**1 :** **What Is Selenium WebDriver/Selenium 2?**  
 **Answer** :

* **Selenium WebDriver software testing tool Is well designed object oriented API which Is developed to automate web and mobile applications testing process**. WebDriver API Is bigger than Selenium RC but It's Architecture Is simple and easy to understand compared Selenium RC API.
* We can automate our web application's software testing process using selenium webdriver.
* We can say it is advanced version of selenium RC software testing tool because some limitations of selenium RC has been overcome In selenium WebDriver software testing tool.
* WebDriver Is designed to provide better support for dynamic changing pages. **Example :**Web page elements of software web application is changing without reloading the page. In this case WebDriver works better.
* Selenium Webdriver software testing tool Is more faster that Selenium RC software testing tool as It Is directly Interacting with web browsers and mimic the behavior of a real user.**Example :**User clicks on button of web page or moving mouse on main menu to get the sub menu list. WebDriver works Same.
* All popular browser vendors are active participants In selenium WebDriver's development and all of them have their own engineers team to Improve this framework.

**2 : Tell Me WebDriver Supported Browsers?**

* **Answer** : Selenium WebDriver API has a many different drivers to test your web application In different browsers. List of Webdriver browser drivers are as bellow**.**
* **Firefox Driver** - For Mozilla Firefox browser
* **Internet Explorer Driver** - For Internet Explorer browser
* **Chrome Driver** - For Google Chrome browser
* **HtmlUnit Driver** - GUI-Less(Headless) browser for Java programs
* **Opera Driver** - For Opera browser

**3 :** **Which Different Element Locators Supported By Selenium WebDriver?**  
 **Answer** : Selenium WebDriver supports bellow given element locators.

* **XPath Locator ->**[**View Example**](http://www.software-testing-tutorials-automation.com/2014/01/how-to-locate-element-by-xpath-in.html)
* **CSSSelector Locator ->**[**View Example**](http://www.software-testing-tutorials-automation.com/2014/01/selenium-webdriver-bycssselector.html)
* **ClassName Locator ->**[**View Example**](http://www.software-testing-tutorials-automation.com/2014/01/locating-web-element-by-classname-in.html)
* **ID Locator ->**[**View Example**](http://www.software-testing-tutorials-automation.com/2014/01/how-to-locate-elements-by-id-in.html)
* **Name Locator ->**[**View Example**](http://www.software-testing-tutorials-automation.com/2014/01/selenium-webdriver-element-locator.html)
* **LinkText Locator ->**[**View Example**](http://www.software-testing-tutorials-automation.com/2014/01/how-to-locate-element-by-link-text-or.html)
* **PartialLinkText Locator ->**[**View Example**](http://www.software-testing-tutorials-automation.com/2014/01/how-to-locate-element-by-link-text-or.html)
* **TagName Locator ->**[**View Example**](http://www.software-testing-tutorials-automation.com/2014/01/element-locators-in-selenium-2-or.html)

**4 :** **Can you tell me the syntax to open/launch Firefox browser In WebDriver software testing tool?**

**Answer** : We can open new Mozilla Firefox browser Instance using bellow given syntax In WebDriver software testing tool.  
 WebDriver driver = new FirefoxDriver();

**5 :** **What Is XPath and what Is use of It In WebDriver?**

**Answer** : In Selenium WebDriver software testing tool, XPath is used to locate the elements. Using XPath, We can navigate through elements and attributes In an XML document to locate software webpage elements like buttons, text box, links, Images etc..

**6 :** **Which tool you are using to find the XPath of any element?**

**Answer** : I am using Mozilla Firefox AddOns FireBug and FirePath to find the XPath of software web elements. see [**THIS POST**](http://www.software-testing-tutorials-automation.com/2015/07/steps-to-install-firebug-and-firepath.html) to download it.

**7 :** **What is the difference between absolute XPath and relative XPath?**

**Answer** :  
 **Absolute XPath** : Absolute XPath Is the full path starting from root node and ends with desired descendant element's node. It will start using single forward slash(/) as bellow.

**Example Of Absolute XPath :**

/html/body/div[3]/div[2]/div[2]/div[2]/div[2]/div[2]/div[2]/div/div[4]/div[1]/div/div/di v/div[1]/div/div/div/div[1]/div[2]/form/table/tbody/tr[1]/td/input

Above XPath Is absolute XPath of calc result box given on [**THIS PAGE**](http://only-testing-blog.blogspot.in/2014/04/calc.html). It starts top node html and ends with input node.

**Relative XPath** : Instead of starting from root node, Relative XPath starts from any In between node or current element's node(last node of element). It will start using double forward slash(//) as bellow.  
 **ExampleOf Relative XPath :**  
 //input[@id='Resultbox']

**8:** **How To Handle Dynamic Changing IDs In XPath.**  
 **Example : //div[@id='post-body-3647323225296998740']/div[1]/form[1]/input[1]**  
 **In this XPath "3647323225296998740" Is changing every time when reloading the page. How to handle this situation?**

**Answer** : There are many different alternatives In such case.  
  
 **Alternative 1** : Look for any other attribute which Is not changing every time In that div node like name, class etc. So If this div node has class attribute then we can write xpath asbellow.

//div[@class='post-body entry-content']/div[1]/form[1]/input[1]

**Alternative 2** : You can use absolute xpath(full xpath) where you not need to give any attributenamesInxpath.

/html/body/div[3]/div[2]/div[2]/div[2]/div[2]/div[2]/div[2]/div/div[4]/div[1]/div/div/div/div[1]/div/div/div/div[1]/div[2]/div[1]/form[1]/input[1]

**Alternative 3** : Use starts-with function. In this xpath's ID attribute, "post-body-" part remain same every time. So you can use xpath as bellow.  
**//div[@id='post-body-3647323225296998740']/div[1]/form[1]/input[1]**

//div[starts-with(@id,'post-body-')]/div[1]/form[1]/input[1]

**Alternative 4** : Use contains function. Same way you can use contains function as bellow.

div[contains(@id,'post-body-')]/div[1]/form[1]/input[1]

**9 :** **How to press ENTER key button on text box In selenium webdriver?**

**Answer** : To press ENTER key using selenium WebDriver software automation tool, We need to use selenium Enum Keys with Its constant ENTER as bellow.

driver.findElement(By.xpath("//input[@id='gbqfq']")).sendKeys(Keys.ENTER);

**10 :** **How many types of waits available In selenium WebDriver**

**Answer** : There are two types of waits available In selenium WebDriver software automation testing tool.

1. Implicit Wait
2. Explicit Wait

[**http://www.software-testing-tutorials-automation.com/2014/05/selenium-webdriver-tutorials-part-two.html**](http://www.software-testing-tutorials-automation.com/2014/05/selenium-webdriver-tutorials-part-two.html)

**11 :** **What Is Implicit Wait In Selenium WebDriver?**

**Answer** : Sometimes, Elements are taking time to be appear on software web application page. Using Implicit wait In webdriver software testing test case, We can poll the DOM for certain amount of time when some element or elements are not available Immediately on webpage.  
  
**Implicit Wait Example** :

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

If you will write above syntax In your test, Your WebDriver test will wait 10 seconds for appearing element on page.

**12 :** **What Is Explicit Wait In Selenium WebDriver?**

**Answer** : Using explicit wait code In selenium webdriver software automation testing tool, You can define to wait for a certain condition to occur before proceeding further test code execution.

**Explicit Wait Example** :

WebDriverWait wait = new WebDriverWait(driver, 20);

wait.until(ExpectedConditions.elementToBeClickable(By.xpath("//input[@id='gbqfq']")));

Above code will wait for 20 seconds for targeted element to be displayed and enabled or we can say clickable.

**13 :** **I wants to pause my test execution for fix 10 seconds at specific point. How can I do It?**

**Answer** : You can use **java.lang.Thread.sleep(long milliseconds)** method to pause the software test execution for specific time. If you wants to pause your test execution for 10 seconds then you can use bellow given syntax In your test.

Thread.sleep(10000);

**14 :** **How does selenium RC software testing tool drive the browser?**

**Answer** :  
When browser loaded In Selenium RC, It ‘injected’ javascript functions into the browser and then It Is using javascript to drive the browser for software application under test.

**15 :** **How does the selenium WebDriver drive the browser?**

**Answer** : Selenium webdriver software testing tool works like real user Interacting with software web page and Its elements. It Is using each browser's native support to make direct calls with browser for your software application under test. There Is not any Intermediate thing In selenium webdriver to Interact with web browsers.

**16 :** **Do you need Selenium Server to run your tests In selenium WebDriver?**

**Answer** : It depends. If you are using only selenium webdriver API to run your tests and you are running your all your tests on same machine then you do not need selenium server because In this case, webdriver can directly Interact with browser using browser's native support.  
  
You need selenium server with webdriver when you have to perform bellow given operations with selenium webdriver.

* When you are using remote or virtual machine to run webdriver tests for software web application and that machine have specific browser version that is not on your current machine.
* When you are using selenium-grid to distribute your webdriver's test execution on different remote or virtual machines.

**17 :** **Bellow given syntax will work to navigate to specified URL In WebDriver? Why?**

* **driver.get("www.google.com");**
* **Answer** : **No**. It will not work and show you an exception like : "Exception in thread "main" org.openqa.selenium.WebDriverException: f.QueryInterface is not a function" when you run your test.  
    
  You need to provide **http://** protocol with URL In **driver.get** method as bellow.
* **driver.get("http://www.google.com");**
* Now It will work.

**18 :** **Can we automate desktop software application's testing using selenium WebDriver?**

**Answer** : No. This Is the biggest disadvantage of selenium WebDriver API. We can automate only web and mobile software application's testing using selenium WebDriver.

