**Selenium Webdriver**

# Selenium Introduction

## About Selenium

Selenium is a software testing framework for the web that facilitates the automation of browsers. The Selenium project produces various tools for automation testing such as Selenium IDE, Selenium Remote Control (RC), Selenium Grid and Selenium 2.0 & WebDriver. Learning all the tools will give you many different options for approaching different automation problems. The entire suits of tools result in a rich set of testing functions specially geared to the needs of testing of web application of all types.

## Why Selenium

* – Selenium is an open source tool with Corporate backing.
* – The tests can then be run against most modern web browsers.
* – Selenium deploys on Windows, Linux, and Macintosh platforms.
* – It allows recording, editing, and debugging tests.
* – Recorded tests can be exported in most language e.g. html, Java, .net, perl, ruby etc.
* – Selenium has the support of some of the largest browser vendors who have taken (or are taking) steps to make Selenium a native part of their browser.

## Selenium Components

* – Selenium 1/ Selenium RC or Remote Control: Selenium RC was the main Selenium project for a long time before the Selenium Webdriver merge brought up Selenium 2, the newest and more powerful tool.
* – Selenium IDE (Integrated Development Environment): The Selenium IDE is a simple but powerful Firefox extension that lets users record and replay sets of browser interactions as test cases.
* – Selenium Grid: Selenium Grid is a server that allows tests to use web browser instances running on remote machines. It allows the Selenium RC solution to scale for large test suites and for test suits that must be run in multiple environments. Different tests can be run at the same time on different remote machines.
* – Selenium Web Driver : It also provides a test domain-specific language (Selenese/Webdriver) to write tests in a number of popular programming languages, including Java, C#, Groovy, Perl, PHP, Python and Ruby. It is the newest addition to the Selenium toolkit. This provides all sort of awesome features, including a more cohesive and object oriented API as well as an answer to the limitation of the old implementation.

## Selenium WebDriver

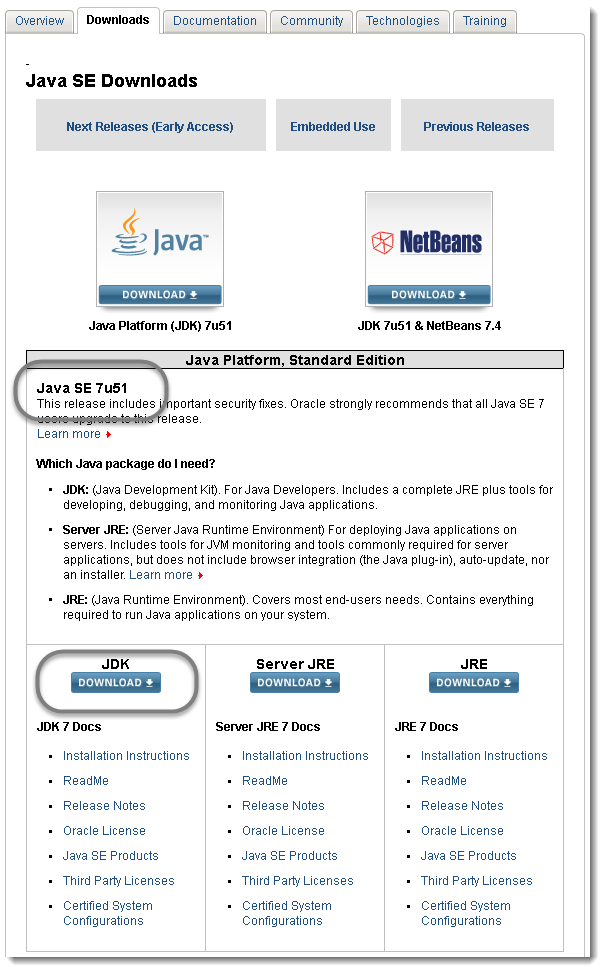
The primary new feature in Selenium 2.0 is the integration of the WebDriver API. WebDriver is designed to provide a simpler, more concise programming interface in addition to addressing some limitations in the Selenium-RC API. It enables you to use a programming language to write test scripts in different programming languages like html, Java, .net , perl, ruby and which enables you to use conditional operations, looping and other programming concepts which makes you test script robust. Selenium-WebDriver was developed to better support dynamic web pages where elements of a page may change without the page itself being reloaded. WebDriver’s goal is to supply a well-designed object-oriented API that provides improved support for modern advanced web-app testing problems.

# Download and Install Java

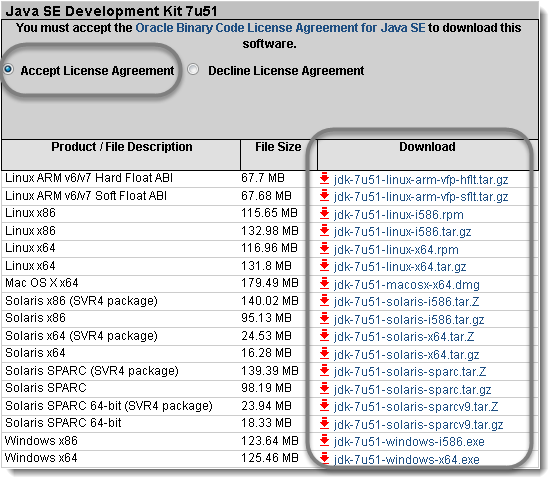
## Steps to Download and Install Java Development Kit (JDK) on Windows

First of all you need to install ***JDK*** (Java development kit) in your system. So your next question will be “how to download and install Java“. [***Click here***](http://www.oracle.com/technetwork/java/javase/downloads/index.html)to download Java and install it in your system as per given installation guide over there or follow the below mentioned steps.

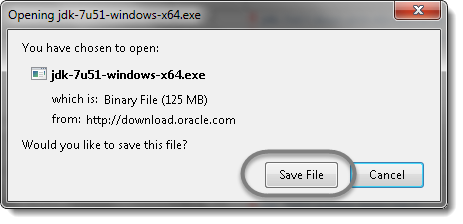
1) Visit the [***Java downloads page on Oracle’s***](http://www.oracle.com/technetwork/java/javase/downloads/index.html) website to find the ***JDK environment*** download. Scroll down until you find ***Java SE Latest Version*** and download ***JDK***.



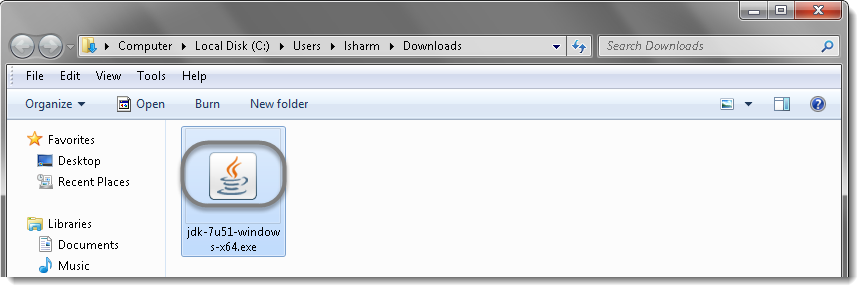
2) Once you have selected download, ***accept*** the terms of service and choose the correct OS corresponding for the specific JDK. (Windows, Mac, Linux, etc.)



