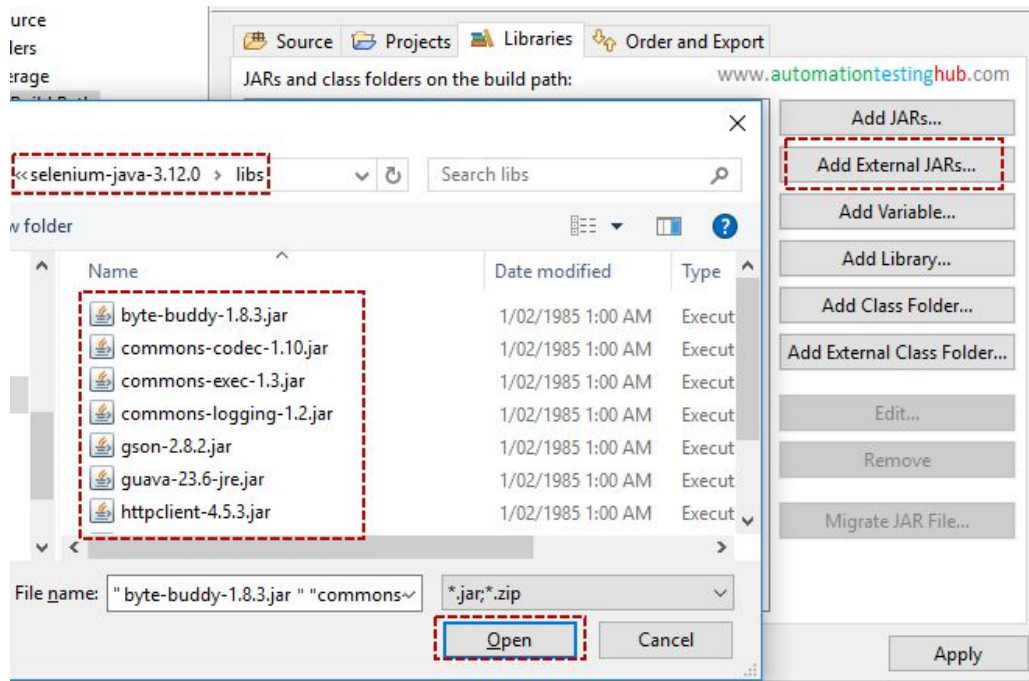
- 4.1. Right-click on "newproject" and select **Properties**.
- 4.2. On the Properties dialog, click on "Java Build Path".
- 4.3. Click on the **Libraries** tab, and then



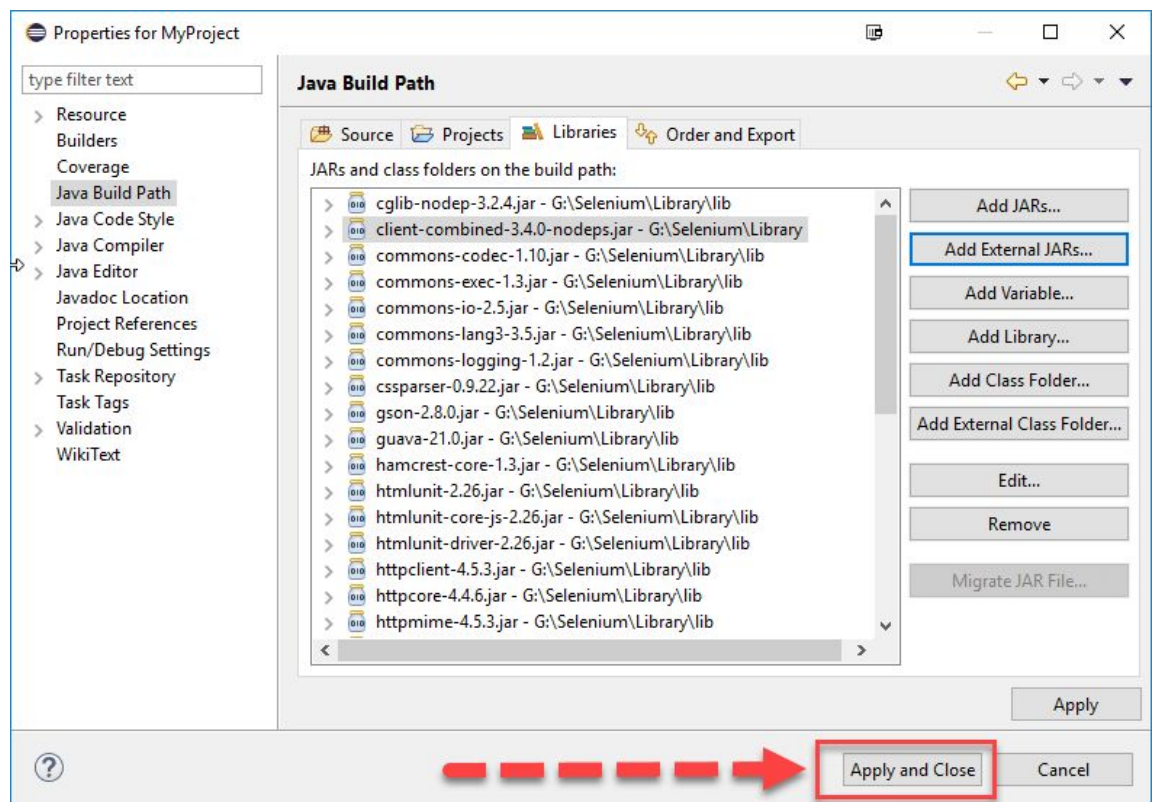
- 4.4. Click on "Add External JARs.."
- 4.5. Now navigate to the folder where you had downloaded Selenium WebDriver JAR files. Select the selenium JAR (named client-combined...) and click on Open button