**19 :** **Can you tell me the alternative driver.get() method to open URL In browser?**

**Answer** : We can use anyone from bellow given two methods to open URL In web browser In selenium webdriver software testing tool.

1. driver.get()
2. driver.navigate().to()

**20 :** **Can you tell me a difference between driver.get() and driver.navigate() methods?**

**Answer** : Main and mostly used functions of both methods are as bellow.  
  
**driver.get()**

* driver.get() method Is generally used for Open URL of software web application.
* It will wait till the whole page gets loaded.

**driver.navigate()**

* driver.navigate() method Is generally used for navigate to URL of software web application, navigate back, navigate forward, refresh the page.
* It will just navigate to the page but wait not wait till the whole page gets loaded.

**21 :** **WebDriver has built In Object Repository. Correct me If I am wrong.**

* **Answer** : No. WebDriver do not have any built In object repository till now. But yes, I am using java .properties file In my framework to store all required element objects In my tests

**22 :** **Can you tell me syntax to set browser window size to 800(Width) X 600(Height)?**

* **Answer** : We can set browser window size using **setSize**method of selenium webdriver software testing tool. To set size at 800 X 600, Use bellow given syntax In your test case.
* driver.manage().window().setSize(new Dimension(500,500));

**23 :** **Can you tell me the names of different projects of selenium software automation testing tool?**

**Answer** : At present, Selenium software automation testing tool has four different projects as bellow.

* **Selenium IDE** : It Is Firefox add-on which allows you to record and playback your software web application's tests In Firefox browser.
* **Selenium RC** : It Is software web application automation tool which allows you to write your tests In many different programming languages.
* **Selenium WebDriver** : It Is well designed object oriented API developed to automate web and mobile software application testing process. You can write your tests In different languages too In selenium webdriver.
* **Selenium Grid** : Grid allows you to execute your tests In parallel by distributing them on different machines having different browser/OS combinations.

**24 :** **Can you tell me the usage of "submit" method In selenium WebDriver?**

* **Answer** : We can use submit method to submit the forms In selenium WebDriver software automation testing tool. Example : Submitting registration form, submitting LogIn form, submitting Contact Us form ect.. After filling all required fields, We can call submit method to submit the form

# Submitting Form Using submit() Method Of Selenium WebDriver

You will find many forms In any software web application like Contact Us form, New User Registration Form, Inquiry Form, LogIn Form etc.. Supposing you are testing one software website where you have to prepare **Login form submission test case In selenium webdriver** then how will you do It? Simplest way Is described In [**THIS POST**](http://www.software-testing-tutorials-automation.com/2014/05/webdriver-test-data-driven-testing.html). If you will see In that example post, we have used .click() method to click on Login button.

**Selenium Webdriver** software testing tool has one special method to **submit any form** and that method name Is **submit()**. submit() method works same as clicking on submit button.

[**WEBDRIVER TUTORIAL PART 2**](http://www.software-testing-tutorials-automation.com/2014/05/selenium-webdriver-tutorials-part-two.html)

**When to use .click() method**

You can use .click() method to click on any button of software web application. Means element's type = "button" or type = "submit", .click() method will works for both.

**When to use .submit() method**

If you will look at firebug view for any form's submit button then always It's type will be "submit" as shown In bellow given Image. In this case, .submit() method Is very good alternative of .click() method.

**Final Notes :**

1. If any form has submit button which has type = "button" then .submit() method will not work.

2. If button Is not Inside <form> tag then .submit() method will not work.

Now let us take a look at very simple example where I have used .submit() method to submit form on software web page. In bellow given example, I have not used .click() method but used .submit() method with company name field. Run bellow given example In eclipse with testng and verify the result.

package Testng\_Pack;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class Form\_Submit {

WebDriver driver = new FirefoxDriver();

@BeforeTest

public void setup() throws Exception {

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

driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

driver.get("http://only-testing-blog.blogspot.in/2014/05/form.html");

}

@AfterTest

public void tearDown() throws Exception {

driver.quit();

}

@Test

public void LogIn\_Test(){

driver.findElement(By.xpath("//input[@name='FirstName']")).sendKeys("MyFName");

driver.findElement(By.xpath("//input[@name='LastName']")).sendKeys("MyLName");

driver.findElement(By.xpath("//input[@name='EmailID']")).sendKeys("My Email ID");

driver.findElement(By.xpath("//input[@name='MobNo']")).sendKeys("My Mob No.");

driver.findElement(By.xpath("//input[@name='Company']")).sendKeys("My Comp Name");

**//To submit form.**

**//You can use any other Input field's(First Name, Last Name etc.) xpath too In bellow given syntax.**

driver.findElement(By.xpath("//input[@name='Company']")).submit();

String alrt = driver.switchTo().alert().getText();

driver.switchTo().alert().accept();

System.out.println(alrt);

}

}

**25 :** **Do you have faced any Issue with "submit" method any time?**

**Answer** : Yes, I have faced Issue like submit method was not working to submit the form. In this case, Submit button of form was located outside the opening <form> and closing </form> tags. In this case submit method will not works to submit the form.  
  
Also If submit button Is located Inside opening **<form>** and closing **</form>** tags but that button's **type**tag's attribute Isn't **submit**then submit method will not work. It(**type**tag's attribute) should be always submit.

**26 :** **What Is the syntax to type value In prompt dialog box's Input field using selenium WebDriver?**

**Answer** : Prompt dialog Is just like confirmation alert dialog but **with option of Input text box** as bellow.

To Input value In that text box of prompt dialog, You can use bellow given syntax.

driver.switchTo().alert().sendKeys("Jhon");

**27 :** **When I am running software web application's tests In Firefox Browser using selenium webdriver, It Is not showing me any bookmarks, addons, saved passwords etc. In that browser. Do you know why?**

**Answer** : Yes. It Is because all those bookmarks, addons, passwords etc.. are saved In your regular browser's profile folder so when you launch browser manually, It will use existing profile settings so

It will show you all those stuffs. But when you run your software web application's tests In selenium webdriver, It Is opening new browser Instance with blank/new profile. So It will not show you bookmarks and all those things In that browser Instance.

You can create custom firefox profile and then you can use It In selenium webdriver test. In your custom profile, you can set all required bookmarks, addons etc.. [**VIEW THIS EXAMPLE**](http://www.software-testing-tutorials-automation.com/2014/05/how-to-create-and-use-custom-firefox.html) to know how to set custom profile of firefox browser.

**28 :** **On Google search page, I wants to search for some words without clicking on Google Search button. Is It possible In WebDriver? How?**

**Answer** : Yes we can do It using WebDriver **sendKeys** method where we do not need to use Google Search button. Syntax Is as bellow.

driver.findElement(By.xpath("//input[@id='gbqfq']")).sendKeys("Search Syntax",Keys.ENTER);

In above syntax, **//input[@id='gbqfq']** Is xPath of Google search text field. First It will enter "Search Syntax" text In text box and then It will press Enter key on same text box to search for words on Google.

**29** : **Give me any five different xPath syntax to locate bellow given Input element.**

<input id="fk-top-search-box" class="search-bar-text fk-font-13 ac\_input" type="text" autofocus="autofocus" value="" name="q" />

**Answer** : Five xPath syntax for above element of software web application page are as bellow.

//input[@id='fk-top-search-box']

//input[contains(@name,'q')]

//input[starts-with(@class, "search-bar-text")]

//input[@id='fk-top-search-box' or @name='q']

//input[starts-with(@id, 'fk-top-search-box') and contains(@class,'fk-font-13')]

**30** : **Can you tell me two drawbacks of xPath locators as compared to cssSelector locator?**

**Answer** : Two main disadvantage of xPath locator as compared to cssSelector locator are as bellow.

* It Is slower than cssSelector locator.
* xPath which works In one browser may not work In other browser for same page of software web application because some browsers (Ex. IE) reads only Lower-cased tag name and Attribute Name. So If used It In upper case then It will work In Firefox browser but will not work In IE browser. Every browser reads xPath In different way. In sort, do not use xPath locators In your test cases of software web application If you have to perform cross browser testing using selenium WebDriver software testing tool.

**31** : **Why xPath locator Is much more popular than all other locator types In WebDriver?**

**Answer** : xPath locators are so much popular In selenium webdriver test case development because

* It Is very easy to learn and understand for any new user.
* There are many functions to build xPath In different ways like contains, starts-with etc.. So If one Is not possible you will have always another option to build xPath of any element.
* Presently many tools and add-ons are available to find xpath of any element.

**32** : **Do you have faced any technical challenges with Selenium WebDriver software test automation?**

**Answer** : Yes, I have faced below given technical challenges during selenium webdriver test cases development and running for software web application.