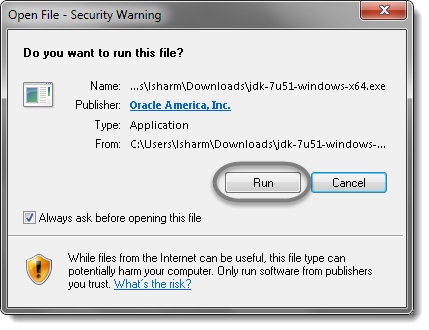
3) ***Save*** the ‘**.exe**‘ file to your disk.



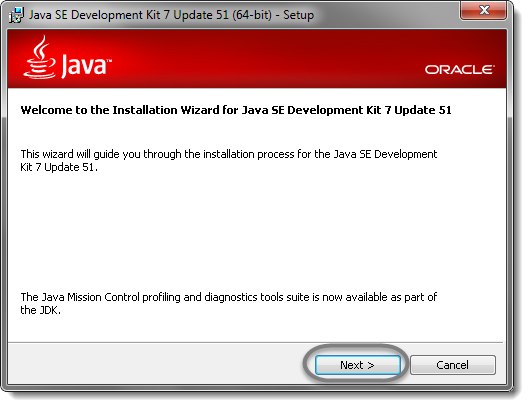
4) Once the download is complete, ***double click*** the file to begin the installation of ***JDK***.



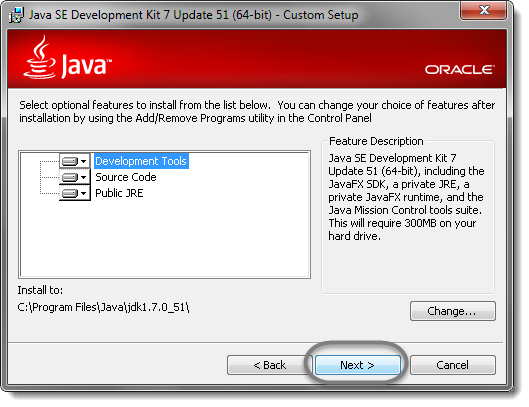
5) To run the installer, click ***Run***.



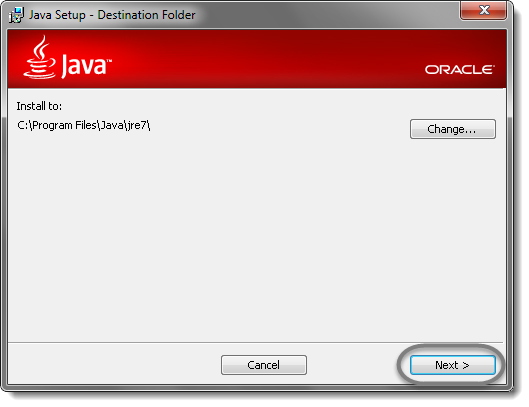
6) The installation process starts. Click the **Next** button to continue the installation.



7) On the next screen you will encounter some options. Just leave these alone and click ***Next*** unless you know what you are doing.



8) After the initial installation is done, a pop up asking you where your source java files will be. You can choose to change where you want to keep your folder but it’s best to stick with what you were given first. Click **Next** to continue.



9) Let the installation finish.



10) A few brief dialog confirm the last steps of the installation process; click **Close** on the last dialog. This will complete Java installation process.

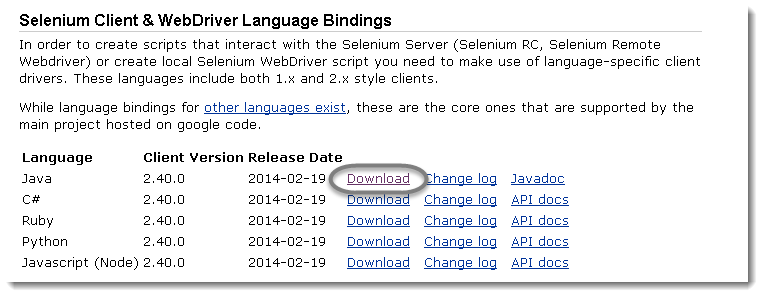


# Download Selenium Webdriver Java client

# Download Selenium WebDriver Java client

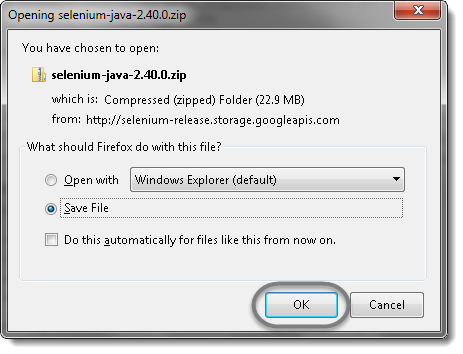
Selenium WebDriver supports many languages and each language has its own client driver. Here we are configuring selenium 2 with java so we need ‘***webdriver Java client driver***‘.

1) [***Click here***](https://docs.seleniumhq.org/download/) to go on WebDriver Java client driver download page for WebDriver download file. On that page click on ‘***Download***‘ link of java client driver as shown in the below image.

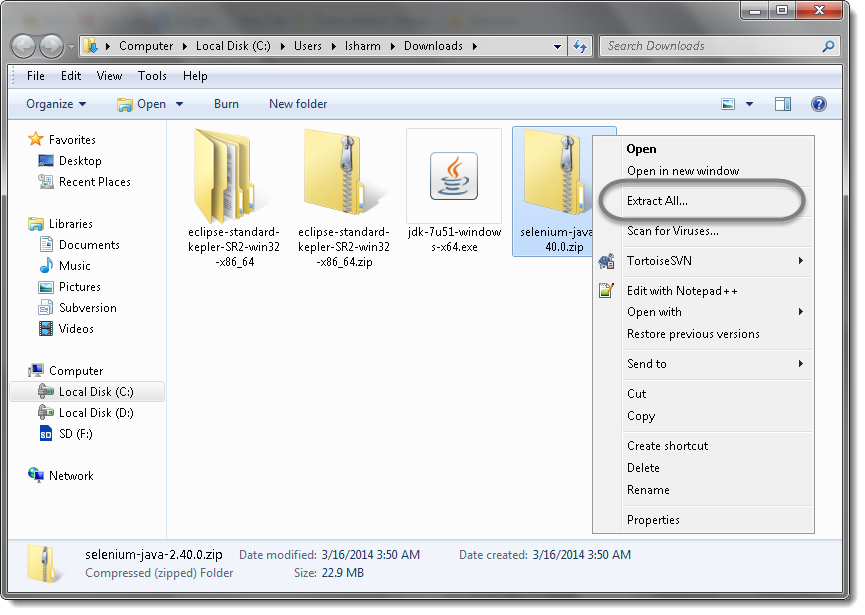


***Note:*** On Oct17, the latest version of Selenium is ***3.5.3***

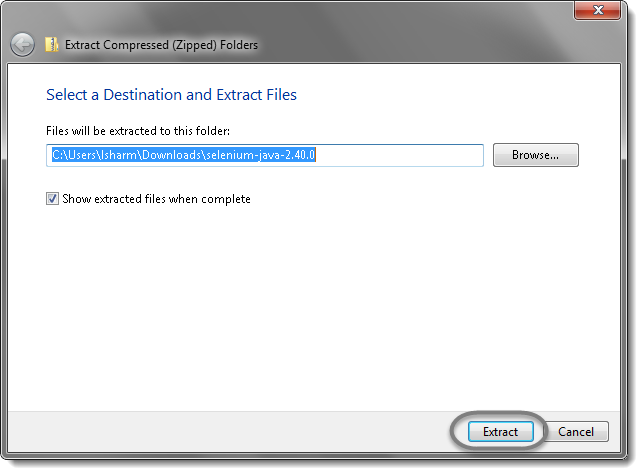
2) ***Save*** the .zip file to your disk.