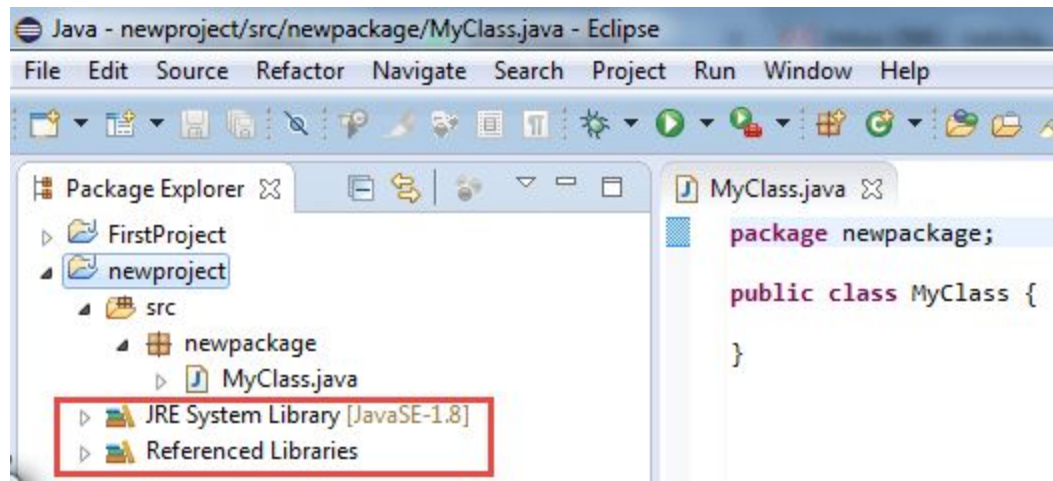
- 4.6. Do the same for all the JARs in the lib folder



4.7. Once done, click "Apply and Close" button



4.8. Add all the JAR files inside and outside the "libs" folder. Your Properties dialog should now look similar to the image below.



4.9. Finally, we are done importing Selenium libraries into our project.

Step 5: Download Selenium ChromeDriver

5.1 Open ChromeDriver download page –

<https://sites.google.com/a/chromium.org/chromedriver/downloads>

After opening the link. Click on **ChromeDriver 73.0.3683.68**

ChromeDriver - WebDriver for Chrome

Search this site

CHROMEDRIVER

CAPABILITIES & CHROME OPTIONS

CHROME EXTENSIONS

CHROMEDRIVER CANARY

CONTRIBUTING

▼ DOWNLOADS

VERSION SELECTION

▼ GETTING STARTED

ANDROID

CHROME OS

▼ LOGGING

PERFORMANCE LOG

MOBILE EMULATION

▼ NEED HELP?

CHROME DOESN'T START OR CRASHES IMMEDIATELY

CHROMEDRIVER CRASHES

CLICKING ISSUES

DEVELOPER WINDOW KEEPS CLOSING

OPERATION NOT SUPPORTED WHEN USING REMOTE DEBUGGING

SECURITY CONSIDERATIONS

Downloads

Click on ChromeDriver 73.0.3683.68

Current Releases

- If you are using Chrome version 74, please download [ChromeDriver 74.0.3729.6](#)
- If you are using Chrome version 73, please download [ChromeDriver 73.0.3683.68](#)
- If you are using Chrome version 72, please download [ChromeDriver 2.46](#) or [ChromeDriver 72.0.3626.69](#)
- For older version of Chrome, please see below for the version of ChromeDriver that supports it.