* Sometimes (Not always), Some elements like text box, buttons etc. are taking more time(more than given Implicit wait time) to appear on page of software web application or to get enabled on page. In such situation, If I have used only Implicit wait then my test case can run fine on first run but It may fail to find element on second run. So we need provide special treatment for such elements so that webdriver script wait for element to be present or get enabled on page of software web application during test execution. We can use Explicit wait to handle this situation. You can find different explicit wait example links on [**THIS PAGE**](http://www.software-testing-tutorials-automation.com/2014/05/selenium-webdriver-tutorials-part-two.html).
* Handling dynamic changing ID to locate element Is tricky. If element's ID Is changing every time when you reload the software web application page and you have to use that ID In XPath to locate element then you have to use functions like starts-with(@id,'post-body-') or contains(@id,'post-body-') In XPath. Other alternate solutions to handle this situation are described in answer of Question 15 of [**THIS PAGE**](http://www.software-testing-tutorials-automation.com/2014/09/selenium-webdriver-latest-job-interview.html).
* Clicking on sub menus which are getting rendered on mouse hover of main menu Is some what tricky. You need to use webdriver's Actions class to perform mouse hover operation. You can[**VIEW FULL EXAMPLE**](http://www.software-testing-tutorials-automation.com/2014/02/selenium-webdriver-generating-mouse.html) on how to generate mouse hover event on main menu.
* If you have to execute your test cases In multiple browsers then one test case can run successfully In Firefox browser but same test case may fail In IE browser due to the timing related Issues (nosuchelement exception) because test execution In Firefox browser Is faster than IE browser. You can resolve this Issue by Increasing Implicit wait time when you run your test In IE browser.
* Above Issue can arise due to the unsupported XPath In IE browser. In this case, You need to you[**OTHER ELEMENT LOCATING METHODS**](http://www.software-testing-tutorials-automation.com/search/label/Webdriver%20Element%20Locators) (ID, Name, CSSSelector etc.)to locate element.
* Handling JQuery elements like moving pricing slider, date picker, drag and drop etc.. Is tricky. You should have knowledge of webdriver's Advanced User Interactions API to perform all these actions. You can find few example links for working with JQuery Items on [**THIS PAGE**](http://www.software-testing-tutorials-automation.com/2014/10/selenium-webdriver-advanced-tutorials.html).
* Working with [**multiple Windows**](http://www.software-testing-tutorials-automation.com/2014/02/example-of-handling-multiple-browser.html), Frames, and some tasks like [**Extracting data from web table**](http://www.software-testing-tutorials-automation.com/2014/05/how-to-extract-table-dataread-table.html),[**Extracting data from dynamic web table**](http://www.software-testing-tutorials-automation.com/2014/06/how-to-handle-dynamic-web-table-in.html), [**Extracting all Links from page**](http://www.software-testing-tutorials-automation.com/2014/02/how-to-getextract-all-links-from-web.html), [**Extracting all text box from page**](http://www.software-testing-tutorials-automation.com/2014/05/selenium-webdriver-extracting-all-text.html) are also tricky and time consuming during test case preparation.
* There Is not any direct command to upload or download files from web page using selenium webdriver. For downloading files usign selenium webdriver, You need to create and set Firefox browser profile with webdriver test case. You can [**VIEW PRACTICAL EXAMPLE**](http://www.software-testing-tutorials-automation.com/2014/05/how-to-download-different-files-using.html).
* Webdriver do not have any built In object repository facility. You can do It using java .properties file to create object repository as described In [**THIS EXAMPLE**](http://www.software-testing-tutorials-automation.com/2014/05/creating-object-repository-using.html).
* Webdriver do not have any built In framework or facility using which we can achieve below given tasks directly : 1. Capturing screenshots, 2. generating test execution log, 3. reading data from files, 4. Generating test result reports, Manage test case execution sequence. To achieve all these tasks, We have to use external services with webdriver like Log4J to generate log, Apache POI API to read data from excel files, Testng XSLT reports to generate test result reports. TestNG to manage test case execution, .properties file to create object repository. All these tasks are very time consuming. [**ON THIS PAGE**](http://www.software-testing-tutorials-automation.com/2014/07/create-data-driven-framework-for.html), I have described how to create data driven framework step by step for selenium webdriver. That framework contains all above functionality.

You can share other webdriver technical challenges which you have faced by commenting below so that your experience can help to others too.

**33** : **Can you tell me a syntax to close current webdriver Instance and to close all opened webdriver Instances?**  
 **Answer** :  
Yes, To close current WebDriver Instance, We can use Close() method as bellow.

driver.close();

If there are opened multiple webdriver Instances and wants to close all of them then we can use webdriver's quit() method as bellow in software automation test.

driver.quit();

**34** : **Is It possible to execute javascript directly during software test execution? If Yes then tell me how to generate alert by executing javascript In webdriver script?**  
 **Answer** :  
Yes, we can execute javascript during webdriver software test execution. To generate alert, You can write bellow given code In your script.

JavascriptExecutor javascript = (JavascriptExecutor) driver;

javascript.executeScript("alert('Javascript Executed.');");

**35** : **Give me a syntax to read javascript alert message string, clicking on OK button and clicking on Cancel button.**  
 **Answer** :  
We can read alert message string as bellow.

String alrtmsg = driver.switchTo().alert().getText();

We can click on OK button of alert as bellow.

driver.switchTo().alert().accept();

We can click on Cancel button of alert as bellow.

driver.switchTo().alert().dismiss();

**36** : **What are the advantages of TestNG over JUnit.**

**Answer** : Advantages of TestNG over JUnit JUnit are as bellow.

 TestNG Annotations are simple and Easy to understand.

 Easy to parameterize the software test cases In TestNG.

 Easy to run software automation test cases In parallel.

 Can generate Interactive XSLT test execution reports using TestNG.

**37** : **What Is the syntax to get value from text box and store It In variable.**

**Answer** : Most of the time, String In text box will be stored as value. So we need to access value attribute(getAttribute) of that text box as shown In bellow example.

String Result = driver.findElement(By.xpath("//input[@id='Resultbox']")).getAttribute("value");

**38** : **What Is the difference between findelement and findElements ?**

**Answer** : findElement Is useful to locate and return single element from page of software web application while findElements Is useful to locate and return multiple elements from software web page.

**39** : **Tell me looks like XPath of sibling Input element which Is after Div in the DOM.**

**Answer** : XPath for above scenario will be something like bellow.

**//div/following-sibling::input**

**40** : **Tell me looks like CSSSelector path of sibling Input element which Is after Div in the DOM.**

**Answer** : CSSSelecor path will looks like bellow.

**css=div + input**

**41** : **What Is Parallelism In TestNG?**

**Answer** : In general software term, Parallelism means executing two part of software program simultaneously or executing software program simultaneously or we can say multithreaded or parallel mode. TestNG has same feature using which we can start multiple threads simultaneously In parallel mode and test methods will be executed In them.

**42** : **What are the benefits of parallelism over normal execution?**

**Answer** : Using parallelism facility of TestNG in selenium webdriver,

* Your software test execution time will be reduced as multiple tests will be executed simultaneously.
* Using parallelism, We can verify multithreaded code In software application.

**43** : **I wants to run test cases/classes In parallel. Using which attribute and value I can do It?**

**Answer** : You have to use **parallel = classes** attribute In testng.xml to run software web app tests parallel. [**VIEW EXAMPLE**](http://www.software-testing-tutorials-automation.com/2014/11/executing-selenium-webdriver-test.html).

**44** : **What Is dependency test In TestNG?**

**Answer** : Dependency Is very good feature of testng using which we can set software test method as dependent test method of any other single or multiple or group of test methods. That means depends-on method will be executed first and then dependent test method will be executed. If depends-on software test method will fail then execution of dependent test method will be skipped automatically. TestNG dependency feature will works only If depends-on test method Is part of same class or part of Inherited base class.

**45** : **What Is the syntax to set test method dependency on multiple test methods.**

**Answer** : We can set test method's dependency on multiple test methods as bellow.

@Test(**dependsOnMethods={"Login","checkMail"}**)

public void LogOut() {

System.out.println("LogOut Test code.");

}

Above test method Is depends on **Login** and **checkMail** test methods.

**46** : **What Is the syntax to set test method disabled.**

**Answer** : We can use attribute **enabled = false** with @Test annotation to set test method disabled. Syntax Is as bellow.

@Test(**enabled = false**)

public void LogOut() {

System.out.println("LogOut Test code.");

}

Disabled software test methods will be excluded automatically during execution.

**47** : **In XPath, I wants to do partial match on attribute value from beginning. Tell me two functions using which I can do It.**

**Answer** : We can use bellow given two functions with XPath to find element for software web page using attribute value from beginning.

1. contains()
2. starts-with()

**48** : **I have used findElements In my software test case. It Is returning NoSuchElementException when not element found. Correct me If I am wrong.**

**Answer** : It Is Incorrect. findElements will never return NoSuchElementException. It will return just an empty list.

**49** : **My Firefox browser Is not Installed at usual place. How can I tell FirefoxDriver to use It?**

**Answer** : If Firefox browsers Is Installed at some different place than the usual place then you needs to provide the actual path of Firefox.exe file as bellow.

**System.setProperty("webdriver.firefox.bin","C:\\Program Files\\Mozilla Firefox\\Firefox.exe");**

driver =new FirefoxDriver();

**50** : **How to create custom firefox profile and how to use It In selenium webdriver software test?**

**Answer** : You can view detailed answer for firefox custom profile on [**THIS PAGE**](http://www.software-testing-tutorials-automation.com/2014/05/how-to-create-and-use-custom-firefox.html).

**How To Create And Use Custom Firefox Profile For Selenium WebDriver**

**What Is the Firefox Profile?**

When you Install Firefox In your computer, **Firefox** creates one default **profile**folder In your local drive to save your preferences like your bookmarks, your preferred home page on Firefox open, your toolbar settings, your saved passwords and all the other settings. You can **create different profiles for your Firefox browser** as per your software testing requirement. Now supposing same computer Is used by two users and both wants their own Firefox settings then both users can create their own Firefox profile to access their own settings when he/she opens Firefox browser.

You can [**LEARN SELENIUM WEBDRIVER STEP BY STEP**](http://www.software-testing-tutorials-automation.com/2014/01/learn-selenium-webdriver-online-free.html) to become master of selenium Webdriver software testing tool.

If you have noticed, When you will run your selenium webdriver software automation test In Firefox browser then WebDriver will open blank Firefox browser like No bookmarks, No saved passwords,  No addons etc.. as shown In bellow given Image.

If you wants access of all these things In your selenium webdriver software automation test browser then you have to create new profile of Firefox and set all required properties In newly created profile and then you can access that profile In webdriver using FirefoxProfile class of webdriver.