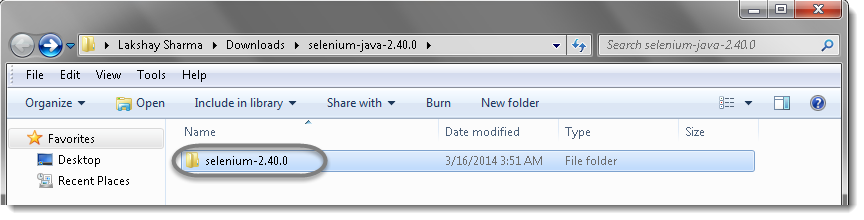
3) Once you have downloaded the archive you will need to **Extract**the zip file, which will create the unzipped Selenium Java folder.



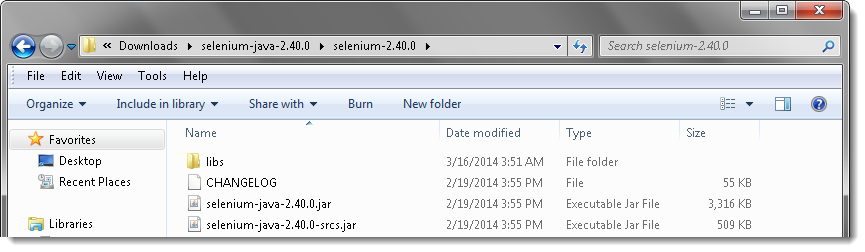
4) You may want to ***Extract*** the archive to the folder where you are maintaining the test ware for your project. I like to extract it at the same location and then move it as per my needs.



5) Once the extraction process is complete, ***Open*** the Selenium folder.



6) There will be ‘***libs***‘ folder, 2 jar files and change log in unzipped folder as shown in bellow figure. We will use all these files for configuring WebDriver in Eclipse.



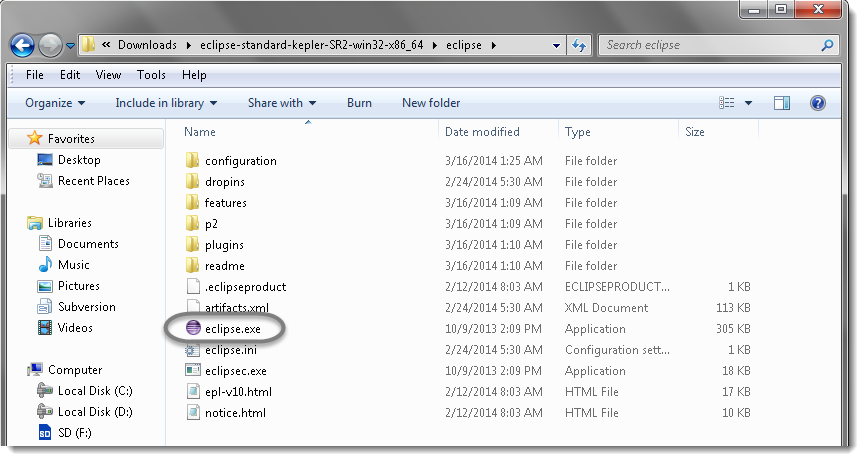
# Configure Eclipse with Selenium WebDriver

To **configure Eclipse with Selenium WebDriver**, we need to perform the following activities:

* ***Launch the Eclipse IDE & Create a Workspace***
* ***Create a new Project***
* ***Create a new Package***
* ***Create a new Class***
* ***Add External libraries to the project***

## Launch the Eclipse IDE & Create a Workspace

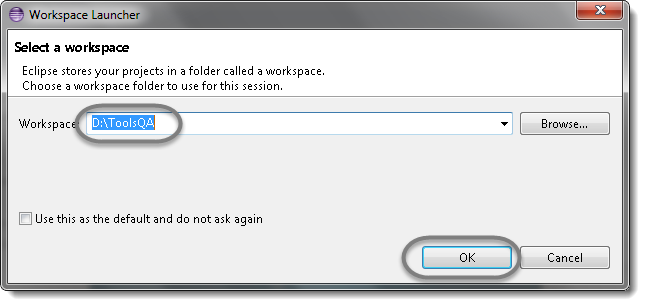
1) Double click on ‘***eclipse.exe***‘ to start eclipse. First time when you start eclipse, it will ask you to select your ***workspace*** where your work will be stored as shown in below image.

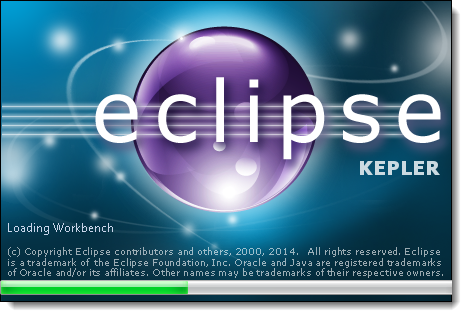


2) Create a ‘***working directory’*** for all of your projects. Think of it like ‘***My Documents’*** in the Windows operating system. It’s a folder which contains a lot of your documents, but there’s nothing to prevent you from creating another folder called ‘My Other Documents‘ (for instance) to house other documents.

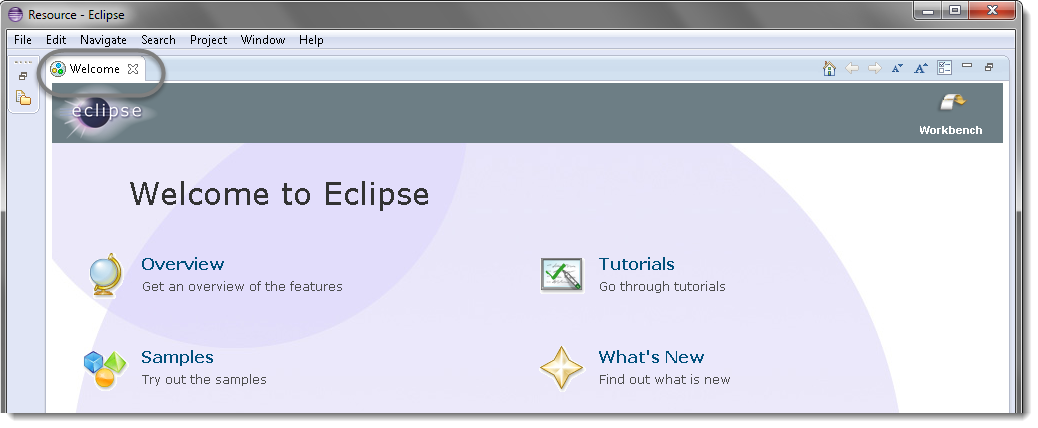
Typically you only need one workspace, and you can think of it as your ‘My Documents‘ for Java code. If you wanted to, you could have more than one, but chances are you won’t have a use for more. I like to choose my own workplace location and will place all my ***ToolsQA*** tutorial projects under it.

You can change it later on from ‘***Switch Workspace***‘ under ‘***File***‘ menu of eclipse. After selecting workspace folder, Eclipse will be open.





3) You may see the window like this, this is the **Welcome window** for Eclipse. You may close this window.

