**4+’klww**

**46et’65k`Automation Testing**

In **automation testing**, the test automation engineer will write the test script or use the automation testing tools to execute the application. On the other hand, in manual testing, the test engineer will write the test cases and implement the software on the basis of written test cases.

In other words, we can say that the main concentration of **test Automation** is to change the manual human activity with systems or devices.

**Advantages:**

* Reusability
* Consistency
* Running tests anytime (24/7)
* Early Bug detection
* Less Human Resources

**Selenium**

Whenever we are talking about the open-source tools of automation testing, one name came into every automation test engineer's mind: Selenium.

Selenium is an open-source tool which means, it does not require any licensing.

We can only test the web application using the selenium tool, and the Stand-alone application cannot automate in Selenium.

It is most commonly used to implement Functional test scripts.

It can be associated with many devices such as **TestNG and JUnit** to manage the test cases and generate the test reports.

Selenium can be integrated with frameworks like Ant and Maven for source code compilation.

Selenium can also be integrated with testing frameworks like TestNG for application testing and generating reports.

Selenium can be used to automate functional tests and can be integrated with automation test tools such as **Maven, Jenkins,**  to achieve continuous testing. It can also be integrated with tools such as **TestNG, &JUnit** for managing test cases and generating reports.

Selenium Limitation:

* Selenium does not support automation testing for desktop applications.
* We can't perform automation tests on web services like SOAP or REST using Selenium.
* We should know at least one of the supported programming languages to create tests scripts in Selenium WebDriver.
* It is not possible to perform testing on images. We need to integrate Selenium with **Sikuli** for image-based testing

**Selenium WebDriver:**

Selenium WebDriver is most important component of selenium suite.

Selenium WebDriver provides a programming interface to create and execute test case.

Selenium WebDriver directly calls the method of different browser so we have separate driver for each browser.

* Mozilla Firefox Driver (Gecko Driver)
* Google Chrome Driver
* Internet Explorer Driver
* Opera Driver
* Safari Driver

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

Here, "WebDriver" is an interface, and we are creating a reference variable "driver" of type WebDriver, instantiated using " **ChromeDriver** " class.

Selenium is not just a single tool but a suite of software's, each having a different approach to support automation testing. It comprises of four major components which include:

1. Selenium Integrated Development Environment (IDE)
2. Selenium Remote Control (Now Deprecated)
3. WebDriver
4. Selenium Grid

Selenium IDE is implemented as Firefox extension which provides record and playback functionality on test scripts. It allows testers to export recorded scripts in many languages like HTML, Java, Ruby, RSpec, Python, C#, JUnit and TestNG. Selenium IDE has limited scope, and the generated test scripts are not very robust, and portable.

Different types of testing's that we can achieve through Selenium are.

* Functional Testing
* Regression Testing
* Sanity Testing
* Smoke Testing
* Responsive Testing
* Cross Browser Testing
* UI testing (black box)
* Integration Testing

**What are the various Components of Selenium?**

There are various components of Selenium. It can work on browsers like Chrome, Firefox, Safari, edge and so on. It also supports more than one operation system like Window, Mac, Linux and so on.

Selenium has below components:

1. Selenium IDE.
2. Selenium RC.
3. Selenium Webdriver.
4. Selenium Grid.

**Selenium IDE:**

Selenium Integrated Development Environment is an important part of the selenium suite. It was first developed as Firefox plugin, however now it is available both on Firefox and Chrome.

Some features are:

* Recording, debugging, and editing of the functional test can be done in Selenium IDE.
* Test scripts in Selenium are developed in Selenese which is scripting language in Selenium.
* Selenium commands help us to perfrom tasks like clicking a button or link, taking input in edit box, obtaining a text from a web elements.

**Selenium RC**

Selenium Remote Control is a server implemented in Java. It can accept commands for browsers using the HTTP.

Some of the features of Selenium RC −

* Automation tests in Selenium RC can be developed in any programming languages like Java, Python, C#and so on.
* To initiate test execution, we have to create an instance of the Selenium RC server.

**Selenium WebDriver:**

Selenium webdriver was developed after Selenium RC. It receives commands and passes them to the browser. This is done with the help of the browser drivers that sends commands to the browser and obtains the results.