First Of all, Let us see how to create new profile of Firefox and then we will see how to use that profile In your test.

**How to Create New Custom Firefox Profile For Selenium WebDriver?**  
To create new firefox profile manually,

* Close Firefox browser from File -> Exit.
* Go to Start -> Run and type "firefox.exe -p" In run window and click OK. It will open "Firefox - Choose User Profile" dialogue as shown In bellow Image.
* Now click on Create profile button. It will open create profile wizard dialogue. Click On Next as shown In bellow given Image.
* On next screen, Give profile name = "WebDriver\_Profile" and click on finish button as shown In bellow given Image.
* It Will create new profile of Firefox.
* To use that newly Created profile, Select that profile and click on Start Firefox button as shown In bellow given Image. It will open Firefox browser with newly created profile.

Now you can make you required settings on this new created profile browser like add your required addons, bookmark your required page, network settings, proxy settings, etc.. and all other required settings.

**How To Access Custom Firefox(Changing User Agent) Profile In Selenium WebDriver Test**

To access newly created Firefox profile In selenium WebDriver software test, We needs to use webdriver's Inbuilt class ProfilesIni and Its method getProfile as shown In bellow given example.

package Testng\_Pack;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.firefox.FirefoxProfile;

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

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class custom\_profile {

WebDriver driver;

@BeforeTest

public void StartBrowser() {

**//Create object of webdriver's inbuilt class ProfilesIni to access Its method getProfile.**

ProfilesIni firProfiles = new ProfilesIni();

**//Get access of newly created profile WebDriver\_Profile.**

FirefoxProfile wbdrverprofile = firProfiles.getProfile("WebDriver\_Profile");

**//Pass wbdrverprofile parameter to FirefoxDriver.**

driver = new FirefoxDriver(wbdrverprofile);

}

@Test

public void OpenURL(){

driver.get("http://only-testing-blog.blogspot.in/2014/05/login.html");

}

@AfterTest

public void CloseBrowser() {

driver.quit();

}

}

When you will run above given example, It will open Firefox browser with newly created profile settings.

**Why needs To Set Firefox Profile In Selenium WebDriver**

To perform some actions In your selenium webdriver software automation test, You need special Firefox profile. Some example actions are as bellow where we need to set Firefox profile. We will learn more about how to perform all those actions In Selenium webdriver software automation testing tool In my upcoming posts.

1. To Download files.   [**VIEW EXAMPLE**](http://www.software-testing-tutorials-automation.com/2014/05/how-to-download-different-files-using.html)

2. To Set Proxy Settings. [**VIEW EXAMPLE**](http://www.software-testing-tutorials-automation.com/2015/03/how-to-set-proxy-settings-in-selenium.html)

3. To Resolve Certificate related errors. [**VIEW EXAMPLE**](http://www.software-testing-tutorials-automation.com/2015/03/how-to-handle-ssl-certificate-error-in.html)

If you have any example where we need to set Firefox profile properties then you can share It with world by commenting bellow.

**51** : **What versions of Internet Explorer are supported by selenium WebDriver software testing tool?**

**Answer** : Till date, Selenium WebDriver software testing tool supports IE 6, 7, 8, 9, 10 and 11 with appropriate combinations of Windows 7, Vista or XP.

**52** : **Tell me the class name using which we can generate Action chain.**

**Answer** : The WebDriver class name Using which we can generate Action chain Is "**Actions**". [**VIEW USAGE OF ACTIONS CLASS**](http://www.software-testing-tutorials-automation.com/2014/09/how-to-drag-and-drop-element-in.html) with practical example on how to generate series of actions to drag and drop element of software web application.

# How To Drag And Drop An Element In Selenium WebDriver

When we talk about **drag and drop** kind of tricky operations, We need to use **Advanced User Interactions API** of selenium WebDriver. It Is not so simple and at the same time not too much hard. We need to write multiple syntax to **perform drag and drop action** because you need to take multiple actions like pick the element by **clicking and holding mouse** then move mouse to destination place and then drop element by releasing mouse. Same thing you have to perform In webdriver.

Selenium WebDriver has Advanced User Interactions API (**Actions**class) to perform this kind of advanced user interactions for rich applications. We can use**Actions**class and It different methods like dragAndDrop, clickAndHold, moveToElement, release and build to composite all the actions of drag and drop operation as shown In bellow given example. At last perform method will perform the action.

package Testing\_Pack;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.interactions.Action;

import org.openqa.selenium.interactions.Actions;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class DragAndDrop {

WebDriver driver = null;

@BeforeTest

public void setup() throws Exception {

driver = new FirefoxDriver();

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

driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);

driver.get("http://only-testing-blog.blogspot.in/2014/09/drag-and-drop.html");

}

@Test

public void dragAndDrop() {

**//Locate element which you wants to drag.**

WebElement dragElementFrom = driver.findElement(By.xpath("//div[@id='dragdiv']"));

**//Locate element where you wants to drop.**

WebElement dropElementTo = driver.findElement(By.xpath("//div[@id='dropdiv']"));

**//Use Actions class and Its members of WebDriver API to perform drag and drop operation.**

Actions builder = new Actions(driver);

Action dragAndDrop = builder.clickAndHold(dragElementFrom)

.moveToElement(dropElementTo)

.release(dropElementTo)

.build();

dragAndDrop.perform();

}

}

Above example will drag the small square element on big square element. This Is just example. You can use this example as a reference to perform drag and drop operation In your web application. If you wants to by specific X-Y pixel offset then you can view [**THIS POST**](http://www.software-testing-tutorials-automation.com/2014/10/selenium-webdriver-drag-and-drop.html).  
  
You can find more tutorials links [**HERE**](http://www.software-testing-tutorials-automation.com/p/selenium-webdriver.html) to start learning selenium WebDriver from basic.  
  
In above example, We have written total 6 lines to drag and drop an element. If you wants convert those 6 lines In only 1 line then you can do It as bellow. Replace bellow given 1 line with 6 lines of above example. It will do same thing.

new Actions(driver).dragAndDrop(dragElementFrom, dropElementTo).build().perform();

**53** : **Do you know method name using which we can builds up the actions chain?**

**Answer** : Method name of Actions class to build up actions chain Is "build()".

**54** : **Can we capture screenshot In Selenium WebDriver software testing tool? How?**

**Answer** : We can use selenium webdriver TakesScreenshot method to capture screenshot. Java File class will be used to store screenshot In your system's local drive.

**55** : **Do you know any external API name using which we can read data from excel file?**

**Answer** :

* We can use jxl API (Java Excel API) to read data from excel file. [**VIEW EXAMPLE**](http://www.software-testing-tutorials-automation.com/2014/02/parameterizationdata-driven-testing-of.html)
* We can use one more powerful API known as Apache POI API to read and write data In excel file. I have created data driven framework using Apache POI API. You can [**VIEW DATADRIVEN FRAMEWORK CREATION TUTORIALS**](http://www.software-testing-tutorials-automation.com/2014/07/create-data-driven-framework-for.html) step by step.

[**http://www.software-testing-tutorials-automation.com/2014/07/create-data-driven-framework-for.html**](http://www.software-testing-tutorials-automation.com/2014/07/create-data-driven-framework-for.html)

**56** : **Tell me any 5 webdriver common exceptions which you faced during software test case execution.**

**Answer** : WebDriver's different 5 exceptions are as bellow.

1. **TimeoutException** - This exception will be thrown when command execution does not complete In given time.
2. **NoSuchElementException** - WebDriver software testing tool will throw this exception when element could not be found on page of software web application.
3. **NoAlertPresentException** - This exception will be generated when webdriver ties to switch to alert popup but there Is not any alert present on page.
4. **ElementNotSelectableException** - It will be thrown when webdriver Is trying to select unselectable element.
5. **ElementNotVisibleException** - Thrown when webdriver Is not able to Interact with element which Is available In DOM but It Is hidden.
6. **StaleElementReferenceException**

# How To Handle staleelementreferenceexception Selenium Webdriver

# staleelementreferenceexception

There are many different exceptions In selenium webdriver software testing tool. I have described 5 of them on [**THIS PAGE**](http://www.software-testing-tutorials-automation.com/2015/01/webdriver-latest-questions.html) which we are facing frequently. One more webdriver exception Is **stale element reference exception**. I think some of you have faced or will face this exception when you run selenium webdriver test for your software web application. Just recently I have faced this exception In my test with exception message as bellow.

**Exception :**

org.openqa.selenium.StaleElementReferenceException: Element not found in the cache - perhaps the page has changed since it was looked up

Command duration or timeout: 10.03 seconds

**What Is StaleElementReferenceException?**

Stale means old or we can say no longer fresh element. Let me describe you In very simple words. Example : You have a search text box on software web page. When you search for some keyword, text box's position get changed on page. So In this case, Look and feel, Identifiers etc. of text box will remain same but what Internally happened Is -> JS library has deleted previous text box and replaced It with new same text box. So now If you will go to use same text box using previously located reference In your software test, You will get **StaleElementReferenceException**In console.

If you will run bellow given example, It will throw **StaleElementReferenceException**.

package Testing\_Pack;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.Keys;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class StaleElement {

WebDriver driver;

@BeforeTest

public void setup() throws Exception {

driver =new FirefoxDriver();

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

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

driver.get("http://www.github.com");

}

@Test

public void getExe() throws InterruptedException{

**//Located element and stored It's reference In variable.**

WebElement Search\_Box = driver.findElement(By.xpath("//input[@name='q']"));

**//Used element reference variable to locate element and perform search.**

Search\_Box.sendKeys("Hello");

Search\_Box.sendKeys(Keys.ENTER);

Thread.sleep(5000);

**//After search operation, Element's position Is changed.**

**//Now I am using same reference variable to clear search text box.**

**//So here, WebDriver will be not able to locate element using same reference and It will throw StaleElementReferenceException.**

Search\_Box.clear();

}

}

This Is just example. You can have different situation or scenario. This will happen only some times but you must know how to resolve It.There may be some other reasons too for this exception. Now let us try to resolve or handle this exception.