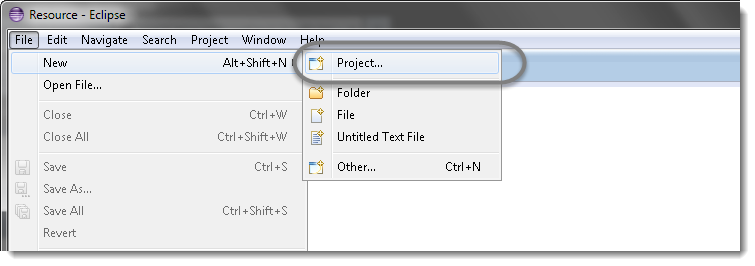


## Create a new Project

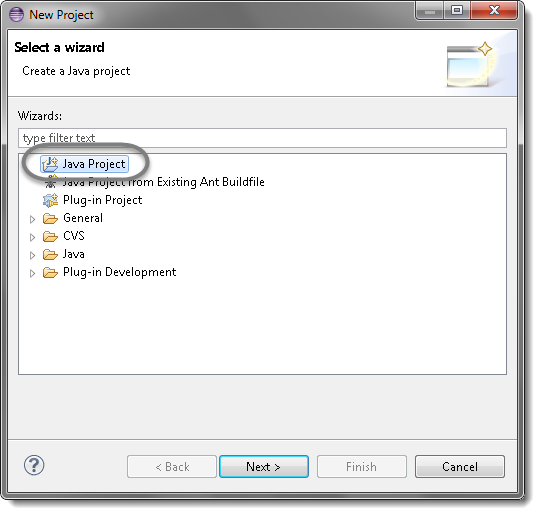
***Projects:*** A collection of related code. Generally speaking, each project encompasses one independent program. Each programming assignment you do will typically require its own project.

Once you’ve established your workspace, you’ll want to create a project and begin writing code. In Eclipse, projects are the next-smallest functional unit after workspaces, but where you might have only one workspace, you will usually have several projects inside one workspace.

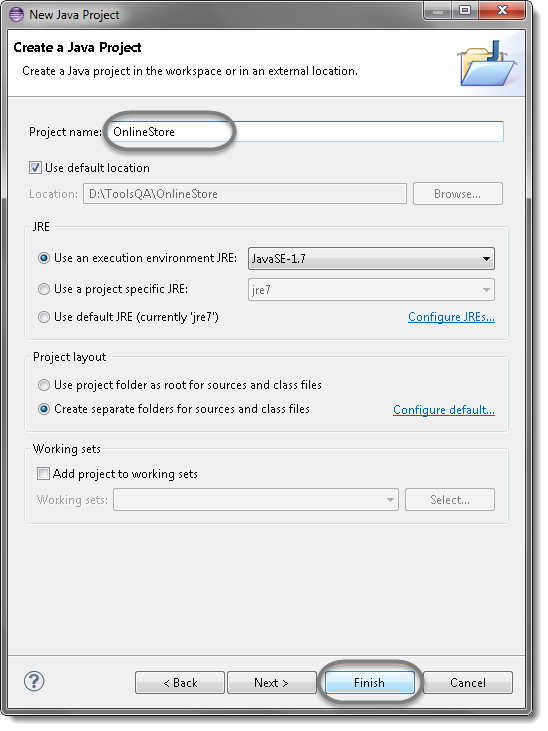
1) Create new Java Project from ***File*** > ***New*** > ***Project*** .



2) Select ***Java Project***and click***Next***.



3) Give your Project name ‘***OnlineStore***‘ as shown in below given figures. Click on ***Finish*** button.



***Note:***I am naming this as OnlineStore, as [***Toolsqa.com***](http://toolsqa.wpengine.com/)gives you the opportunity to practice automation on [***Live demo site***](http://www.store.demoqa.com/). This demo website is a complete ecommerce website where a customer can register and purchase electronic items.

***Website :***[***www.store.demoqa.com***](http://www.store.demoqa.com/)

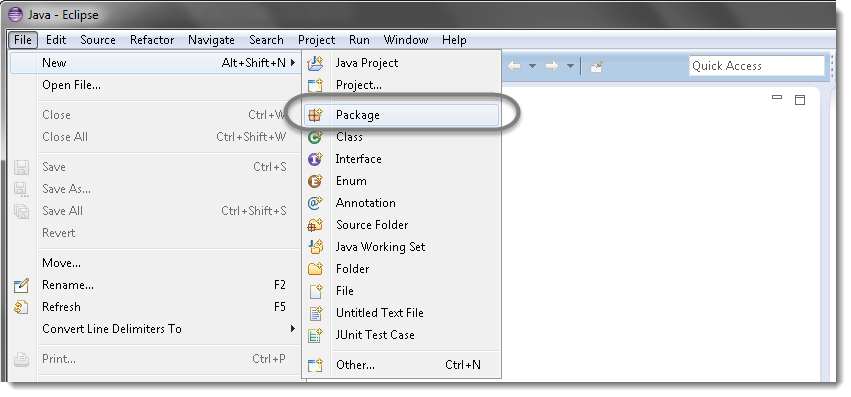
4) You may or may not see this message but if in case you get any, check ***Remember my decision*** and click on ***Yes***.



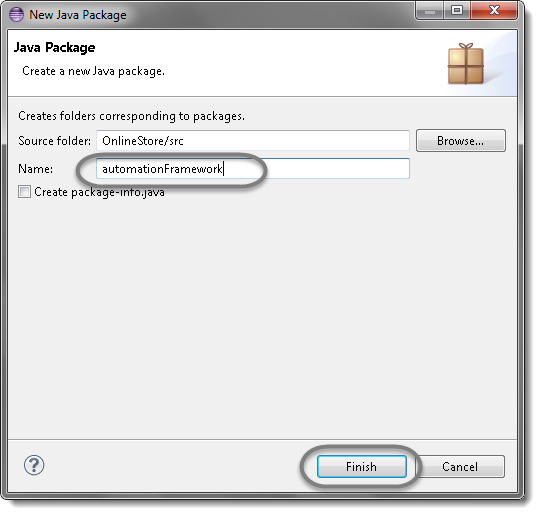
Now your new created project ‘OnlineStore‘ will display in eclipse project explorer.

## Create a new Package

1) Right click on Project name ‘***OnlineStore***‘ and select ***New*** > ***Package***.



2) Give your Package name ‘***automationFramework***‘ and click on ***Finish*** button.

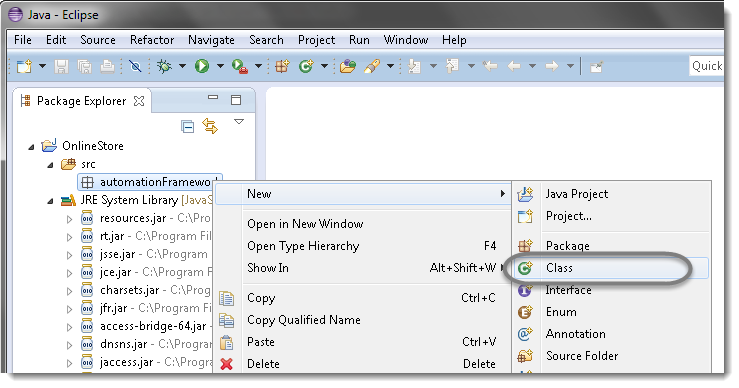


Now you can see a new package with name ‘automationFramework‘ under project name ‘OnlineStore‘.

