**Selenium1**

**What Is Selenium?**

Selenium is a free automated testing suite used to automate web applications across different browsers and platforms. It supports various programming languages like Java, Dot Net, PHP, Python, Perl, Ruby etc and various browsers like Mozilla Firefox, Google Chrome, Safari, Internet Explorer etc.

The major four components of Selenium are:

* Selenium IDE
* Selenium RC
* WebDriver
* Selenium Grid



**Selenium IDE**

Selenium IDE is nothing just a Mozilla Firefox add-on that allows recording, editing and debugging tests. It was previously known as Selenium Recorder. It is able to record tests as well as playback them in the same plugin. Selenium IDE will not work after the Firefox version 54 as its support is limited to Firefox version 54.

**Selenium RC**

Selenium Remote Control (RC) is a server that accepts commands for the browser via HTTP. It solves the limitation of Selenium IDE and supports various programming languages.

Using Selenium IDE, we can record and run the script but only in the Firefox browser. But using Selenium RC we can run the same recorded script in any browser but we have to start and stop the server to execute the test scripts.

**Selenium Grid**

Selenium Grid is a testing tool which allows us to run our test scripts on different OS and on different browsers. It is a part of the Selenium Suite which specializes in running multiple tests across different browsers, operating system, and machines. With Selenium Grid, one server acts as the hub and others act as a node.

In Selenium Grid, the hub is a computer which is the central point where we can load our tests into. Hub also acts as a server because of which it acts as a central point to control the network of Test machines. A node can be referred as a test machine which opts to connect with the hub.

**Selenium Webdriver**

Selenium WebDriver is one of the most powerful and popular tools of Selenium toolkit. Unlike Selenium IDE, WebDriver allows you to execute your tests against different browsers.

It is an extended version of Selenium RC. It aims to provide a friendly API which is easy to explore and understand, easier to use than the Selenium-RC API, which helps to make your tests scripts easier to read and maintain.

Basically, they used to remove the server part from the WebDriver so that the performance does not emerge as an issue in the WebDriver. It means simply write your code and it will directly communicate with the browsers.

**Architecture of selenium Webdriver 3**



As mentioned clearly in the above image, Firefox driver (and other browser’s drivers) extends Remote WebDriver class and the Remote WebDriver class implements the WebDriver interface.

FirefoxDriver is a term of class that has been written or designed specifically for the Firefox browser. It includes the use of methods that are implemented and can be instantiated further. It can perform all the methods on the Firefox browser as defined in the interface WebDriver.

Remote WebDriver is an implementation class of the WebDriver interface that an automation test engineer can use to execute their test scripts via the Remote WebDriver server on a remote machine.

WebDriver is an interface provided by Selenium WebDriver. As we know that interfaces in Java are the collection of constants and abstract methods (methods without any implementation). The WebDriver interface serves as a contract that each browser specific implementation like ChromeDriver, FireFoxDriver must follow.

Selenium WebDriver architecture consists of 3 layers:

* Language Binding
* WebDriver API
* Browser Drivers

**Language Binding**

Language Bindings was developed to support multiple languages.

This term can be defined as a bunch of languages which are developing a framework, interacting with the Selenium WebDriver and working on various browsers and other devices.

Suppose, if you want to use the browser driver in Java, so you need to use the Java bindings for Selenium WebDriver. If you want to use the browser driver for C#, Ruby, Python etc then use the binding for that language.

**Selenium WebDriver API**

This API is a medium of communication with programming languages and browsers.

This API sends the commands taken from language level bindings interpret it and sent it to respective driver. In short, WebDriver API has a common library which sends commands to the respective drivers.

**Browser Drivers**

Browser drivers help in communication with the browser. Drivers receive commands directly from the Server and performing the actions on different browsers as per different commands using the remote WebDriver. We have so many browsers like Mozilla Firefox Browser, Chrome Browser, Opera Browser and IE Browser etc. Each browser will contain separate driver and each driver knows how to drive the browser that it corresponds to.

For example, Chrome driver knows how to handle the details of Chrome browser and drive it to do things like clicking the button, going into pages, getting data from the browser itself, the same thing for Firefox, IE, and so on.

