**Selenium Class**

# Chapter#3 – Launching Browsers in Selenium

1. **Understanding the browser launching command**
2. **Launching Firefox Browser**
3. **Launching Chrome Browser**
4. **Launching Internet Explorer Browser**
5. **Launching Safari Browser**

## Understanding the browser launching command

As we have studied in previous tutorials that Selenium WebDriver calls the native methods of the different browsers to automate them. Hence, in Selenium we have different WebDrivers for different browsers like - FirefoxDriver for Firefox browser, ChromeDriver for Google Chrome, InternetExplorerDriver for Internet Explorer, etc. Now let’s take an example of launching a Firefox browser and understand the command in detail.

WebDriver driver = new FirefoxDriver();

This is the java implementation of launching a browser in Selenium. Here, ‘WebDriver’ is an interface and we are creating a reference variable ‘driver’ of type WebDriver, instantiated using ‘FirefoxDriver’ class.

For those who are not very proficient in Java, an interface is like a contract that classes implementing it must follow. An interface contains a set of variables and methods without anybody(no implementation, only method name, and signature). We cannot instantiate objects from interfaces. Hence, the below line of code is incorrect and throws a compile-time error saying “Cannot instantiate the type WebDriver”.

WebDriver driver = new WebDriver();

For instantiation of driver objects, we need classes like FirefoxDriver or ChromeDriver which have implemented the WebDriver interface. In other words, these driver classes have followed the contract of WebDriver by implementing all the methods of the WebDriver interface. Thus, making all the different types of driver classes uniform, following the same protocol.

Please note that we can also create a reference variable of type FirefoxDriver like this-

FirefoxDriver driver = new FirefoxDriver();

But having a WebDriver reference object helps in multi-browser testing as the same driver object can be used to assign to any of the desired browser-specific drivers.

## Launching Firefox Browser

Firefox is one of the most widely used browsers in automation. The following steps are required to launch the firefox browser.

Download geckodriver.exe from GeckoDriver Github Release Page. Make sure to download the right driver file based on your platform and OS version.

Set the System Property for “webdriver.gecko.driver” with the geckodriver.exe path – System.setProperty(“webdriver.gecko.driver”,”geckodriver.exe path”);

public class FirefoxBrowserLaunchDemo {

public static void main(String[] args) {

//Creating a driver object referencing WebDriver interface

WebDriver driver;

//Setting webdriver.gecko.driver property

System.setProperty("webdriver.gecko.driver", pathToGeckoDriver + "\\geckodriver.exe");

//Instantiating driver object and launching browser

driver = new FirefoxDriver();

//Using get() method to open a webpage

driver.get("https://artoftesting.com");

//Closing the browser

driver.quit();

}

## Launching Chrome Browser

For running the Chrome browser in Selenium, we need to set the webdriver.chrome.driver system property to point to a chrome driver executable file

Download the latest ChromeDriver binary from Chromium.org download page and place the executable on your local machine.

Set the webdriver.chrome.driver property to the chromeDriver.exe’s location as-

System.setProperty(“webdriver.chrome.driver”, “chromeDriver.exe path”);

public class ChromeBrowserLaunchDemo {

public static void main(String[] args) {

//Creating a driver object referencing WebDriver interface

WebDriver driver;

//Setting the webdriver.chrome.driver property to its executable's location

System.setProperty("webdriver.chrome.driver", "/lib/chromeDriver/chromedriver.exe");

//Instantiating driver object

driver = new ChromeDriver();

//Using get() method to open a webpage

driver.get("https://artoftesting.com");

//Closing the browser

driver.quit();

}

}

## Launching Internet Explorer Browser

Like ChromeDriver, InternetExplorer driver also requires setting up the “webdriver.ie.driver” property with the location of IEDriverServer.exe. The IEDriverServer.exe can be downloaded from here. Following code, snippet can be used to launch IE browser-

public class IEBrowserLaunchDemo {

public static void main(String[] args) {

//Creating a driver object referencing WebDriver interface

WebDriver driver;

//Setting the webdriver.ie.driver property to its executable's location

System.setProperty("webdriver.ie.driver", "/lib/IEDriverServer/IEDriverServer.exe");

//Instantiating driver object

driver = new InternetExplorerDriver();

//Using get() method to open a webpage

driver.get("https://artoftesting.com");

//Closing the browser

driver.quit();

}

}

## Launching Safari Browser

The Safari browser doesn’t require any additional configuration and can be directly launched by instantiating with SafariDriver. The following code snippet can be used to launch the Safari browser-

public class SafariBrowserLaunchDemo {

public static void main(String[] args) {

//Creating a driver object referencing WebDriver interface

WebDriver driver;

//Instantiating driver object with SafariDriver

driver = new SafariDriver();

//Using get() method to open a webpage

driver.get("https://artoftesting.com");

//Closing the browser

driver.quit();

}

}