Some of the features of Selenium RC −

* Automation tests can be written in multiple programming languages like the Java, C#, Python, JavaScript, and so on.
* Selenium webdriver supports browsers like Chrome, Firefox, Safari, IE, and so on.
* Selenium webdriver works on more than one platforms like Windows, Mac, Linux, Android, and so on.
* Headless execution can be achieved with the help of the HTMLUnit Driver.
* Selenium webdriver does not require a server to initiate test execution and it communicates directly with the browser.
* Selenium webdriver is open-source and comes without any licensing cost.

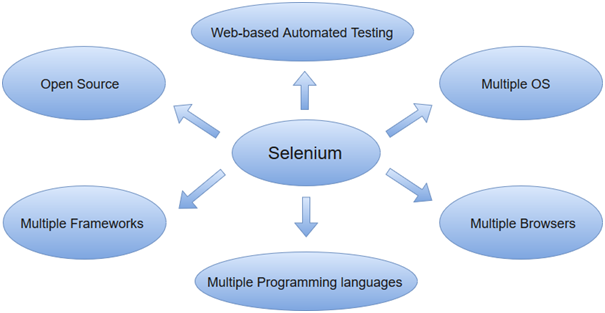
Selenium web Driver is an Automation Tool. With the help of selenium web driver, we can automate web applications. Selenium web driver is supports different languages like: Java, C#, Python and all. And same selenium web driver to support different – different browsers like- chrome, Firefox, i.e., Safari and others.

**Selenium Grid:**

Selenium Grid is mainly used for parallel testing. It enables us to execute varied tests in multiple machines simultaneously.

Some of the features of Selenium Grid −

* Presence of a hub machine which directs the execution on multiple machines. Also, there are more than one node machines where the actual execution takes place.
* Selenium Grid supports testing on multiple browsers and platforms.
* Reduces execution time by allowing parallel execution of tests.



Selenium WebDriver Locating elements:

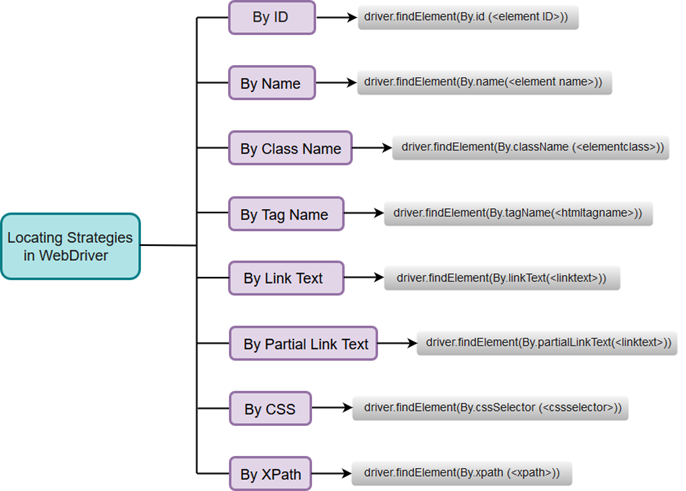
Locating web elements in WebDriver is performed with the help of findElement() and findElements() method.

**What do you mean by XPath?**

XPath is also defined as XML Path. It is a language used to query XML documents. It is an important approach to locate elements in Selenium. XPath consists of a path expression along with some conditions. Here, we can easily write XPath script/query to locate any element in the webpage. It is developed to allow the navigation of XML documents. The key factors that it considered while navigating are selecting individual elements, attributes, or some other part of an XML document for specific processing. It also produces reliable locators. Some other points about XPath are as follows.

* XPath is a language used for locating nodes in XML documents.
* XPath can be used as a substitute when you don't have a suitable id or name attribute for the element you want to locate.
* XPath provides locating strategies like:
  + XPath Absolute
  + XPath Attributes

In selenium we have 8 types of locators, by these locators we can perform actions over the web element that are the present of web applications.



Locators:

1. Xpath
2. CSS
3. Class name
4. Id
5. Tag name
6. Link text
7. Partial link text
8. Name

Xpath is also two types-

 Relative Xpath

 Absolute Xpath

1. Relative Xpath- is always start with the (//). And in relative xpath we can create using` different – different attributes. And also, we have multiple method using Xpath-

* Like-
* Text()
* Contains()
* Following
* Sibling
* Ancestor
* And
* Or

2. Absolute Xpath: is always start with (/). We don’t prefer absolute xpath. We only prefer relative Xpath.

**Explain XPath Absolute and XPath attributes.**

**XPath Absolute:**

* XPath Absolute enables users to mention the complete XPath location from the root HTML tag to the specific elements.
* Syntax: //html/body/tag1[index]/tag2[index]/.../tagN[index]
* Example: //html/body/div[2]/div/div[2]/div/div/div/fieldset/form/div[1]/input[1]