**Pros & Cons of Selenium WebDriver**

Everything is loaded with some pros and cons, so let’s dig into the advantages and disadvantages associated with the same…

**Advantages:**

1. Selenium is an open source, freeware, and portable tool

2. Selenium supports various languages that include Java, Perl, Python, C# etc

3. Selenium supports many operating systems like Windows, Macintosh, Linux, Unix etc

4. Selenium supports many browsers like Internet Explorer, Chrome, Firefox, Opera, Safari etc

5. WebDriver is faster, as compared to RC

6. Unlike RC you don’t have to start a server in WebDriver

7. You can simulate the movement of a mouse using selenium

8. It allows you to simulate keyboard key press events using different classes

9. Find coordinates of any object easily using WebDriver

10. Integration with the testing framework like JUnit or TestNG is very easy with the WebDriver

**Disadvantages:**

1. Selenium does not provide any built-in IDE for script generation and it needs other IDE like Eclipse for writing scripts.

2. It supports Web-based Applications only.

3. We can’t automate audio & video related test cases by Selenium WebDriver.

4. No built-in Reporting facility.

**How to download and configure WebDriver-3 in eclipse**

[**http://toolsqa.com/selenium-webdriver/configure-eclipse-with-selenium-webdriver/**](http://toolsqa.com/selenium-webdriver/configure-eclipse-with-selenium-webdriver/)

[**https://www.guru99.com/installing-selenium-webdriver.html**](https://www.guru99.com/installing-selenium-webdriver.html)

/\*

To set Javadoc and Source attachment for Selenium

1) Right click on Project Folder

2) Click Properties

3) Click Java Build Path

4) Select Libraries tab

5) Select Selenium-java-3.9.1 library

6) Click Edit

7) Select Selenium-java-3.9.1 library

8) Click user libraries

9) Expand client-combined-3.9.1.jar

10) Select Source Attachment and click Edit

11) Select External Locaton and provide path for "client-combined-3.9.1-sources.jar" and click ok

12 ) Select Javadoc location and click Edit

Update below URL in Javadoc location path

https://seleniumhq.github.io/selenium/docs/api/java/

12) Click ok

13) Click Apply and Close

14) Click Finish

15) Click Apply and Close

\*/

// If working with Firefox below are the prerequisites

// 1) Keep old FF version by 1 or 2. Turn off automatic updates..as Selenium version 3.9.1 may not support latest FF version