If you are using Chrome from Dev or Canary channel, please download [ChromeDriver 74.0.3729.6](#). This is not officially supported, but in most cases it should work without major issues.

For more information on selecting the right version of ChromeDriver, please see the [Version Selection](#) page.





ChromeDriver 74.0.3729.6

Supports Chrome version 74

- Fixed a bug that generated unexpected debug.log file on Windows
- Fixed mouse clicking and drag / drop inside SVG document
- Added cache-control header in responses from ChromeDriver
- Fixed the type of error when click is intercepted by a different element
- Fixed a bug that caused ChromeDriver to fail on Linux devices without /dev/shm
- Fixed some types of double click events
- Fixed Get Sessions command

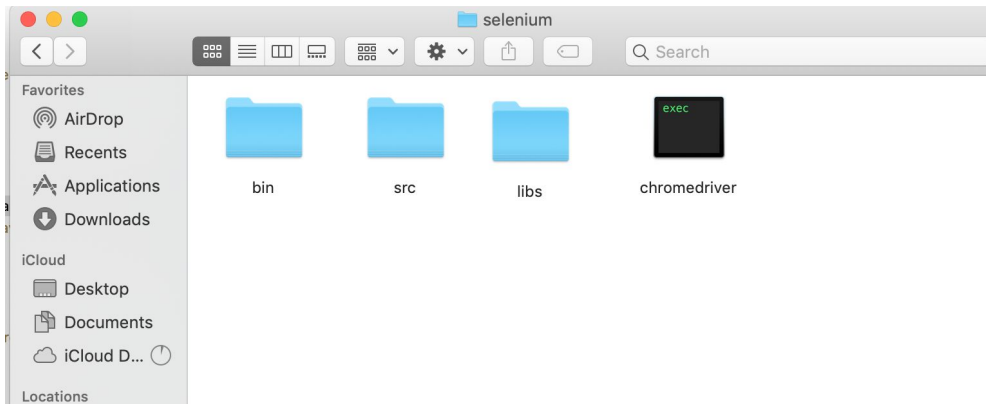
5.2 You will be navigated to ChromeDriver download page which contains ChromeDriver for Mac, Windows and Linux operating systems.

Index of /73.0.3683.68/

Name	Last modified	Size	ETag
Parent Directory	-	-	-
 chromedriver_linux64.zip	2019-03-07 22:34:54	4.78MB	16d6ef61ff19649df9251d742ed85c62
 chromedriver_mac64.zip	2019-03-07 22:34:56	6.67MB	9af243f31d0e7e0444bdcc11ec35aa5d
 chromedriver_win32.zip	2019-03-07 22:34:58	4.38MB	47bf69232c8b62139e642927cccf2e4
 notes.txt	2019-03-07 22:41:43	0.00MB	6899c47dec5549c1145f209a409c19a1

5.3 Once the zip file is downloaded, you can unzip it to retrieve **chromedriver.exe**

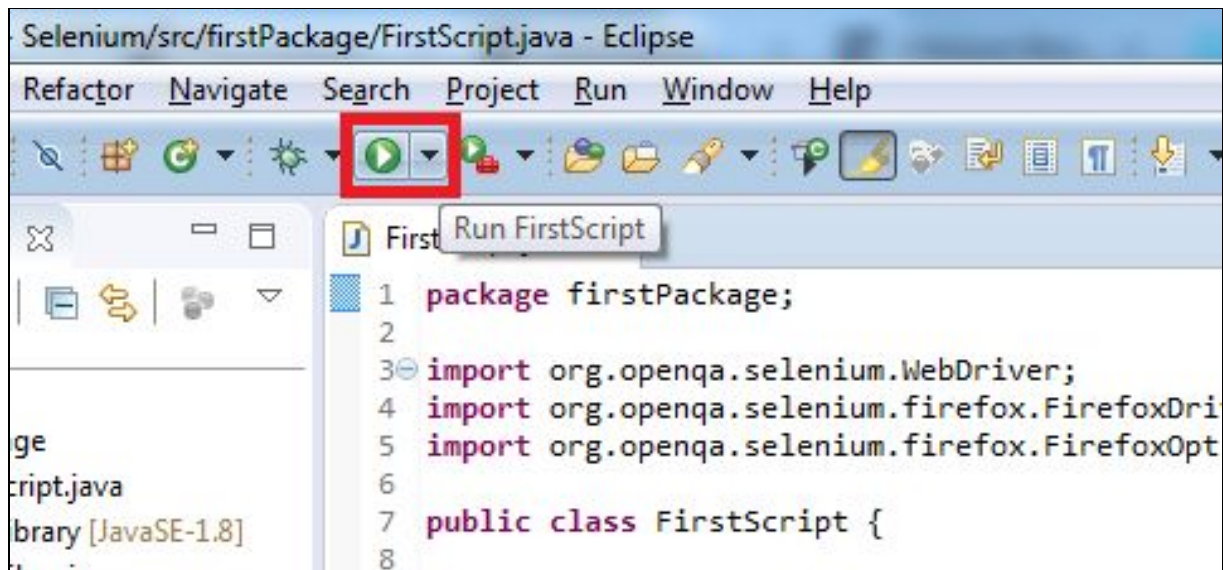
5.4 Copy the chromedriver.exe to “newproject” folder