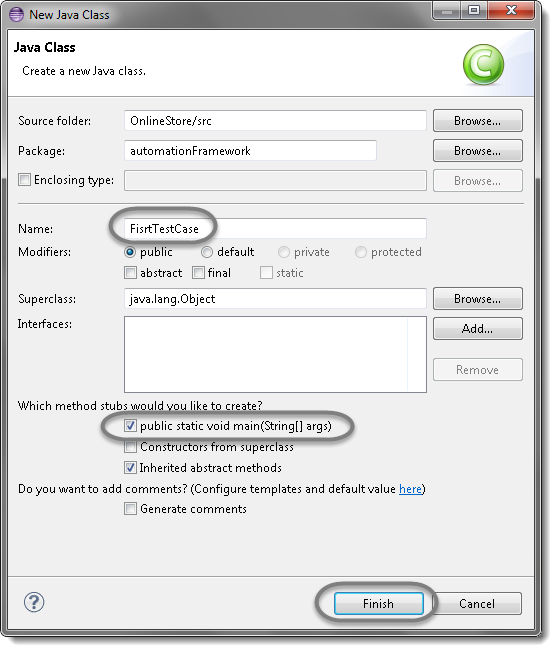
## Create a new Class

Now that you have a project set up, you’re going to want start writing some new classes.

1) Right click on Package ‘***automationFramework***‘ and select ***New*** > **Class.**

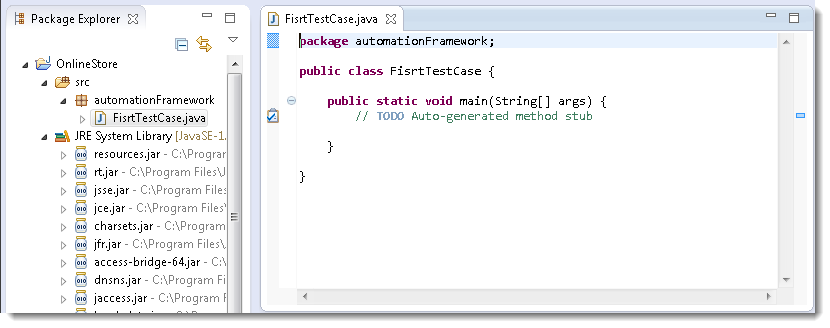


2) Give your Class name ‘***FirstTestCase***‘, check the option ‘***public static void main***‘ and click on ***Finish*** button. This will bring up totally a sweet class creation window.



***Note:***In case of not creating class for Main test case, please do not click ‘public static void main’. We need to select it only in case of writing test cases which we are going to execute and from where we call other classes. For functional classes, POM classes or any other classes we don’t need this to be checked.

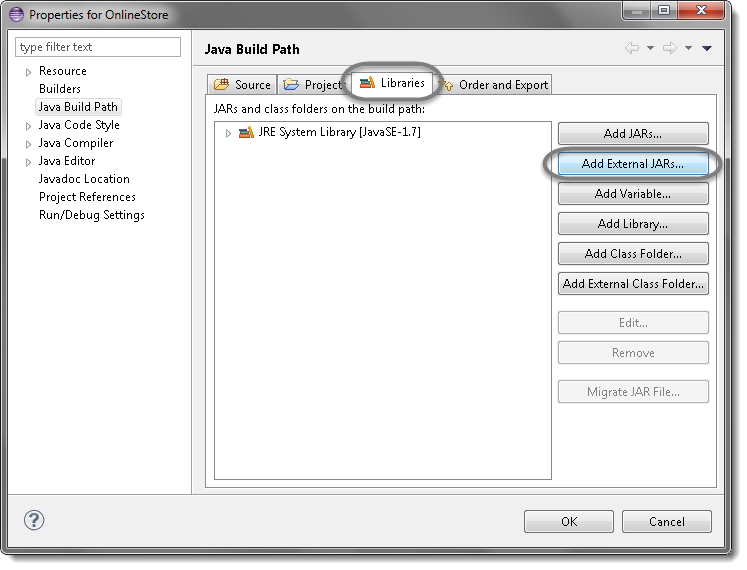
3) Now your Eclipse window will looks like bellow.



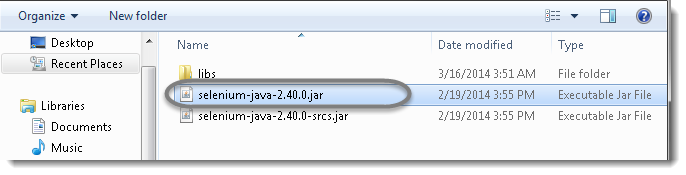
## Add External Jars to Java build path

Now you need to add Selenium WebDriver’s Jar files in to Java build path.

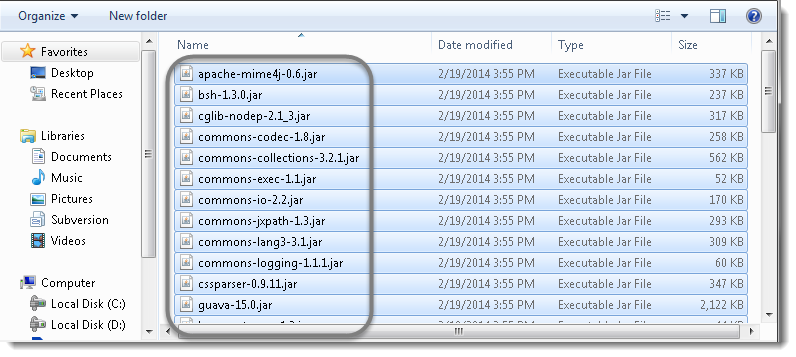
1) Right click on Project ‘***OnlineStore***‘ > ***Select Properties*** > **Java build path.**Then navigate to ***Libraries*** tab and click ***Add External JARs***.



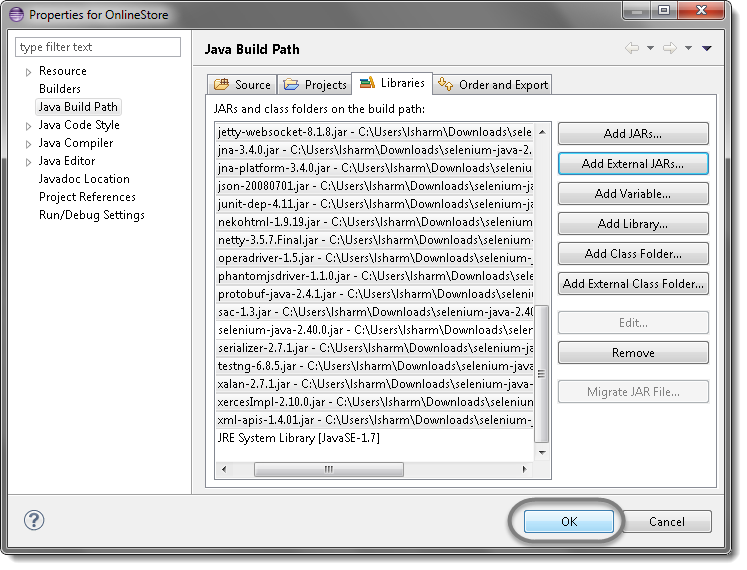
2) Add Selenium Java jar, you may add the source file too.