**XPath Attributes:**

* XPath Attributes is always recommended when you don't have a suitable id or name attribute for the element you want to locate.
* Syntax: //htmltag[@attribute1='value1' and @attribute2='value2']
* Example: //input[@id='passwd' and @placeholder='password']

**What are the WebDriver supported Mobile Testing Drivers?**

WebDriver supported "mobile testing drivers" are:

* AndroidDriver
* IphoneDriver
* OperaMobileDriver

**Handling Dynamic Xpath:**

If a simple [XPath](https://www.guru99.com/xpath-selenium.html) is not able to find a complicated web element for our test script, we need to use the functions from XPath 1.0 library. With the combination of these functions, we can create more specific XPath. Let’s discuss a 3 such functions –

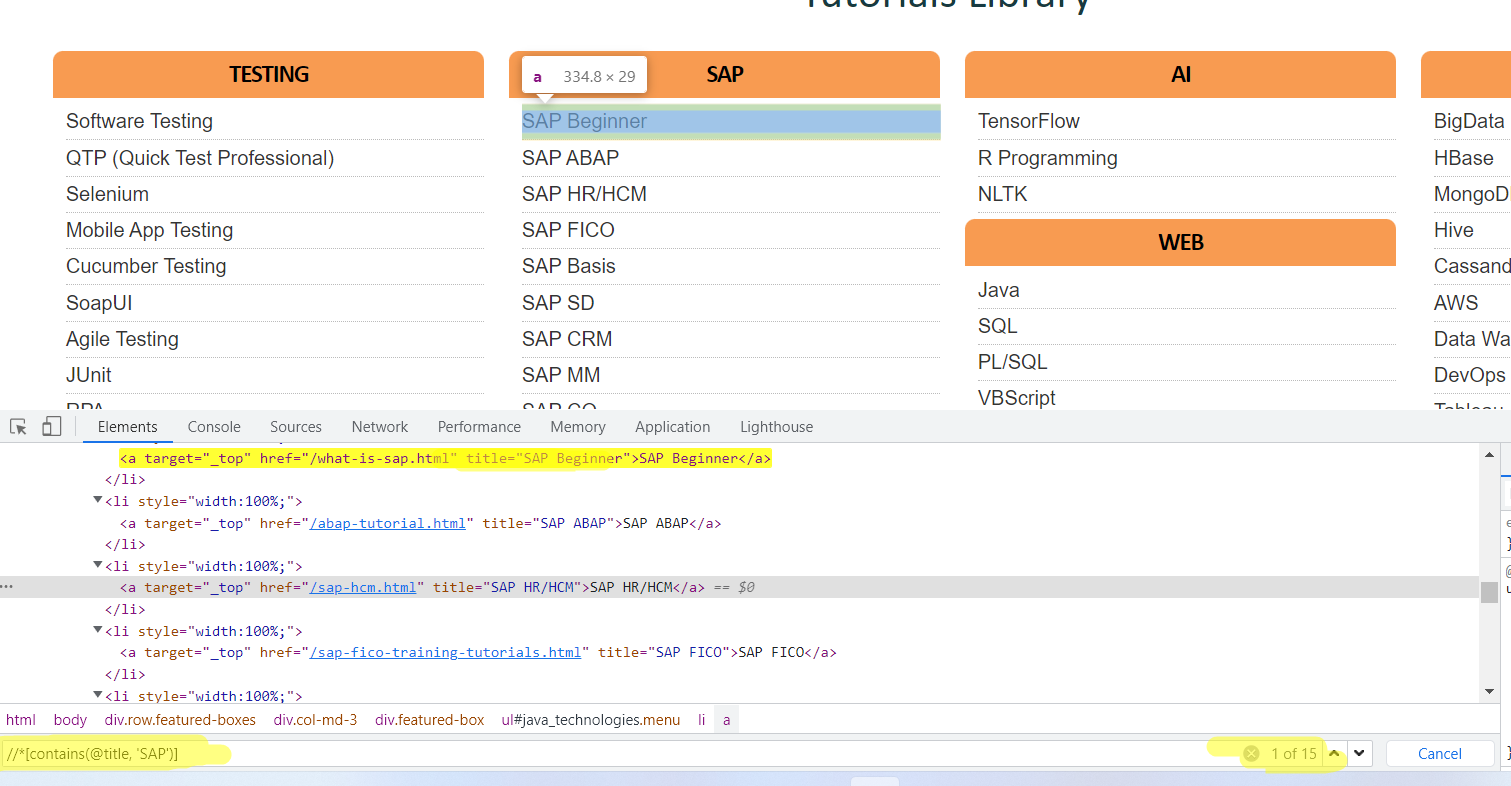
1. Contains
2. Sibling
3. Ancestor
4. And OR
5. Parent
6. Starts with
7. XPath Axes