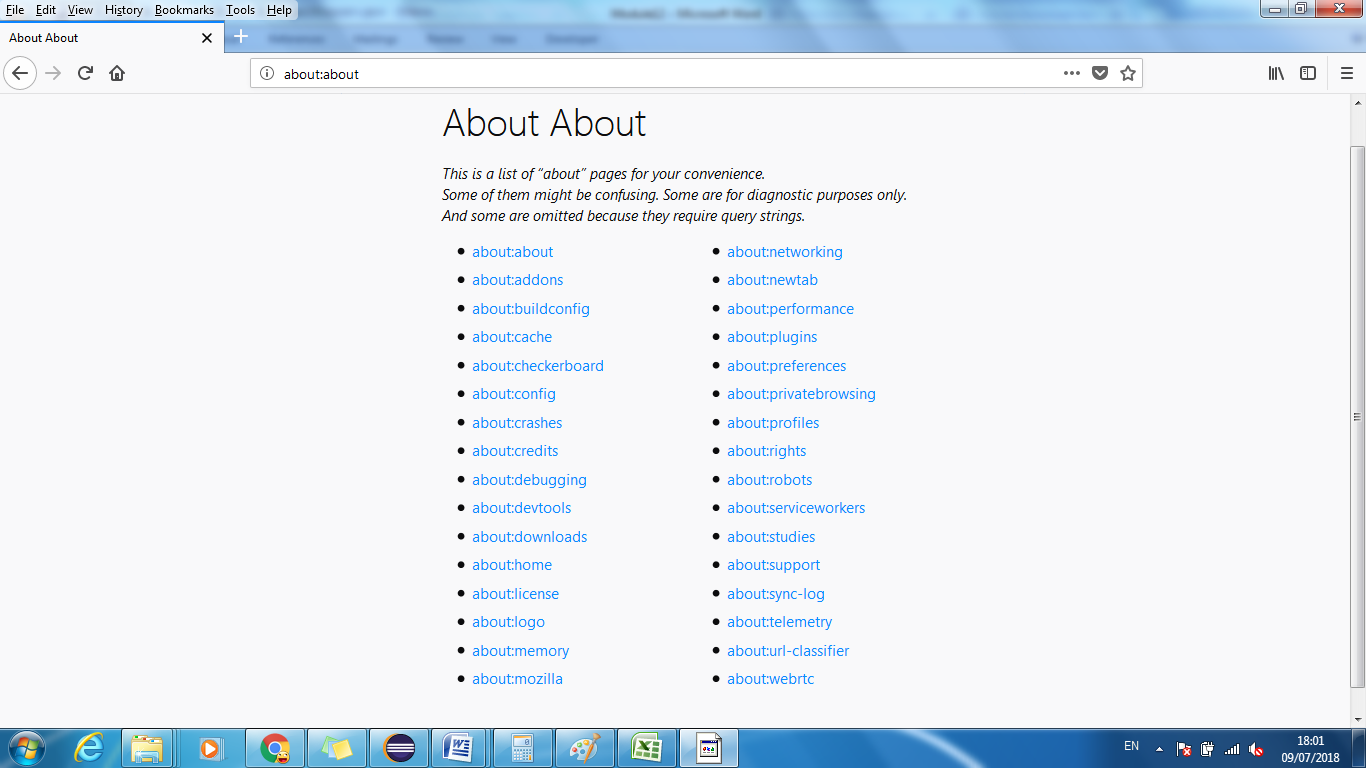
// In FF goto Tools->Options-> Click Never check for updates

// Link for older version of FF - https://ftp.mozilla.org/pub/firefox/releases/

**FireFox Drivers**

It controls the [Firefox](http://getfirefox.com/) browser. When we instantiate a Firefox browser using Selenium, the extensions added to our normal Firefox browser are always missing from the FireFoxDriver because extensions are never loaded by default in *FireFoxDriver*. A few settings are also changed by default ([see the source to see which ones](https://github.com/SeleniumHQ/selenium/blob/master/javascript/firefox-driver/webdriver.json))

Open Firefox and type about:about in URL bar. Below screen will appear. Select about:config, it will show all the above Preference Name along with others will be shown.

****

The following are the system properties (read using System.getProperty() and set using System.setProperty() in Java code

Note: In order to get below values…check javadocs of FirefoxDriver.SystemProperty in Eclipse by double clicking on SystemProperty

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Firefox System Property** | **Constant Name** | **Definition** |
| 1 | FirefoxDriver.SystemProperty.BROWSER\_BINARY | "webdriver.firefox.bin" | System property that defines the location of the Firefox executable file. |
| 2 | FirefoxDriver.SystemProperty.BROWSER\_LIBRARY\_PATH | "webdriver.firefox.library.path" | System property that defines the additional library path (Linux only). |
| 3 | FirefoxDriver.SystemProperty.BROWSER\_LOGFILE | "webdriver.firefox.logfile" | System property that defines the location of the file where Firefox log should be stored. |
| 4 | FirefoxDriver.SystemProperty.BROWSER\_PROFILE | "webdriver.firefox.profile" | System property that defines the profile that should be used as a template. When the driver starts, it will make a copy of the profile it is using, rather than using that profile directly. |
| 5 | FirefoxDriver.SystemProperty.DRIVER\_USE\_MARIONETTE | "webdriver.firefox.marionette" | Boolean system property that instructs FirefoxDriver to use Marionette backend, overrides any capabilities specified by the user |
| 6 | FirefoxDriver.SystemProperty.DRIVER\_XPI\_PROPERTY | "webdriver.firefox.driver" | System property that defines the location of the webdriver.xpi browser extension to install in the browser. If not set, the prebuilt extension bundled with this class will be used. |

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**public** **class** SeleniumSetGetSystemProperties {