5.6 Once your browser setup is complete, you would have written the code snippet which launches the browser. The example below shows how the code would look like if you use Chrome

```
1 package firstPackage;
2
3 import org.openqa.selenium.WebDriver;
4 import org.openqa.selenium.chrome.ChromeDriver;
5
6 public class FirstScript {
7
8     public static void main(String[] args) {
9         System.setProperty("webdriver.chrome.driver",
10             "chromedriver");
11         WebDriver driver=new ChromeDriver();
12
13         driver.get("https://www.mdbootstrap.com");
14     }
15 }
```

5.7 To run the script, click on the run icon from the toolbar



If everything works fine, you will see that a new browser window will open and <https://www.mdbootstrap.com> webpage would be opened in the browser.

Locators in Selenium:

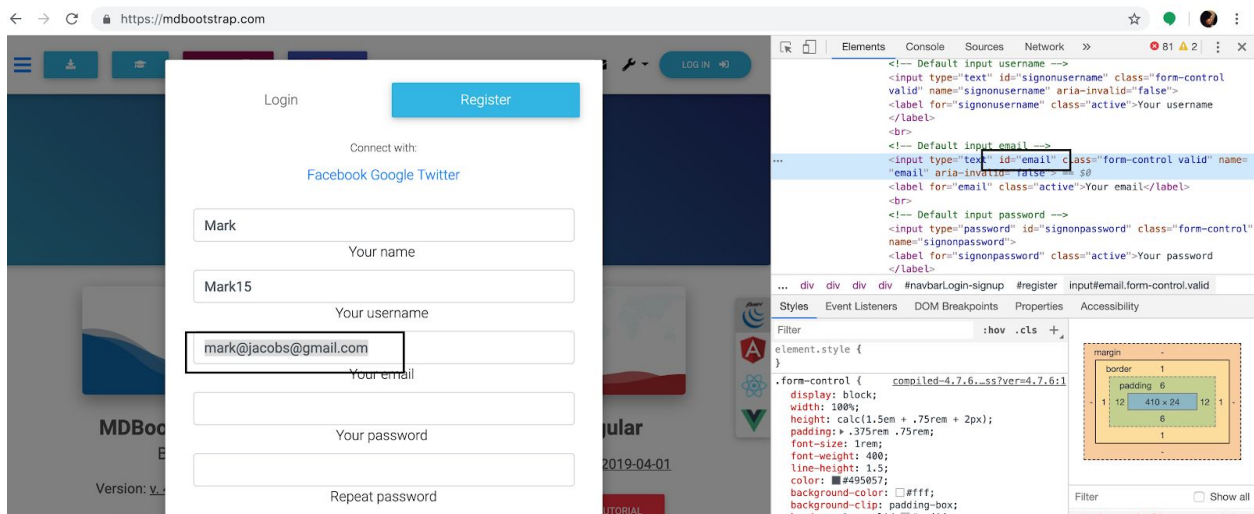
Selenium provides a number of Locators to precisely locate a GUI element. Locating elements in Selenium WebDriver is performed with the help of `findElement()` and `findElements()` methods provided by Selenium WebDriver and WebElement class.

- **Locating by id:**

Locating by id is the most common way of locating elements since ID's are unique for each element. We can locate the element using id attribute.

Syntax:

```
driver.findElement(By.id("email")).sendKeys("user.test@gmail.com");
```



- **Locating by Name**

We use name locator to identify the elements on our webpage which is similar to locating by ID except we use “name=” prefix. If there are multiple elements with the same Name locator then the first element on the page is selected and the test might fail.

Example :

```
driver.findElement(By.name("username")).sendKeys("abc12@gmail.com");
```

- **Locating by ClassName**

We use className locator to identify the element which matches values specified in the attribute name "class"

Example :

```
driver.findElement(By.className("button-collapse")).click();
```

ClassName with spaces like "button collapse" shows it has two different class names and test fails.