**Contains()**

is a method used in XPath expression. It is used when the value of any attribute changes dynamically, for example, login information.

The contain feature has an ability to find the element with partial text as shown in below XPath example.

In this example, we tried to identify the element by just using partial text value of the attribute.



In above example: we have many links which have text SAP. So to find the xpath we have used xpath:

//\*[contains(@title, 'SAP')]

Here instead of title may be other attributes like id, name class etc.

**Using OR & AND:**

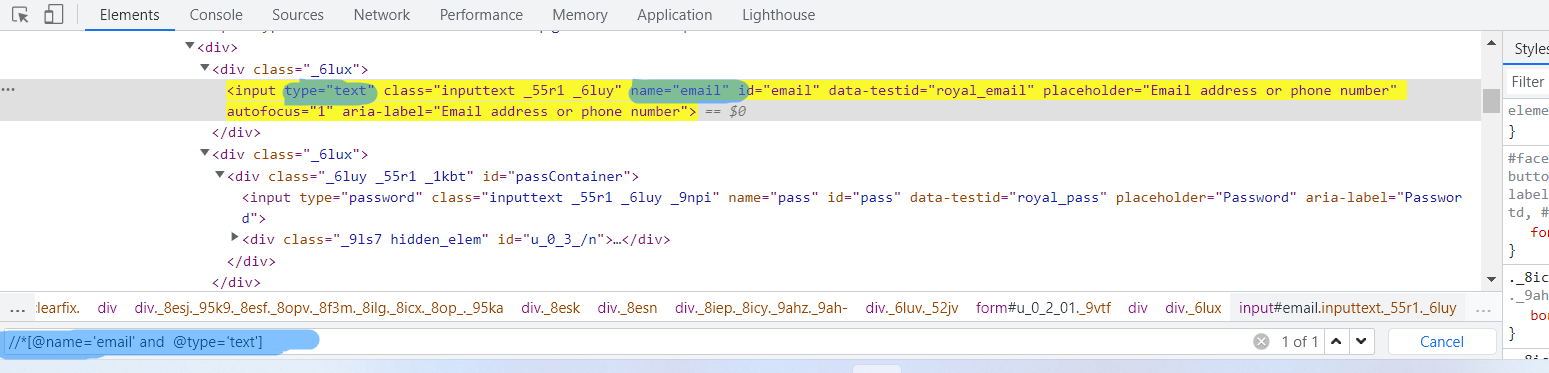
In OR expression, two conditions are used, whether 1st condition OR 2nd condition should be true. It is also applicable if any one condition is true or maybe both. Means any one condition should be true to find the element.

//\*[@name='email' or @id='pass'] here only 1 condition should be true



In AND expression, two conditions are used, both conditions should be true to find the element. It fails to find element if any one condition is false.

//\*[@name='email' and @type='text']



**Starts-with**

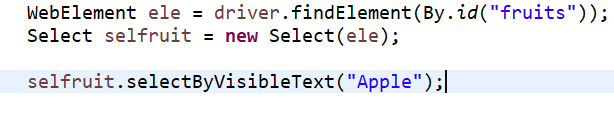
Using Starts-with function, you can find the element whose attribute dynamically changes on refresh or other operations like click, submit, etc.