**public** **static** **void** main(String[] args) {

System.***out***.println("Setting FirefoxDriver System Property...");

System.***out***.println();

System.*setProperty*(FirefoxDriver.SystemProperty.***BROWSER\_BINARY***,

"C:\\Program Files (x86)\\Mozilla Firefox\\firefox.exe");

System.*setProperty*(FirefoxDriver.SystemProperty.***BROWSER\_LIBRARY\_PATH***, "www");

System.*setProperty*(FirefoxDriver.SystemProperty.***BROWSER\_LOGFILE***, "d:\\system.log");

// System.setProperty(FirefoxDriver.SystemProperty.BROWSER\_LOGFILE, "NULL"); // If you do not want any logs to be created by Selenium

// System.setProperty(FirefoxDriver.SystemProperty.BROWSER\_PROFILE, "Jigar");

System.*setProperty*(FirefoxDriver.SystemProperty.***DRIVER\_USE\_MARIONETTE***, "ttt");

// System.setProperty(FirefoxDriver.SystemProperty.DRIVER\_XPI\_PROPERTY, "yyy");

System.***out***.println("Getting FirefoxDriver System Property...");

System.***out***.println(System.*getProperty*(FirefoxDriver.SystemProperty.***BROWSER\_BINARY***));

System.***out***.println(System.*getProperty*(FirefoxDriver.SystemProperty.***BROWSER\_LIBRARY\_PATH***));

System.***out***.println(System.*getProperty*(FirefoxDriver.SystemProperty.***BROWSER\_LOGFILE***));

System.***out***.println(System.*getProperty*(FirefoxDriver.SystemProperty.***BROWSER\_PROFILE***));

System.***out***.println(System.*getProperty*(FirefoxDriver.SystemProperty.***DRIVER\_USE\_MARIONETTE***));

System.***out***.println(System.*getProperty*(FirefoxDriver.SystemProperty.***DRIVER\_XPI\_PROPERTY***));

/\* System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Setting FirefoxDriver System Property...");

System.out.println();

System.setProperty("webdriver.firefox.bin", "aaaaaa");

System.setProperty("webdriver.firefox.library.path", "ssssss");

System.setProperty("webdriver.firefox.logfile", "dddddd");

System.setProperty("webdriver.firefox.profile", "ffffff");

System.setProperty("webdriver.firefox.marionette", "gggggg");

System.setProperty("webdriver.firefox.driver", "hhhhhh");

\*/

System.***out***.println("Getting FirefoxDriver System Property...");

System.***out***.println("BROWSER\_BINARY = " + System.*getProperty*(FirefoxDriver.SystemProperty.***BROWSER\_BINARY***));

System.***out***.println("BROWSER\_LIBRARY\_PATH = " + System.*getProperty*(FirefoxDriver.SystemProperty.***BROWSER\_LIBRARY\_PATH***));

System.***out***.println("BROWSER\_LOGFILE = " + System.*getProperty*(FirefoxDriver.SystemProperty.***BROWSER\_LOGFILE***));

System.***out***.println("BROWSER\_PROFILE = " + System.*getProperty*(FirefoxDriver.SystemProperty.***BROWSER\_PROFILE***));

System.***out***.println("DRIVER\_USE\_MARIONETTE = " + System.*getProperty*(FirefoxDriver.SystemProperty.***DRIVER\_USE\_MARIONETTE***));

System.***out***.println("DRIVER\_XPI\_PROPERTY = " + System.*getProperty*(FirefoxDriver.SystemProperty.***DRIVER\_XPI\_PROPERTY***));

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

}

}

Normally the Firefox binary is assumed to be in the default location for your particular operating system:

|  |  |
| --- | --- |
| OS | Expected Location of Firefox |
| Linux | firefox (found using "which") |
| Mac | /Applications/Firefox.app/Contents/MacOS/firefox-bin |
| Windows | %PROGRAMFILES%\Mozilla Firefox\firefox.exe |