So Exception Is generating on Search\_Box.clear(); line because software web application page has been refreshed and search text box's position Is changed during previous search operation. In some cases, After refreshing the page or javascript action, Position of element will remains same but still you will get this exception.

I have two possible solutions to handle this situation.

**Proposed Solution 1 :**  
Create a function and perform text box clear operation Inside It as shown In bellow example using while loop. Replace @Test method of above example with bellow given.

@Test

public void getExe() throws InterruptedException {

**// Located element and stored It's reference In variable.**

WebElement Search\_Box = driver.findElement(By.xpath("//input[@name='q']"));

**// Used element reference variable to locate element and perform search.**

Search\_Box.sendKeys("Hello");

Search\_Box.sendKeys(Keys.ENTER);

Thread.sleep(5000);

**// After search operation, Element's position Is changed.**

**//Call function with element name to perform clear operation.**

handleStaleElement("q");

}

**// This function will handle stalelement reference exception**

public void handleStaleElement(String elementName) {

int count = 0;

**//It will try 4 times to find same element using name.**

while (count < 4) {

try {

**//If exception generated that means It Is not able to find element then catch block will handle It.**

WebElement staledElement = driver.findElement(By.name(elementName));

**//If exception not generated that means element found and element text get cleared.**

staledElement.clear();

} catch (StaleElementReferenceException e) {

e.toString();

System.out.println("Trying to recover from a stale element :" + e.getMessage());

count = count + 1;

}

count = count + 4;

}

}

**Proposed Solution 2 :**  
You can do same thing using for loop Inside @Test method as bellow.

@Test

public void getExe() throws InterruptedException {

**// Located element and stored It's reference In variable.**

WebElement Search\_Box = driver.findElement(By.xpath("//input[@name='q']"));

**// Used element reference variable to locate element and perform search.**

Search\_Box.sendKeys("Hello");

Search\_Box.sendKeys(Keys.ENTER);

Thread.sleep(5000);

**// After search operation, Element's position Is changed.**

**//use for loop.**

for(int i=0; i<4;i++)

try {

driver.findElement(By.xpath("//input[@name='q']")).clear();

break;

} catch(StaleElementReferenceException e) {

e.toString();

System.out.println("Trying to recover from a stale element :" + e.getMessage());

}

}

If anyone of you have faced same problem of **staleelementreferenceexception**and resolved It In any other way then you can post your solution by commenting bellow so others can use It.

**57** : **Tell me different ways to type text In text box In selenium software test.**

**Answer** : We can type text In text box of software web application page using bellow given ways In selenium test.  
  
**1.Using .SendKeys() method**

driver.findElement(By.xpath("//input[@id='fname']")).sendKeys("Using sendKeys");

**2. Using JavascriptExecutor**

((JavascriptExecutor)driver).executeScript("document.getElementById('fname').value='Using JavascriptExecutor'");

**58** : **Tell me different ways to verify element present or not on page.**

**Answer** : We can check If element Is present or not on page of software we application using bellow given 2 simple ways.  
  
1. Using .size() method

Boolean elePresent = driver.findElements( By.id("ID of element") ).size() != 0;

If above syntax return "false" means element Is not present on page and "true" means element Is present on page.  
  
2. Using .isEmpty() method

Boolean elePresent = driver.findElements(By.id("ID of element")).isEmpty();

If this returns "true" means element Is not present on page and "false" means element Is present on page.

**59** : **Why we need to customize Firefox browser profile In webdriver test?**

**Answer** : Webdriver launch fresh browser Instance when we run software tests In Firefox browser using selenium webdriver. Fresh browser Instance do not have any Installed add-ons, saved passwords, bookmarks and any other user preferences. So we need to create custom profile of Firefox browser to get any of this thing In Webdriver launched browser.

**60** : **How to customize Firefox browser profile for webdriver software test?**

**Answer** : You can do It In two different ways.

1. You can create your desired firefox browser profile before running software automation test and then you can use It In your selenium webdriver software test. [**VIEW EXAMPLE**](http://www.software-testing-tutorials-automation.com/2014/05/how-to-create-and-use-custom-firefox.html).
2. You can customize your firefox browser profile run time before launching webdriver's Firefox browser Instance. [**VIEW EXAMPLE**](http://www.software-testing-tutorials-automation.com/2014/05/how-to-download-different-files-using.html).

**61** : **What is Difference between getAttribute() and getText()?**

**Answer** :

* getAttribute() method Is useful to read software web app element's attribute value like id, name, type etc. [**VIEW EXAMPLE**](http://www.software-testing-tutorials-automation.com/2014/12/testng-timeout-test-for-selenium.html).
* getText() method Is useful to read text from element or alert

**62** : **What is the difference between WebDriver and Remote WebDriver?**

**Answer** : Simple answer for this questions Is as bellow.

* **WebDriver** : Webdriver Is an Interface or we can say software testing tool using which we can create automated test cases for web application and then run on different browsers like IE, Google chrome, Firefox etc.. We can create test cases In different languages. [**VIEW MORE DETAIL**](http://www.software-testing-tutorials-automation.com/2013/07/what-is-selenium-webdriver.html).
* **Remote WebDriver** : Remote WebDriver Is useful to run test cases In same machine or remote machines using selenium Grid.

**63** : **I have total 200 test cases. I wants to execute only 20 test cases out of them. Can I do It In selenium WebDriver? How?**

**Answer** : Yes. If you are using TestNG with selenium webdriver software testing tool then you can do Is using grouping approach as described In [**THIS PAGE**](http://www.software-testing-tutorials-automation.com/2014/12/how-to-group-and-run-selenium-test.html). Create separate group for those 20 test cases and configure testng.xml file accordingly to run only those 20 test cases.  
  
Also If you are using data driven framework then you can configure It In excel file. You can configure such data driven framework at your own by following steps given on

**64** : **Can you tell me three different ways to refresh page. Do not use .refresh() method.**

**Answer** : We can refresh browser In many different ways. Three of them are as bellow.

driver.get(driver.getCurrentUrl());

driver.navigate().to(driver.getCurrentUrl());

driver.findElement(By.xpath("//h1[@class='title']")).sendKeys(Keys.F5);

**65** : **I wants to pass parameter In software test case through testng.xml file. How can I do It?**

**Answer** : You can use <parameter> node under <test> node In testng.xml file with parameter name and value. Then you can use @Parameters annotation with parameter name In your test case of software web application.

**66** : **My page contains file upload field but I am not able to upload file using selenium webdriver software testing tool. Is there any other way using which I can upload file In selenium test?**

**Answer** : If you are not able to upload file using selenium webdriver then you can create file upload script In AutoIT and then you can use It In selenium webdriver software test. You can refer bellow given articles to learn more about It

**Upload File In Selenium WebDriver Using AutoIt**

As you know, **File uploading** Is very hard In **selenium webdriver** because It Is not able to handle file uploading dialog. So we will use **AutoIT**with selenium webdriver to upload file In web applications. We have already created AutoIt script **(Script To Upload File.exe)** In previous post which can select file from **File Upload** dialog. We will learn how to Integrate that AutoIt script with selenium webdriver In this section.

In java, It Is very easy to execute any executable file. We can run any executable script file using**Runtime.getRuntime().exec("File Path")** method. We will use this java method In our selenium webdriver test script to handle file upload dialog.

So now final selenium webdriver with AutoIt Integration test script Is as bellow.  
  
**Note**: "**Script To Upload File.exe**" and "**Test.txt**" files should be located at "E:\\AutoIT" folder.

package AutoIt;

import java.io.IOException;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class AutoIt\_Test {

WebDriver driver;

@BeforeTest

public void setup() throws Exception {

driver =new FirefoxDriver();

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

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

driver.get("http://only-testing-blog.blogspot.in/2014/01/textbox.html");

}

@Test

public void testCaseOne\_Test\_One() throws IOException, InterruptedException {

**//Click on browse button.**

driver.findElement(By.name("img")).click();

**//To execute autoIt script .exe file which Is located at E:\\AutoIT\\ location.**

Runtime.getRuntime().exec("E:\\AutoIT\\Script To Upload File.exe");

}

}

Run above script In eclipse.

* It will click on browser button to open File Upload dialog using selenium webdriver.
* Then **Runtime.getRuntime().exec("E:\\AutoIT\\Script To Upload File.exe");** syntax will execute AutoIt generated "**Script To Upload File.exe**" file. It will select file and click on open button of File Upload dialog.

On completion of script, you will see that Test.txt file Is uploaded as shown In bellow given Image.

This way we can upload file using AutoIt In selenium webdriver.

**67** : **I wants to set size and position of my browser window. Do you know how to do it in selenium webdriver?**

**Answer** : We can use **setSize** function to set size of window and setPosition function to set position of browser window.

**How To Set/Get Window Position And Size In Selenium WebDriver**

Sometimes you need to **set window position** and **size**or **get window size**and **position**In selenium software test. Selenium webdriver software testing tool has many useful methods using which we can play with web browser window for different purpose. One of them Is window().maximize(); which we use In each and every

webdriver software test to maximizing the window. Same way, webdriver has window size and position related Independent methods.

**What Is window size and position?**

Before learning how to set/get size and position, We need to understand It.