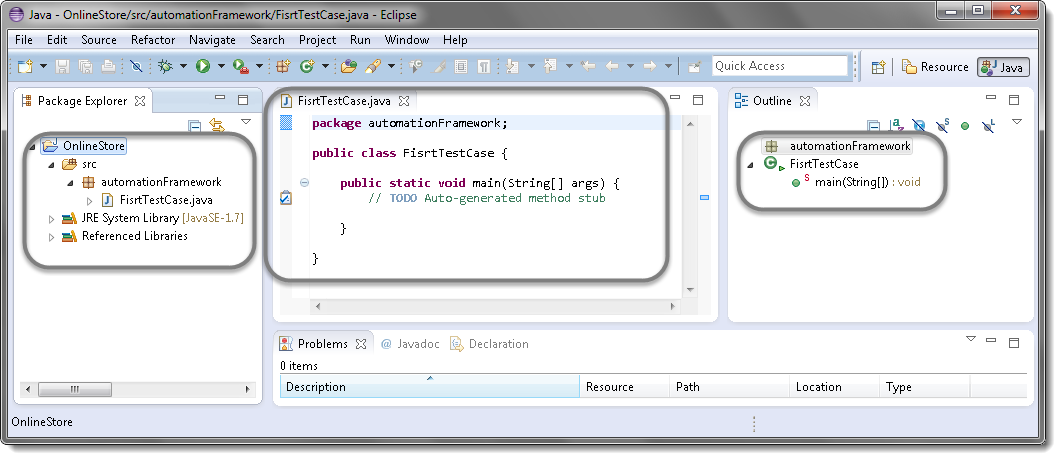
3) Add all the jars from the ***libs*** folder as well.



4) Click **OK**.



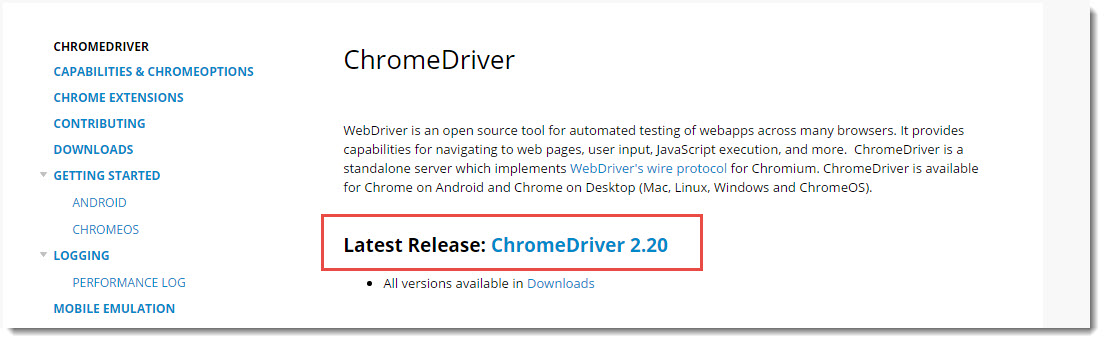
That’s all about configuration of WebDriver with eclipse. Now you are ready to write your test script in eclipse and run it in WebDriver.



# Running tests in Chrome browser

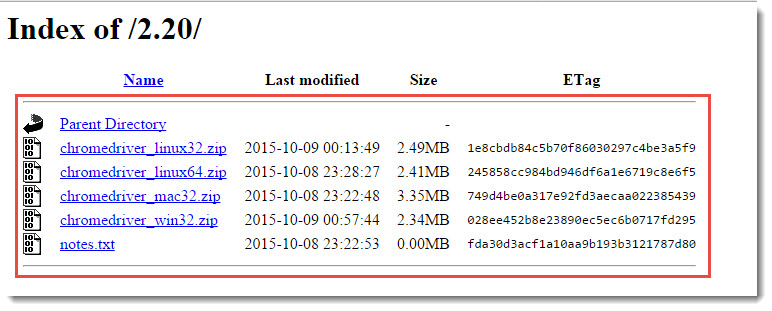
## Chrome Driver Server

First we have to download the Chrome Driver Server from the Chromium project [***here***.](https://sites.google.com/a/chromium.org/chromedriver/) This website details out all the features that are available in the Chrome driver. We are, at present, interested in downloading the Server. On the page you will find a section of latest release. That is where we will have a link to download the latest release of Chrome WebDriver. Refer to the image below



Click on the link and you can follow it to download the Chrome driver. The main reason why I wanted to show you this page is to make you aware of the brilliant amount of documentation that is available for chrome driver.

If you are lost there, just follow this [***link***](http://chromedriver.storage.googleapis.com/index.html?path=2.20/)to directly download the chrome driver.



Download the zip file based on the OS platform that you have. Unzip the zip file and keep it somewhere on a know location on your PC. Now lets see how we can run tests in a Chrome Driver.

## Launching Chrome Browser using Selenium WebDriver

Launching a Chrome driver is easy as launching any other driver.

***WebDriver driver = new ChromeDriver();***

But, the chrome driver executable must be set in to you machine system environments or it should be explicitly set in the code. If not set, you would face the below exception:

### ***Exception : The path to the driver executable must be set by the webdriver.chrome.driver system property***

## Option 1: Set System Properties Explicitly

In order to launch Chrome browser we have to do two steps

1. Set a system property “***webdriver.chrome.driver***” to the path of your ***ChromeDriver.exe*** file
2. ***Instantiate*** a ***ChromeDriver*** class.

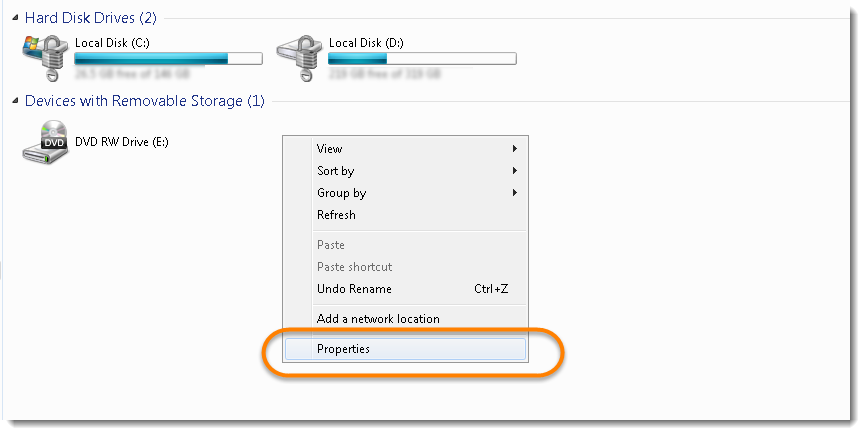
Here is a sample code to do that:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public class LaunchingChrome {  public static void main(String[] args) {  String exePath = "C:\\Users\\abc\\Desktop\\Server\\chromedriver.exe";  System.setProperty("webdriver.chrome.driver", exePath);  WebDriver driver = new ChromeDriver();  driver.get("http://toolsqa.wpengine.com/automation-practice-form/");  }  } |

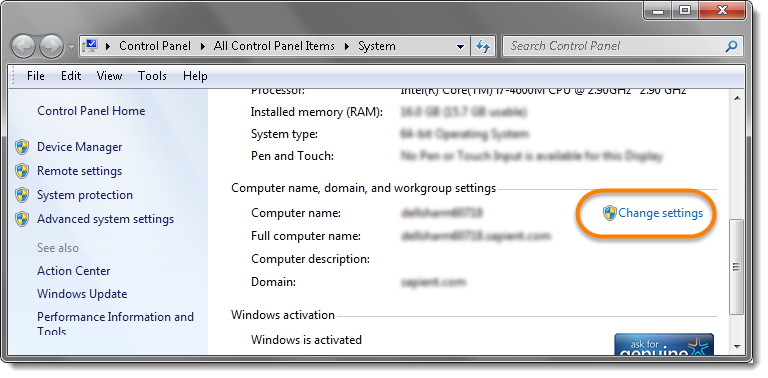
So that’s it, now we can write any test that we want using chrome driver.