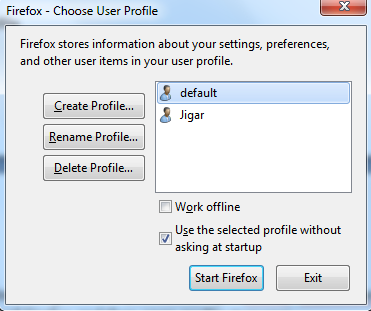
**By default, the Firefox driver creates an anonymous profile**

**Creating\Renaming\Deleting Firefox Profile manually from Run window**

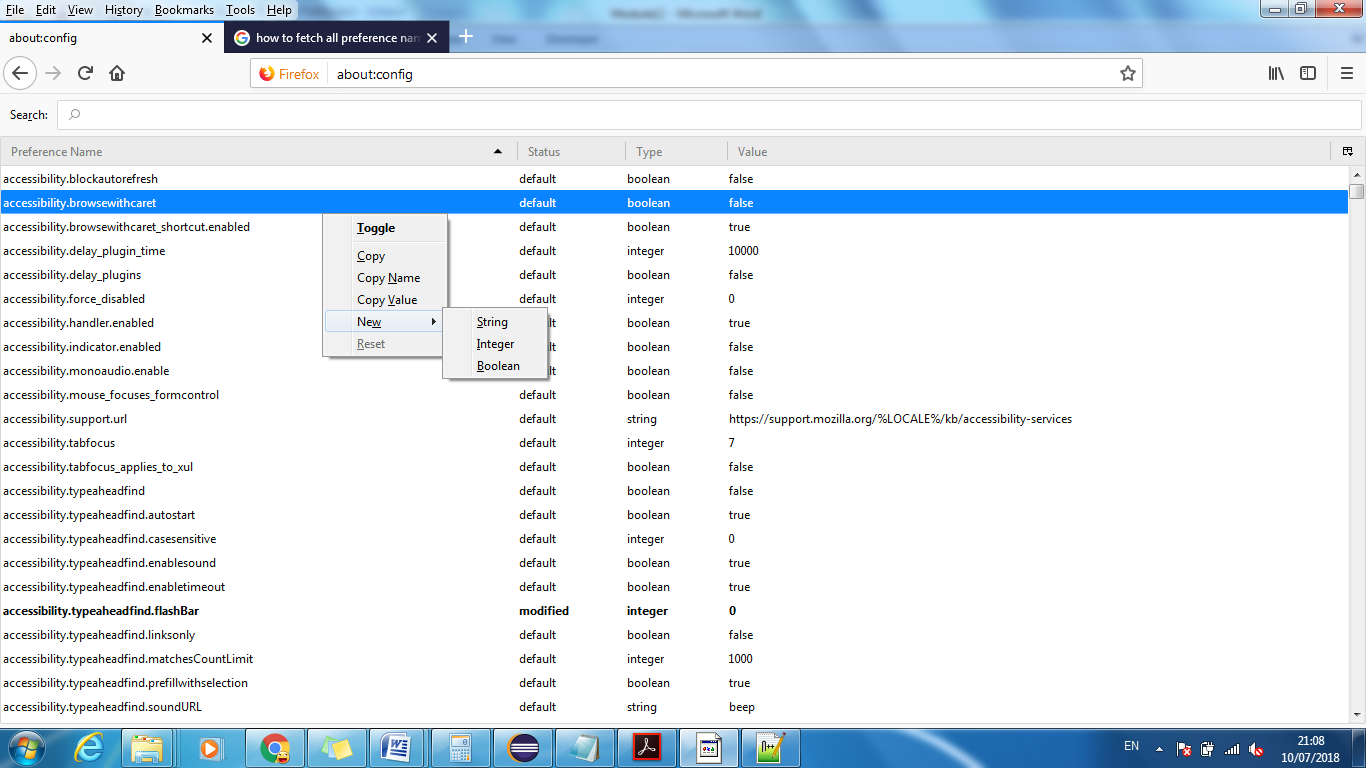
Firefox profile can be created using Firefox’s profile manager by running following command in Run command of windows. Quit all instances of Firefox and then open profile manager

* firefox –ProfileManager

Below window appears. Add a profile ‘Jigar’ by clicking ‘Create Profile’ button and following the instruction.