* Window size means height and width of window.
* Window position means distance of window from left side(X Coordinates) of screen and top side(Y Coordinates) of screen.

Bellow given Image describes you more about window size and position.

**window().setSize() to set window size**

We can use window().setSize() method to set the size of window In selenium webdriver software test. We can set window width to 300 and height to 500 dimensions using bellow given syntax In selenium webdriver.

driver.manage().window().setSize(new Dimension(300,500));

**window().getSize() to get window size**

We can use window().getSize() method to get size of window.

Bellow given syntax will return window height using getSize().getHeight().

driver.manage().window().getSize().getHeight()

Bellow given syntax will return window width using getSize().getWidth().

driver.manage().window().getSize().getWidth()

**window().setPosition() to set position of window**

We can set window position to 50 points from left side and 200 points from top side using bellow given syntax.

driver.manage().window().setPosition(new Point(50,200));

**window().getPosition() to get position of window**

Use window().getPosition().getX() to get window position from left side as bellow.

driver.manage().window().getPosition().getX()

To get window position from top side of screen, use window().getPosition().getY() In your software test as bellow.

driver.manage().window().getPosition().getY()

Full webdriver test to explore all above methods of webdriver window Is as bellow.

package Testing\_Pack;

import org.openqa.selenium.Dimension;

import org.openqa.selenium.Point;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class WindowSizePosition {

WebDriver driver;

@BeforeTest

public void setup() throws Exception {

driver = new FirefoxDriver();

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

}

@Test(priority=1)

public void setGetWinSize(){

**//WebDriver setSize method used to set window size width = 300 and height = 500.**

driver.manage().window().setSize(new Dimension(300,500));

**//WebDriver getSize method used to get window width and height.**

System.out.println("Window height Is -> "+driver.manage().window().getSize().getHeight());

System.out.println("Window width Is -> "+driver.manage().window().getSize().getWidth());

}

@Test(priority=2)

public void setGetWinPosition(){

**//WebDriver setPosition method used to set window position x coordinate = 50 and y coordinate = 100.**

driver.manage().window().setPosition(new Point(50,200));

**//WebDriver getPosition method used to get window position x,y coordinates.**

System.out.println("Window position X coordinates Is -> "+driver.manage().window().getPosition().getX());

System.out.println("Window position Y coordinates Is -> "+driver.manage().window().getPosition().getY());

}

}

This way you can re-size/set position of your webdriver window during test execution.

**68** : **Is there any way to get size and position of browser window in selenium webdriver?**

**Answer** : Yes.. We can get it using webdriver functions getSize and getPosition.

**How To Set/Get Window Position And Size In Selenium WebDriver**

Sometimes you need to **set window position** and **size**or **get window size**and **position**In selenium software test. Selenium webdriver software testing tool has many useful methods using which we can play with web browser window for different purpose. One of them Is window().maximize(); which we use In each and every

webdriver software test to maximizing the window. Same way, webdriver has window size and position related Independent methods.

**What Is window size and position?**

Before learning how to set/get size and position, We need to understand It.

* Window size means height and width of window.
* Window position means distance of window from left side(X Coordinates) of screen and top side(Y Coordinates) of screen.

Bellow given Image describes you more about window size and position.

**window().setSize() to set window size**

We can use window().setSize() method to set the size of window In selenium webdriver software test. We can set window width to 300 and height to 500 dimensions using bellow given syntax In selenium webdriver.

driver.manage().window().setSize(new Dimension(300,500));

**window().getSize() to get window size**

We can use window().getSize() method to get size of window.

Bellow given syntax will return window height using getSize().getHeight().

driver.manage().window().getSize().getHeight()

Bellow given syntax will return window width using getSize().getWidth().

driver.manage().window().getSize().getWidth()

**window().setPosition() to set position of window**

We can set window position to 50 points from left side and 200 points from top side using bellow given syntax.

driver.manage().window().setPosition(new Point(50,200));

**window().getPosition() to get position of window**

Use window().getPosition().getX() to get window position from left side as bellow.

driver.manage().window().getPosition().getX()

To get window position from top side of screen, use window().getPosition().getY() In your software test as bellow.

driver.manage().window().getPosition().getY()

Full webdriver test to explore all above methods of webdriver window Is as bellow.

package Testing\_Pack;

import org.openqa.selenium.Dimension;

import org.openqa.selenium.Point;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class WindowSizePosition {

WebDriver driver;

@BeforeTest

public void setup() throws Exception {

driver = new FirefoxDriver();

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

}

@Test(priority=1)

public void setGetWinSize(){

**//WebDriver setSize method used to set window size width = 300 and height = 500.**

driver.manage().window().setSize(new Dimension(300,500));

**//WebDriver getSize method used to get window width and height.**

System.out.println("Window height Is -> "+driver.manage().window().getSize().getHeight());

System.out.println("Window width Is -> "+driver.manage().window().getSize().getWidth());

}

@Test(priority=2)

public void setGetWinPosition(){

**//WebDriver setPosition method used to set window position x coordinate = 50 and y coordinate = 100.**

driver.manage().window().setPosition(new Point(50,200));

**//WebDriver getPosition method used to get window position x,y coordinates.**

System.out.println("Window position X coordinates Is -> "+driver.manage().window().getPosition().getX());

System.out.println("Window position Y coordinates Is -> "+driver.manage().window().getPosition().getY());

}

}

**69** : **I wants to scroll my software web application page by 300 pixel. Tell me how can i do it?**

**Answer** : We can use **javascript executor with window.scrollBy(x,y)** to scroll page in x or y directions.

# Scroll Down-Up Web Page Using Javascript Executor In Selenium Webdriver

Yes, We can **scroll down page** using **javascript executor In selenium** webdriver. If you remember, Earlier I have posted many different posts where we can use javascript executor to perform different actions on web page like [**Highlighting Element**](http://www.software-testing-tutorials-automation.com/2014/01/how-to-highlight-element-using-selenium.html), [**Generating Alert**](http://www.software-testing-tutorials-automation.com/2014/01/generating-alert-in-selenium-webdriver.html), [**Get Domain Name**](http://www.software-testing-tutorials-automation.com/2014/01/selenium-webdriver-get-domain-name.html), [**Get Page Title**](http://www.software-testing-tutorials-automation.com/2014/01/executing-javascript-in-selenium.html) etc.. Now If you needs to scroll down the page then you can use **Javascript Executor** In Selenium Webdriver test.

**Why need to scroll down page ?**

If you see some websites specially e-commerce website, More products will be displayed only If scroll down the page. Same thing on facebook website. More posts will be loaded only If you scroll down the page. In this kind of scenario, You can use javascript executor to scroll down page If you wants to take some action on page element which appears on scroll down. You can check presence of scroll as described In[**THIS POST**](http://www.software-testing-tutorials-automation.com/2015/02/how-to-verify-scroll-present-on-browser.html).

**Scroll by given pixel offset**

I have created test script to scroll down-up web page In horizontal or vertical direction baser on given x y pixel offset.

Bellow given example will first scroll down(X direction) web page by 600 pixels and then It will scroll up It by 300 pixels.

package Testing\_Pack;

import java.io.IOException;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class Scrolling {

WebDriver driver;

@BeforeTest

public void setup() throws Exception {

driver =new FirefoxDriver();

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

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

driver.get("http://only-testing-blog.blogspot.in");

}

@Test

public void Scroll\_Page() throws IOException, InterruptedException {

**//To scroll down web page by 600 pixels In x(vertical) direction.**

**//You can y parameter to scroll page In horizontal direction.**

JavascriptExecutor javascript = (JavascriptExecutor) driver;

javascript.executeScript("window.scrollBy(0,600)", "");

Thread.sleep(3000);

**//To scroll up web page by 300 pixels In x(vertical) direction.**

javascript.executeScript("window.scrollBy(0,-300)", "");

}

}

**Scroll down to bottom of page**

If you wants to scroll down to bottom of the page, You can use bellow given @Test method In above example.

@Test

public void Scroll\_Page() throws IOException, InterruptedException {

**//Scroll down to bottom of the page.**

JavascriptExecutor javascript = (JavascriptExecutor) driver;

javascript.executeScript("window.scrollTo(0, document.body.scrollHeight)", "");

}

**Scroll to element In selenium WebDriver**  
If you wants to scroll till some element on page then you can use element parameter In javascript as shown In bellow example.

**//Scroll till element.**

JavascriptExecutor je = (JavascriptExecutor) driver;

WebElement element = driver.findElement(By.xpath("//div[@id='dragdiv']"));

je.executeScript("arguments[0].scrollIntoView(true);",element);

**70. How to switch back from a frame ?**

Ans- use method defaultContent().

|  |  |
| --- | --- |
| 1 | Syntax – driver.switchTo().defaultContent(); |

**71. What is the use of AutoIt tool ?**

Ans- Some times while doing testing with selenium, we get stuck by some interruptions like a window based pop up. But selenium fails to handle this as it has support for only web based application. To overcome this problem we need to use AutoIT along with selenium script. AutoIT is a third party tool to handle window based applications. The scripting language used is in VBScript.

**72.How to perform double click using WebDriver ?**

Ans- use doubleClick() method.

|  |  |
| --- | --- |
| 1  2  3 | Syntax- Actions act = new Actions(driver);    act.doubleClick(webelement); |

**73.What is the use of contextClick() ?**

Ans- It is used to right click.

**74.What is the difference b/w getWindowHandles() and getWindowHandle() ?**

Ans- getWindowHandles()- is used to get the address of all the open browser and its return type is Iterator<String>.

getWindowHandle()- is used to get the address of the current browser where the conrol is and return type is String.

**75.What are different components of your framework ?**

Ans- Library- Assertion, ConfigLibrary, GenericLibrary, ProjectSpecificLibrary, Modules.