## Option 2: Set property in Environment Variables:-

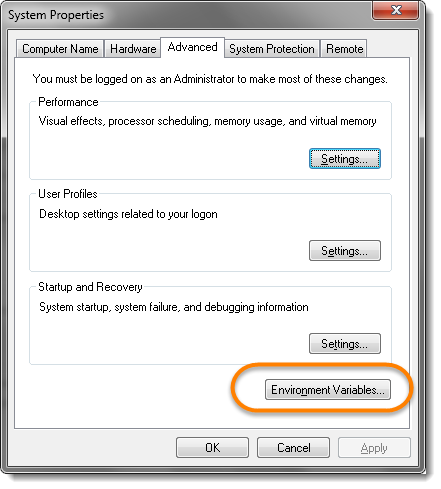
1. Go to ***My Computer*** and ***Right click*** to get the context menu.



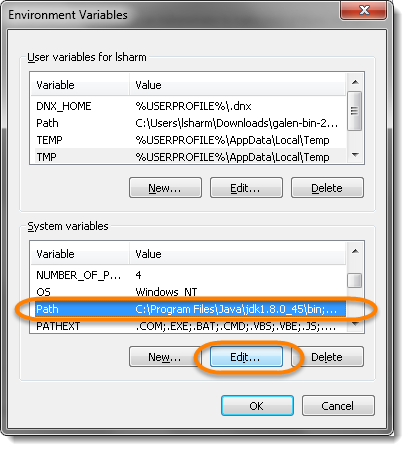
2. Click on the ***Change Settings*** on the opened window.



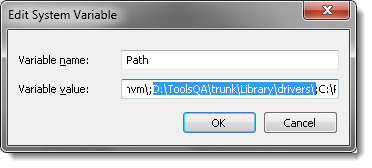
3. Go to ***Advance*** tab in the System Properties window and click on ***Environment Variables***.



4. Now under the System variables, select ***Path*** and click on ***Edit***.



5. At the end of the string use ***semicolon*** and paste the path of the ChromeDriver. On my machine my ChromeDriver exe resides in ***D:\ToolsQA\trunk\Library|drivers\***



***Note***: Once the path is set, you would not need to set the System property every time in the test script. Your test script would simply work without the System Property code.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | public class LaunchingChrome {    public static void main(String[] args) {  WebDriver driver = new ChromeDriver();  driver.get("http://toolsqa.wpengine.com/automation-practice-form/");  }  } |

### ***Issues***

***org.openqa.selenium.WebDriverException: java.net.SocketException: Connection reset***

Most of the time this comes when there is a version mismatch of Desktop Chrome version and ChromeDriver.exe. Just make sure that you are using the latest Chrome Browser version and ChromeDriver.exe. Look for the latest [**here**](https://chromedriver.storage.googleapis.com/index.html) or upgrade your chrome browser version. Or you may be try degrading the ChromeDriver version to old earlier version in case you are already on latest. Also to make sure to delete the existing ChromeDriver from the path or change the name.

# WebDriver Browser Commands

The very first question which comes to my mind and has been asked in many interviews is ***What is Selenium WebDriver***? Is it an Automation Tool? Is it a Class? Is it an Interface or what actually it is? To answer this question we need to understand the ***Advance Java OOPs concepts*** first and then we would be able to visualise the ***WebDriver Implementation***. For the sake of simplicity, we will avoid this WebDriver Implementation topic for now and will cover this in later chapters. As of now we start with all the methods we get from WebDriver.

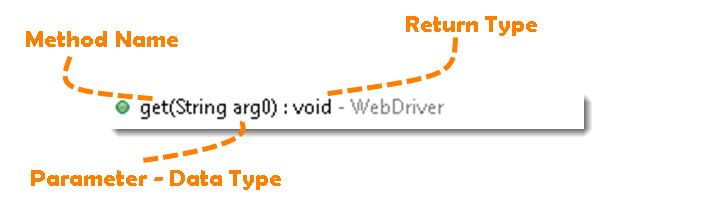
Now the next question is, How to access the methods of WebDriver? To check what all we have in WebDriver, create a driver object from WebDriver and press ***dot key.*** This will list down all the methods of WebDriver.



***Note:*** Methods followed by ***Object*** keyword are the generic methods gets from Object Class in Java. You will find these method for every object of java language.

* The suggestions marked in ***Blue Color*** are Nested Classes under WebDriver and will be covered in detail separately in the following chapters.
* The suggestions marked in ***Green Color*** are also Interfaces like WebDriver and will be covered in detail separately in the following chapters.
* The suggestions marked in ***Violet Color*** are similar methods like ***Orange*** but will be covered in detail separately in the following chapters.

Let’s just start discussing the ***Orange colored*** methods of ***Selenium WebDriver***but before that try to understand the syntax of the suggestions display by Eclipse for WebDriver.



**Method:** A Java method is a collection of statements that are grouped together to perform an operation.

* ***Method Name:*** To access any method of any class, we need to create an object of class and then all the public methods will appear for the object.
* ***Parameter:*** It is an argument which is passed to a method as a parameter to perform some operation. Every argument must passed with the same data type. For e.g. ***get(String arg0) : void.***This is asking for a ***String type***argument.
* ***Return Type:*** Method can returns a value or returning nothing (void). If the ***void*** is mentioned after the method, it means the method is returning no value. And if it is returning any value, then it must display the type of the value for e.g. ***getTitle() : String***.

Now it would be very easy to understand the WebDriver commands in the below chapter. The very first thing you like to do with Selenium is to ***Opening*** a new browser, ***Perform*** few tasks and ***Closing*** the browser. Below are the numbers of commands you can apply on the Selenium opened browser.

## Get Command

***get(String arg0) : void*** – This method ***Load***a new web page in the current browser window. Accepts String as a parameter and returns nothing.

**Command**– ***driver.get(appUrl);***

Where **appUrl**is the website address to load. It is best to use a fully qualified URL.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | driver.get("http://www.google.com");    //Or can be written as    String URL = "http://www.DemoQA.com";  driver.get(URL); |

## Get Title Command

***getTitle() : String*** – This method fetches the ***Title*** of the current page. Accepts nothing as a parameter and returns a String value.

***Command – driver.getTitle();***

As the return type is String value, the output must be stored in String object/variable.

|  |  |
| --- | --- |
| 1  2  3  4  5 | driver.getTitle();    //Or can be used as    String Title = driver.getTitle(); |

## Get Current URL Command

***getCurrentUrl() : String*** – This method fetches the string representing the ***Current URL*** which is opened in the browser. Accepts nothing as a parameter and returns a String value.

***Command – driver.getCurrentTitle();***

As the return type is String value, the output must be stored in String object/variable.

|  |  |
| --- | --- |
| 1  2  3  4  5 | driver.getCurrentUrl();    //Or can be written as    String CurrentUrl = driver.getCurrentUrl(); |

## Get Page Source Command

***getPageSource() : String*** – This method returns the ***Source Code***of the page. Accepts nothing as a parameter and returns a String value.

***Command – driver.getPageSource();***

As the return type is String value, the output must be stored in String object/variable.

|  |  |
| --- | --- |
| 1  2  3  4 | driver.getPageSource();    //Or can be written as  String PageSource = driver.getPageSource(); |

## Close Command

***close() : void*** – This method **Close** only the current window the WebDriver is currently controlling. Accepts nothing as a parameter and returns nothing.