****

1. Open Firefox with ‘Jigar’ profile manually.
2. Type about:config in address bar and press enter. It will list all the preferences available.



**How to get and set preference name using selenium?**

**Preference names are listed in about:config in Firefox**

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.openqa.selenium.firefox.FirefoxOptions;

**import** org.openqa.selenium.firefox.FirefoxProfile;

**import** org.openqa.selenium.firefox.GeckoDriverService;

**import** org.openqa.selenium.firefox.internal.ProfilesIni;

**public** **class** Firefox\_Profile {

**public** **static** **void** main(String[] args) {

// System property that defines the location of the GeckoDriver executable. Use

// one of the below

// we can avoid setting below setproperty by updating the geckodriver.exe path in path variable

System.*setProperty*(GeckoDriverService.***GECKO\_DRIVER\_EXE\_PROPERTY***, "D:\\SeleniumDrivers\\geckodriver.exe");

System.*setProperty*("webdriver.gecko.driver", "D:\\SeleniumDrivers\\geckodriver.exe");

ProfilesIni allProfiles = **new** ProfilesIni();

FirefoxProfile profile = allProfiles.getProfile("Jigar");

System.***out***.println(profile.getStringPreference("Test1", "No such preference name exist"));

profile.setPreference("Test1", "Hello World");

System.***out***.println(profile.getStringPreference("Test1", "No such preference name exist"));

FirefoxOptions fo = **new** FirefoxOptions().setProfile(profile);

WebDriver driver = **new** FirefoxDriver(fo);

}

}

**FireFox Options**

**import** java.util.concurrent.TimeUnit;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.PageLoadStrategy;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.openqa.selenium.firefox.FirefoxOptions;

**import** org.openqa.selenium.firefox.FirefoxProfile;

**import** org.openqa.selenium.firefox.internal.ProfilesIni;

// Options Classes

**public** **class** Firefox\_Options {

**public** **static** **void** main(String[] args) {

// pre requisites

ProfilesIni allprofiles = **new** ProfilesIni(); // all Firefox profiles on your PC

FirefoxProfile firefoxfprofile = allprofiles.getProfile("Jigar"); // Fetch and pass profile "Jigar"

FirefoxOptions ffo = **new** FirefoxOptions();

ffo.setProfile(firefoxfprofile);

WebDriver driver = **new** FirefoxDriver(ffo);

driver.get("https://login.yahoo.com/");

driver.findElement(By.*id*("login-username")).sendKeys("helllo");

// notifications

firefoxfprofile.setPreference("dom.webnotifications.enabled", **false**); // turn off notifications -

// certificate - get rid certificate error

firefoxfprofile.setAcceptUntrustedCertificates(**true**);

firefoxfprofile.setAssumeUntrustedCertificateIssuer(**false**);

// page load strategy

ffo.setPageLoadStrategy(PageLoadStrategy.***NONE***); // None - Type as soon as possible

// proxy

firefoxfprofile.setPreference("network.proxy.type", 1);

firefoxfprofile.setPreference("network.proxy.socks", "10.10.100.84");

firefoxfprofile.setPreference("network.proxy.socks\_port", 8080);

firefoxfprofile.setPreference("browser.startup.homepage", "https://login.yahoo.com/");

ffo.setProfile(firefoxfprofile);

// after copying code from one project to another..to import relevant packages

// use ctrl+shit+o (Source->Organize Imports) \*/

// WebDriver driver = new FirefoxDriver(ffo);

driver.manage().timeouts().implicitlyWait(10, TimeUnit.***SECONDS***);

driver.get("https://login.yahoo.com/");

driver.findElement(By.*id*("login-username")).sendKeys("helllo");

}

}

**Firefox Profile**

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

public class Firefox\_Profile {

public static void main(String[] args) {

System.setProperty(FirefoxDriver.PROFILE, "C:\\Users\\jigar.mehta1\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles\\z2i8dtfs.Jigar");

WebDriver driver = new FirefoxDriver();

driver.get("https://login.yahoo.com/");

}

}

**Basic Drivers**

**import** java.util.Properties;

**import** java.util.concurrent.TimeUnit;

**import** org.apache.http.impl.conn.tsccm.~~WaitingThread~~;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.openqa.selenium.chrome.ChromeDriverService;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.openqa.selenium.ie.InternetExplorerDriver;