Drivers folder, Jars folder, excel file.

**76.How to check all checkboxes in a page ?**

Ans-

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | List&lt;webElement&gt; chkBox = driver.findElements(By.xpath(“//htmltag[@attbute='checkbox']”));    for(int i=0; i&lt;=chkBox.size(); i++){    chkBox.get(i).click();    } |

**77.Count the number of links in a page.**

Ans- use the locator By.tagName and find the elements for the tag //a then use loop to count the number of elements found.

|  |  |
| --- | --- |
| 1  2  3  4  5 | Syntax- int count = 0;    List&lt;webElement&gt; link = driver.findElements(By.tagName(“a”));    System.out.println(link.size()); // this will print the number of links in a page. |

**78.What is main difference between RC and webdriver ?**

Ans- Selenium RC injects javascript function into browsers when the web page is loaded.

Selenium WebDriver drives the browser using browser’s built-in support.

**79.Why you choose webdriver over RC ?**

Ans-

a) Native automation faster and a little less prone to error and browser configuration,

b) Does not Requires Selenium-RC Server to be running

c) Access to headless HTMLUnitDriver can allow really fast tests

d) Great API etc.

**80. Which one is better xpath or CSS ?**

Ans- xpath.

**81. How will you handle dynamic elements ?**

Ans- By writing relative xpath.

**82.what are the different assertions or check points used in your script ?**

Ans- The common types of validations are:

a) Is the page title as expected,

b) Validations against an element on the page,

c) Does text exist on the page,

d) Does a javascript call return an expected value.

**83. What is actions class in WebDriver ?**

Ans- Actions class is used to control the actions of mouse.

**84. How do you read data from excel ?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | FileInputStream fis = new FileInputStream(“path of excel file”);    Workbook wb = WorkbookFactory.create(fis);    Sheet s = wb.getSheet(“sheetName”);    String value = s.getRow(rowNum).getCell(cellNum).getStringCellValue(); |

**85.** **What is the use of xpath ?**

Ans- it is used to find the WebElement in web page. It is very useful to identify the dynamic web elements.

**86.** **What are different types of locators ?**

Ans- There are 8 types of locators and all are the static methods of the By class.

By.id(), By.name(), By.tagName(), By.className(), By.linkText(), By.partialLinkText(), By.xpath, By.cssSelector().

**87.** **What is the difference between Assert and Verify?**

Ans- Assert- it is used to verify the result. If the test case fail then it will stop the execution of the test case there itself and move the control to other test case.

Verify- it is also used to verify the result. If the test case fail then it will not stop the execution of that test case.

**88.** **What is the alternate way to click on login button?**

Ans- use submit() method but it can be used only when attribute type=submit.

**89.** **How do you verify if the checkbox/radio is checked or not ?**

Ans- We can use isSelected() method.

Syntax –

|  |  |
| --- | --- |
| 1 | driver.findElement(By.xpath("xpath of the checkbox/radio button")).isSelected(); |

If the return value of this method is true then it is checked else it is not.

**90.** **How do you handle alert pop-up ?**

Ans- To handle alert pop-ups, we need to 1st switch control to alert pop-ups then click on ok or cancle then move control back to main page.

Syntax-

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | String mainPage = driver.getWindowHandle();    Alert alt = driver.switchTo().alert(); // to move control to alert popup    alt.accept(); // to click on ok.    alt.dismiss(); // to click on cancel.    //Then move the control back to main web page-    driver.switchTo().window(mainPage); → to switch back to main page. |

**91** **How do you click on a menu item in a drop down menu?**

Ans- If that menu has been created by using select tag then we can use the methods selectByValue() or selectByIndex() or selectByVisibleText(). These are the methods of the Select class.

If the menu has not been created by using the select tag then we can simply find the xpath of that element and click on that to select.

**92** **How do you simulate browser back and forward ?**

|  |  |
| --- | --- |
| 1  2  3 | driver.navigate().back();    driver.navigate().forward(); |

**93.** **What is the difference between ‘/’ and ‘//’ ?**

Ans- //- it is used to search in the entire structure.

/- it is used to identify the immediate child.

**94.** **Write the code for Reading and Writing to Excel through Selenium ?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | FileInputStream fis = new FileInputStream(“path of excel file”);    Workbook wb = WorkbookFactory.create(fis);    Sheet s = wb.getSheet("sheetName");    String value = s.getRow(rowNum).getCell(cellNum).getStringCellValue(); // read data    s.getRow(rowNum).getCell(cellNum).setCellValue("value to be set"); //write data    FileOutputStream fos = new FileOutputStream(“path of file”);    wb.write(fos); //save file |

**95.** **What is a Framework ?**

Ans- A framework is set of automation guidelines which help in

Maintaining consistency of Testing, Improves test structuring, Minimum usage of code, Less Maintenance of code, Improve re-usability, Non Technical testers can be involved in code, Training period of using the tool can be reduced, Involves Data wherever appropriate.

There are five types of framework used in software automation testing:

1-Data Driven Automation Framework

2-Method Driven Automation Framework

3-Modular Automation Framework

4-Keyword Driven Automation Framework

5-Hybrid Automation Framework , its basically combination of different frameworks. (1+2+3).

**96.** **How to get the number of frames on a page ?**

|  |  |
| --- | --- |
| 1  2  3 | List &lt;WebElement&gt; framesList = driver.findElements(By.xpath("//iframe"));    int numOfFrames = frameList.size(); |

**97.** **How to check if an element is visible on the web page ?**

Ans- use isDisplayed() method. The return type of the method is boolean. So if it return true then element is visible else not visible.

|  |  |
| --- | --- |
| 1 | driver.findElement(By.xpath("xpath of elemnt")).isDisplayed(); |

**98.** **How to check if a button is enabled on the page ?**

Ans- Use isEnabled() method. The return type of the method is boolean. So if it return true then button is enabled else not enabled.

|  |  |
| --- | --- |
| 1 | driver.findElement(By.xpath("xpath of button")).isEnabled(); |

**99.** **How to check if a text is highlighted on the page ?**

Ans- To identify weather color for a field is different or not-

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | String color = driver.findElement(By.xpath("//a[text()='Shop']")).getCssValue("color");    String backcolor = driver.findElement(By.xpath("//a[text()='Shop']")).getCssValue("background-color");    System.out.println(color);    System.out.println(backcolor); |

Here if both color and back color different then that means that element is in different color.

**100.** **How to check the checkbox or radio button is selected ?**

Ans- Use isSelected() method to identify. The return type of the method is boolean. So if it return true then button is selected else not enabled.

driver.findElement(By.xpath("xpath of button")).isSelected();

**101.** **How to get the title of the page ?**

Ans- Use getTitle() method.

|  |  |
| --- | --- |
| 1 | Syntax- driver.getTitle(); |

**102.** **How do u get the width of the textbox ?**

|  |  |
| --- | --- |
| 1  2  3 | driver.findElement(By.xpath(“xpath of textbox ”)).getSize().getWidth();    driver.findElement(By.xpath(“xpath of textbox ”)).getSize().getHeight(); |

**103.** **What is the use of getOptions() method ?**

Ans- getOptions() is used to get the selected option from the dropdown list.

**104.How will you handle alert?**

**Ans:**

* **Actually** “Web Driver” not control the Java script alert
* Hence we used the ‘Alert’ interface for ‘Java script’ control
* “Alert” is nothing but a interface it will react based on input coming from the web driver
* Web driver done the some action based on the scenario and send to the ‘Alert’ interface with through the variable
* Based on the ‘Web driver’ inputs ‘Alert’ interface react
* Alert interface have multiple methods with in this

**Syntax:**

Alertat =wd.switchTo().alert();

Below the details for above syntax:

* “wd.switchTo().alert();” this section is done by the web driver and out value of this actuion will store in the variable
* “at “ – This is the user defined variable it name will change based on the user understanding. This is used to store the result of the ‘web driver’ for ‘Alert’ interface
* Alert – it’s an interface it will react based on the input which is stored in the variable

**105.How to verify tooltip for a web element?**

**Ans:**

WebElement tooltip=wd.findElement(By.xpath("html/body/p[4]/input"));

         String data=tooltip.getAttribute("title");

    System.out.println(data);

**106.How to enter a text box inside Frame2 which is available inside frame1 and that is available inside a webpage**

**Ex.**

Frame1

txtbx

Frame 2

webpage

**Ans:**

We can use the ‘switchto().frame()’ methods to find the textbox which is located in the frame with index (or) id (or) name etc. find below the code with index

Sk.switchTo().frame(1);

**107.How to click all links within a page?**

**Ans:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**publicclass** Getalllinks {

**publicstaticvoid** main(String[] args) **throws** InterruptedException {

// **TODO** Auto-generated method stub

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

sk.get("http://rgtdblrpt/rvw");

Thread.*sleep*(2000L);

List<WebElement>link=sk.findElement(By.*tagName*("a"));

**for**(**int**i=0;i<link.size();i++){

String a=link.get(i).getText();

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

}

}

}

**108.What kind of build configuration tool you are using?**

**Ans:**

Maven

**109.How will you decide whether to automate or not?**

**Ans:**

Based on the following thinks we decide the selected project can able to automate or not

* Requirement should not change regularly
* Project should be long term
* Project should be a web application
* Project have more functional also that functionality are regularly used

**When we start the Automation Testing:**

Most of the projects not fully cover by the automation. Major functionality only cover in the automation and basic smoke and sanity cases are done by manual testing only.