***Command – driver.close();***

Quit the browser if it’s the last window currently open.

|  |  |
| --- | --- |
| 1 | driver.close(); |

## Quit Command

***quit() : void*** – This method **Closes** all windows opened by the WebDriver. Accepts nothing as a parameter and returns nothing.

***Command – driver.quit();***

Close every associated window.

|  |  |
| --- | --- |
| 1 | driver.quit(); |

### Practice Exercise – 1

1. Launch a new Firefox browser.
2. Open Store.DemoQA.com
3. Get Page Title name and Title length
4. Print Page Title and Title length on the Eclipse Console.
5. Get Page URL and verify if the it is a correct page opened
6. Get Page Source (HTML Source code) and Page Source length
7. Print Page Length on Eclipse Console.
8. Close the Browser.

Solution

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | package automationFramework;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.firefox.FirefoxDriver;  public class WebDriverCommands {    public static void main(String[] args) {  // Create a new instance of the FireFox driver  WebDriver driver = new FirefoxDriver();    // Storing the Application Url in the String variable  String url = "http://www.store.demoqa.com";    //Launch the ToolsQA WebSite  driver.get(url);    // Storing Title name in the String variable  String title = driver.getTitle();    // Storing Title length in the Int variable  int titleLength = driver.getTitle().length();    // Printing Title & Title length in the Console window  System.out.println("Title of the page is : " + title);  System.out.println("Length of the title is : "+ titleLength);    // Storing URL in String variable  String actualUrl = driver.getCurrentUrl();    if (actualUrl.equals(url)){  System.out.println("Verification Successful - The correct Url is opened.");  }else{  System.out.println("Verification Failed - An incorrect Url is opened.");  //In case of Fail, you like to print the actual and expected URL for the record purpose  System.out.println("Actual URL is : " + actualUrl);  System.out.println("Expected URL is : " + url);  }    // Storing Page Source in String variable  String pageSource = driver.getPageSource();    // Storing Page Source length in Int variable  int pageSourceLength = pageSource.length();    // Printing length of the Page Source on console  System.out.println("Total length of the Pgae Source is : " + pageSourceLength);    //Closing browser  driver.close();  }  } |

***Output***

Title of the page is : ONLINE STORE | Toolsqa Dummy Test site  
Length of the title is : 38  
Verification Failed – An incorrect Url is opened.  
Actual URL is : http://store.demoqa.com/  
Expected URL is : http://www.store.demoqa.com  
Total length of the Pgae Source is : 35646

### Practice Exercise – 2

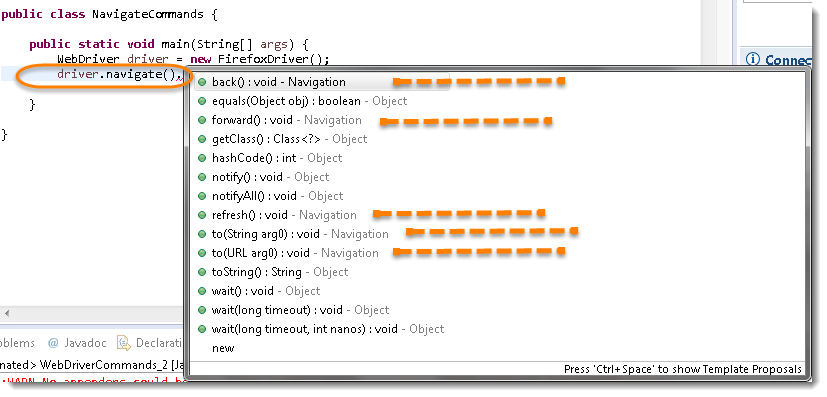
1. Launch a new Firefox browser.
2. Open http://demoqa.com/frames-and-windows/
3. Use this statement to click on a New Window button “driver.findElement(By.xpath(“.//\*[@id=’tabs-1′]/div/p/a”)).click();”
4. Close the browser using close() command

You will notice that only one window will close. Next time use quit() command instead of close(). At that time selenium will close both the windows.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | package automationFramework;  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.firefox.FirefoxDriver;    public class WebDriverCommands\_2 {  public static void main(String[] args) {  WebDriver driver = new FirefoxDriver();  driver.get("http://demoqa.com/frames-and-windows/");  driver.findElement(By.xpath(".//\*[@id='tabs-1']/div/p/a")).click();  driver.close();  }  } |

# Browser Navigation Commands

After successfully running our first test case on Firefox Browser now we are stepping towards grasping the essential ***Browser Navigation Commands*** in Selenium. Thus we are going to discuss about various navigation commands that we would be using in our day to day automation testing. The navigate interface exposes the ability to move backwards and forwards in the browser’s history.

To access the navigation’s method, just type ***driver.navigate().***. The intellisence feature of eclipse will automatically display all the public methods of **Navigate Interface**shown in the below image.  


***Note:*** Only methods which are followed by ***Navigation*** keyword are belongs to navigate. Rest followed by ***Object***keyword are the generic methods gets from Object Class in Java. You will find these method for every object of java language.

## Navigate To Command

***to(String arg0) : void*** – This method ***Loads*** a new web page in the current browser window. It accepts a String parameter and returns nothing.

**Command** – ***driver.navigate().to(appUrl);***

It does exactly the same thing as the ***driver.get(appUrl)*** method. Where **appUrl**is the website address to load. It is best to use a fully qualified URL.

|  |  |
| --- | --- |
| 1 | driver.navigate().to("http://www.DemoQA.com"); |

## Forward Command

***forward() : void*** – This method does the same operation as clicking on the ***Forward Button*** of any browser. It neither accepts nor returns anything.

**Command** – ***driver.navigate().forward();***

Takes you forward by one page on the browser’s history.

|  |  |
| --- | --- |
| 1 | driver.navigate().forward(); |

## Back Command

***back() : void*** – This method does the same operation as clicking on the ***Back Button*** of any browser. It neither accepts nor returns anything.

**Command** – ***driver.navigate().back();***

Takes youback by one page on the browser’s history.

|  |  |
| --- | --- |
| 1 | driver.navigate().back(); |

## Refresh Command

***refresh() : void*** – This method ***Refresh*** the current page. It neither accepts nor returns anything.

**Command** – ***driver.navigate().refresh();***

Perform the same function as pressing F5 in the browser.

|  |  |
| --- | --- |
| 1 | driver.navigate().refresh(); |

### Practice Exercise

1. Launch new Browser
2. Open DemoQA.com website
3. Click on Registration link using “driver.findElement(By.*xpath*(“.//\*[@id=’menu-item-374′]/a”)).click();“
4. Come back to Home page (Use ‘Back’ command)
5. Again go back to Registration page (This time use ‘Forward’ command)
6. Again come back to Home page (This time use ‘To’ command)
7. Refresh the Browser (Use ‘Refresh’ command)
8. Close the Browser

Solution

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | package automationFramework;  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.firefox.FirefoxDriver;  public class NavigateCommands {  public static void main(String[] args) {  // Create a new instance of the FireFox driver  WebDriver driver = new FirefoxDriver();    // Open ToolsQA web site  String appUrl = "http://www.DemoQA.com";  driver.get(appUrl);    // Click on Registration link  driver.findElement(By.xpath(".//\*[@id='menu-item-374']/a")).click();    // Go back to Home Page  driver.navigate().back();    // Go forward to Registration page  driver.navigate().forward();    // Go back to Home page  driver.navigate().to(appUrl);    // Refresh browser  driver.navigate().refresh();    // Close browser  driver.close();  }  } |