**import** org.openqa.selenium.ie.InternetExplorerDriverService;

**public** **class** BasicDrivers {

**public** **static** **void** main(String[] args) **throws** InterruptedException {

System.*setProperty*("webdriver.chrome.driver","D:\\SeleniumDrivers\\chromedriver.exe");

// System.setProperty(ChromeDriverService.CHROME\_DRIVER\_EXE\_PROPERTY, "D:\\SeleniumDrivers\\chromedriver.exe");

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

driver.navigate().to("https://www.google.co.in");

driver.manage().window().maximize();

// 1) Download IEDriverServer from seleniumhq.com

// 2) Keep Zoom level settings to 100%

// 3) Disable the 'Enable Protected Mode' in Tools->Internet Options->Security and level to least positions

// 4) Enable 'Allow active content to run in files on My Computer' in Tools->Internet Options->Advanced

// 5) Should have IE11..download IE drivers accordingly if its 32 bit or 64 bit

// we can avoid setting setproperty by updating the IEDriverServer path in path variable

// System.setProperty("webdriver.ie.driver", "D:\\SeleniumDrivers\\IEDriverServer.exe");

// System.setProperty(InternetExplorerDriverService.IE\_DRIVER\_EXE\_PROPERTY, "D:\\IEDriverServer\\IEDriverServer.exe");

// InternetExplorerDriver driver = new InternetExplorerDriver();

// driver.navigate().to("https://in.yahoo.com/?p=us");

// Thread.sleep(8000);

// driver.close(); // closes browser which has the focus but not selenium session

// driver.quit(); //closes driver exe and browser and selenium session...good practise to use .quit at the end of program

// Make sure the MS Edge Driver exe version is same as the Edge version

// driver.navigate().to("https://mail.rediff.com/cgi-bin/login.cgi");

driver.manage().timeouts().implicitlyWait(10, TimeUnit.***SECONDS***);

driver.get("https://mail.rediff.com/cgi-bin/login.cgi"); // Its important to provide protocol in URL e.g. www.rediffmail.com may not work

driver.findElement(By.*id*("login1")).sendKeys("hellooooo");

}

}

**Chrome Options**

**import** org.openqa.selenium.PageLoadStrategy;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.openqa.selenium.chrome.ChromeDriverService;

**import** org.openqa.selenium.chrome.ChromeOptions;

**public** **class** Chrome\_Options {

**public** **static** **void** main(String[] args) {

// pre-requisites - Chrome should be up to date

//http://www.seleniumeasy.com/selenium-tutorials/using-chrome-options-for-webdriver-tests

// Below link will give list of all Chrome command line switch

// https://peter.sh/experiments/chromium-command-line-switches/#load-extension

ChromeOptions ops = **new** ChromeOptions();

// Logs

System.*setProperty*(ChromeDriverService.***CHROME\_DRIVER\_LOG\_PROPERTY***, "D:\\Chrome.log");

// System.setProperty(ChromeDriverService.CHROME\_DRIVER\_SILENT\_OUTPUT\_PROPERTY, "false"); // set to false if you want logs to be shown in console else set to true

// Notification, info bars and maximized

// ops.addArguments("--disable-notifications");

// ops.addArguments("disable-infobars");

ops.addArguments("--start-maximized");

// ops.addArguments("--proxy-server=http://10.99.99.99:8080");

// ops.addArguments("user-data-dir=C:\\Users\\jigar.mehta1\\AppData\\Local\\Google\\Chrome\\User Data\\"); //Dont mention "Default" in the end of the path

ops.addArguments("user-data-dir=C:\\Users\\jigar.mehta1\\AppData\\Local\\Google\\Chrome\\User Data\\Profile 1\\");

// page load strategy

ops.setPageLoadStrategy(PageLoadStrategy.***EAGER***);

// binary

ops.setBinary("path of chrome exe");

// proxy

// profiling

// certificate

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

driver.get("https://login.yahoo.com/");

// driver.close();

// driver.quit();

}

}