- **Locating by Link Text**

This type of locator applies only to hyperlink texts. We access the link by prefixing our target with "linkText=" and then followed by the hyperlink text.

Example : **driver.findElement(By.linkText("Login")).click();**

- **Locating by CSS Selector**

It is the combination of an element selector and a selector value which identifies the web element within a web page.

Example :

```
driver.findElement(By.cssSelector("#password2")).sendKeys("SelTest1234");
```

- **XPath in Selenium WebDriver**

Sometimes we may not be able to identify the element using the locators such as id, class, name, etc. In those cases, we use XPath to find an element on the web page. Xpath is unique. It has two types :

1. Absolute path : It is the direct way to find the element which means it identifies the element very fast, but the disadvantage of the absolute XPath is that if there are any changes made in the path added or removed then that XPath gets failed and does not run. It starts with a single forward slash (/)

Example: /html/head/body/table/tbody/tr/th

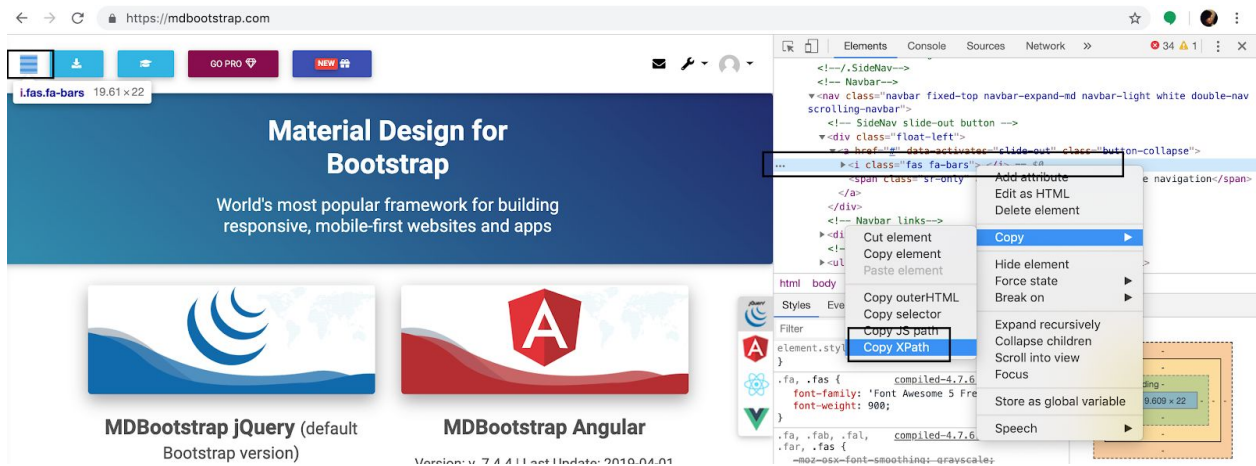
2. Relative path : It starts with the double forward slash (//), which means it can search the element anywhere at the webpage. It can be started from the middle or in between to search.

Example : **//*[@id=\"AJAXAuthRegisterBtn\"]**

To get a xpath : Right click on the element you are inspecting go to copy -> copy xpath

Syntax:

```
driver.findElement(By.xpath("//*[@id=\"AJAXAuthRegisterBtn\"]")).click();
```



Advantages:

1. **Open Source:** Selenium allows users to share, extend, and modify the available code.
2. **Support Languages:** Selenium supports a range of languages, including Java, Perl, Python, C#, Ruby, Groovy, Java Script, etc.
3. **Server Starting Not Required:** A major benefit of automation testing with Selenium WebDriver is that you don't need to start any server prior to testing.
4. **Supports Operating Systems:** Selenium can operate and support across multiple Operating Systems (OS) like Windows, Mac, Linux, UNIX, etc.
5. **Support across browsers:** Selenium provides support across multiple browsers, namely, Internet Explorer, Chrome, Firefox, Opera, Safari, etc. This becomes highly resourceful while executing tests and testing it across various browsers simultaneously.
6. **It can be integrated with Maven, Jenkins & Docker to achieve Continuous Testing.**

Limitations:

1. **It only supports Web based applications:** We can use Selenium only to test web applications. We cannot test desktop applications
2. **Handling pop up windows:** Windows-based pops are part of the operating system. It's beyond selenium's capabilities.
3. **Handling captcha:** Handling captcha is a limitation in selenium. There are some third-party tools to automate captcha but still, we cannot achieve 100% results.
4. **There is limited reporting facility.** But we can overcome that issue by integrating it with frameworks like TestNG or Junit.