Before start the automation testing we need to check the following thinks

* System should be stable to test
* First round of function testing and retesting should be completed in the manual testing
* At the time of Regression testing need to start the ‘Automation’

**110.How will you take screenshot**

**Ans:**

* We can take the screenshot with using ‘Takesscreenshot’ interface
* We need to wrap this interface with webdriver

**Syntax:**

File Screenshot=((TakesScreenshot)wd).get ScreenshotAs(OutputType.File);

**111.How to check a webelement present in a browser which is actually visible but not editable or enabled?**

**Ans:**

* We can cable test with ‘isEnabled()’ method to verify the selected element is visible or not
* **isEnabled()** webdriver method will verify and return true if specified element is enabled. Else it will return false. Generic syntax to store and print element's status value is as bellow.

**Syntax:**

############################################################

boolean fname = driver.findElement(By.xpath("//input[@name='fname']")).isEnabled();

System.out.print(fname);

**112.How to run same test in different machines?**

**Ans:**

Can able to run the same test in different machine with using ‘Selenium Grid”

**For Security site:**

* Some of the sites are secure site they need confirmation for the site view on that time we need to by-pass the alert so we use following code

##################################################

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

  // TODO Auto-generated method stub

    FirefoxProfile prof=new FirefoxProfile();

    prof.setAcceptUntrustedCertificates(true);

    WebDriver wd=new FirefoxDriver(prof);

    wd.get("<http://www.cacert.org/>");

    Thread.sleep(2000L);

    wd.findElement(By.xpath("html/body/div[1]/div[3]/div/p[3]/a")).click();

    Thread.sleep(1000L);

   }

**113.Difference between implicitly wait vs explicitly wait?**

**Ans:**

**Implicitly wait:**

* It’s webdriver wait
* It will applicable to next line from where we give
* Time unit seconds only applicaple

#######################################################

i.e.:

Line 1;

wd.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

Line2 ;

#######################################################

* As per above example webdriver check the result of first line code in first second of the given time unit
* Then finally check the end time only (as per our example will check on 1st second and 30th second only)
* Webdriver will wait for the given time (30 sec) if line1 is reached between the given time (15 sec) – This is the disadvantages of the implicitly wait

**Explicitly wait:**

* It’s also a webdriver wait
* We need to give where we want
* We need to give the time period in ‘polling’ method
* Second only allow in the time period

i.e:

#####################################################################

WebDriverWait wt=new WebDriverWait(wd, 30);

WebElement tp=wd.findElement(By.xpath(".//\*[@id='main']/div[3]/button"));

wt.until(ExpectedConditions.elementToBeClickable(locator).visibilityOfElementLocated(By.xpath(".//\*[@id='main']/div[2]/button")));

  wt.pollingEvery(3, TimeUnit.SECONDS);

#####################################################################

* In above example we need to give the time period
* Then give the element locators in ‘until’ method
* After that need to give interval period in ‘polling’ method
* Based on the polling period webdriver check the code by given time period
* If reached any point then move to next line execution if not reach wait for the total given time period after that only error throw

**114.What is the page factory & Page object model? Y we need to use over others?**

**Ans:**

**Page objects Model:**

* Page Object Model is a design pattern to create Object Repository for web UI elements.
* Under this model, for each web page in the application there should be corresponding page class.
* This Page class will find the WebElements of that web page and also contains Page methods which perform operations on those WebElements.
* Name of these methods should be given as per the task they are performing i.e., if a loader is waiting for payment gateway to be appear, POM method name can be waitForPaymentScreenDisplay()

**Page Factory:**

* Page Factory is an inbuilt page object model concept for Selenium WebDriver but it is much optimized.
* Here as well we follow the concept of separation of Page Object repository and Test methods. Additionally with the help of PageFactory class we use annotations **@FindBy** to find WebElement. We use initElements method to initialize web elements

**@FindBy** can accept **tagName, partialLinkText, name, linkText, id, css, className, xpath**as attributes.

**Advantages of POM:**

* Page Object Patten says operations and flows in the UI should be separated from verification. This concept makes our code cleaner and easy to understand.
* Second benefit is the object repository is independent of testcases, so we can use the same object repository for a different purpose with different tools. For example, we can integrate POM with TestNG/JUnit for functional testing and at the same time with JBehave/Cucumber for acceptance testing.
* Code becomes less and optimized because of the reusable page methods in the POM classes.

Methods get more realistic names which can be easily mapped with the operation happening in UI. i.e. if after clicking on the button we land on the home page, the method name will be like 'gotoHomePage()'.

############################################################################

**115.Where your dependencies will be stored?**

**Ans:**

Dependences stored in POM.xml in Maven project

**116.How to handle multiple popup windows? How will you return to actual window?**

**Ans:**

**Step 1:** After opening the website, we need to get the main window handle by using driver.getWindowHandle();  
The window handle will be in a form of lengthy alpha numeric  
**Step 2:** We now need to get all the window handles by using driver.getWindowHandles();  
**Step 3:** We will compare all the window handles with the main Window handles and perform the operation the window which we need.

Sample Program:

############################################################################

public class TestNaukri {

@Test

public void TestPopUp() throws InterruptedException{

// Open browser

WebDriver driver=new FirefoxDriver();

// Maximize browser

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

// Load app

driver.get("http://www.naukri.com/");

// It will return the parent window name as a String

String parent=driver.getWindowHandle();

// This will return the number of windows opened by Webdriver and will return Set of St//rings

Set<String>s1=driver.getWindowHandles();

// Now we will iterate using Iterator

Iterator<String> I1= s1.iterator();

while(I1.hasNext())

{

   String child\_window=I1.next();

// Here we will compare if parent window is not equal to child window then we            will close

if(!parent.equals(child\_window))

{

driver.switchTo().window(child\_window);

System.out.println(driver.switchTo().window(child\_window).getTitle());

driver.close();

}

}

// once all pop up closed now switch to parent window

driver.switchTo().window(parent);

}

}

**117.What kind of metrics you collected after & before your automation test?**

**Ans:**

**What is a metric?** Metric is a standard of measurement. A metric is a measurement criteria that helps measure past and present performance and/or predict future performance.  
Most of the Metrics have similar following category of which they are comprised of:  
**Quality:** Purposeful/meaningful measured results of test/execution which represent the product quality. Some of the examples are: Defects logged, Usability, performance, scalability and customer satisfaction.  
**Progress:** Specific parameters which help identify the test progress are compared against the success criteria, are collected iteratively over time which can be later on used to represent the progress in epitomized way.  
**Coverage:** In order to measure the test scope and success, meaningful parameters called Coverage is required.

**What is ATM and how good it should be?**  
Automated Test Metrics are used to measure the past, present and future performance of its process and the relevant efforts been put. A metrics stands good only if it is related to the performance of the effort and it can only happen if there are clearly defined goals pertaining to the automation effort. A well defined automated Test metric has following characteristics:  
-Its clearly measurable  
-Its purposeful/meaningful  
-Epitomized data/Graphical data representation is derived from easy collected data  
-It should be valid input criteria for Automation improvement.  
-Importantly, it should be simple.

**Automation Test Coverage**  
Automation Test Coverage metrics actually determines via test execution results that whether we have actually achieved what we had covered in past during automation of test cases.  
Altogether with manual test coverage, against the total number of tests this type of metric can measure the completeness of the test coverage and can measure how much automation is being executed.

ATC  %    =    Automation Coverage / Total Coverage

Here ‘Total coverage’ means, requirements, units/components, or code coverage.

**Automation Progress**  
Automation Progress points towards the progress of Number of Test cases automated out of the total number of test cases which are automatable.

AP %    =    Number of test cases automated / Number of test cases automatable

The automation progress is tracked over the time, as the automation progresses against the defined Automation milestone tasks, the metric can present valuable data for the time it will take for the whole set of test cases to get automated.

**Automation Index**  
Every project have a sufficient duration, either has automatable test procedures or requires automation from scratch/begin. To fit either of these criteria’s, Automation index or Percent Automatable can be defined:

AI or PA %    =    Number of test cases automatable / Total Number of test cases

**Defect Density**  
Defect measurement is vital for both manual and automated process. If any module/ component of application under test it detected with high density of defects, then it automatically becomes prone to retesting using automation. Defect density is measured using the metrics, the total known defects divided by the size of the software entity being measured.

DD    =    No. Of Known Defects / Size of the Software/Application

**Defect Aging**  
Defect Aging is the difference between – the date when a defect is detected and the current date (if the defect is still open) or the date the defect was fixed (if the defect is already fixed).

DA ( in Time)    =    Defect Fix Date (OR Current Date)   –   Defect Detection Date

**Defect Trend Analysis**  
Defect trend analysis is the trend of defects found over time. This trend truly shows the health/stability of the project. Defect trend is considered to be improving if the no. of defects is reducing over a time, trend is considered to be worse if the defects are increasing. In an agile methodology where the test deliverables needs to be fit in iterative cycles, if the defects count for component/module is reducing in progressive cycles, then it helps in determining the closure of the feature test.

DTA    = No. Of Known Defects / No. Of test Procedures Executed

**Defect removal Efficiency**  
Defect removal efficiency is measured in the percentage of effectiveness of defect removal efforts. This metric if used in collaboration with Automation definitely helps improve the quality of the product. Greater the percentage value, better the quality of the product.

DRE    =    (No. Of Defects found during testing ) /(  ( No. Of Defects found during testing + No. of Defects found after delivery ) )

**118 .Which one is current in the following statement?**

**WebDriver wd=new FirfoxDriver();**

**FirefoxDriver =new FirefoxDriver();**

**Ans:**

Both are correct

* In first statement ‘Webdriver’ its an interface it have set of rules for the all browsers so can handle all browsers with using that interface
* But in second statement ‘FirefoxDriver’ it’s an interface it have only firefox related rules only
* Hence if I used the ‘WebDriver’ I can handle the all browsers with this interface