//\*[starts-with(@id,'message')]

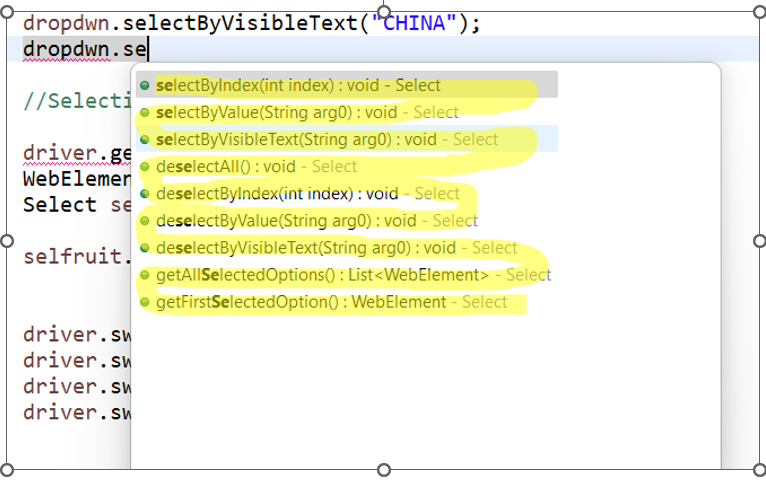
**Handling Drop Down Using Select Class:**

In Selenium, It defined a class **Select**which is used to select the dropdown value and implement the HTML SELECT tag . When we inspect the dropdown then we can see the Select tag in html view.

Example:



Other select methods are:



Please note that, we can use select class only when there is select tag available in DOM, if there is not select tag in DOM we will need to find other way.

**How to handle drop down without SELECT class in selenium OR how to handle drop down without div tag in selenium OR how to handle dropdown without <select> tag in selenium?**

Selenium Webdriver provides a ‘**Select’** class which can be only use where Select tag used for dropdown.

We cannot use Select class where dropdown is created without select tag. This is the limitation of Select class.

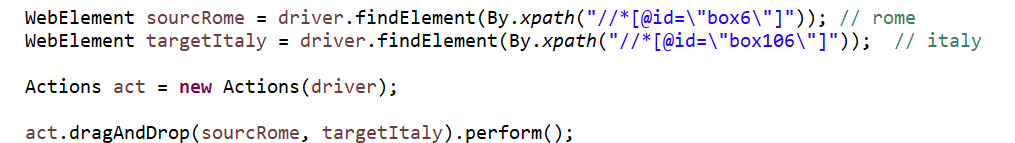
For handling this scenario, we will click on dropdown and get the all options then Iterate options using **for** loop and select desired options.

Look at below example:

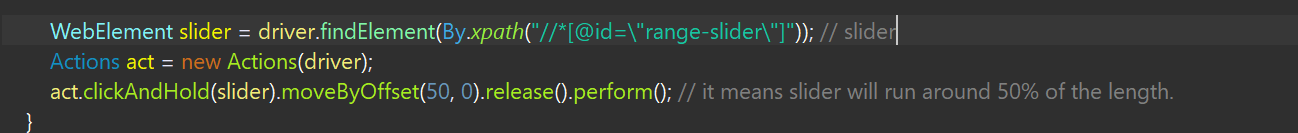


**Drag and Drop:**

We can do drag and drop using action class check below example:

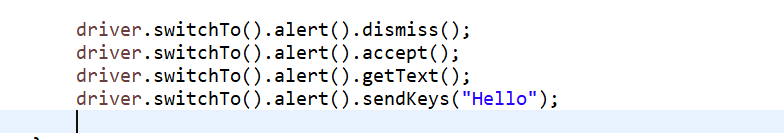


**Slider Using Action Class:**

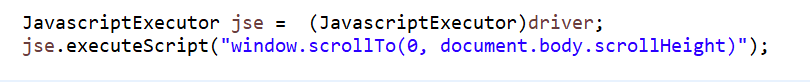


**Handling Alerts:**

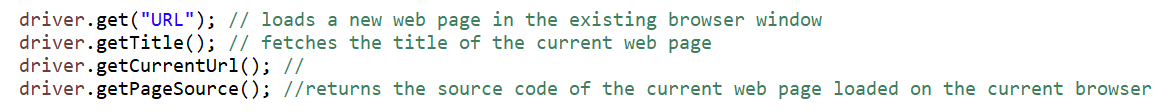
To click on cancel, OK, capture alert message, send data in alert box below are the commands:



**Scrolling Down to page:**



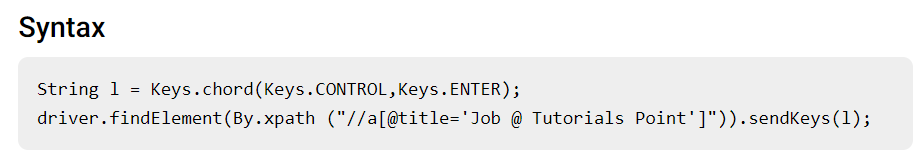
**Browser Commands:**



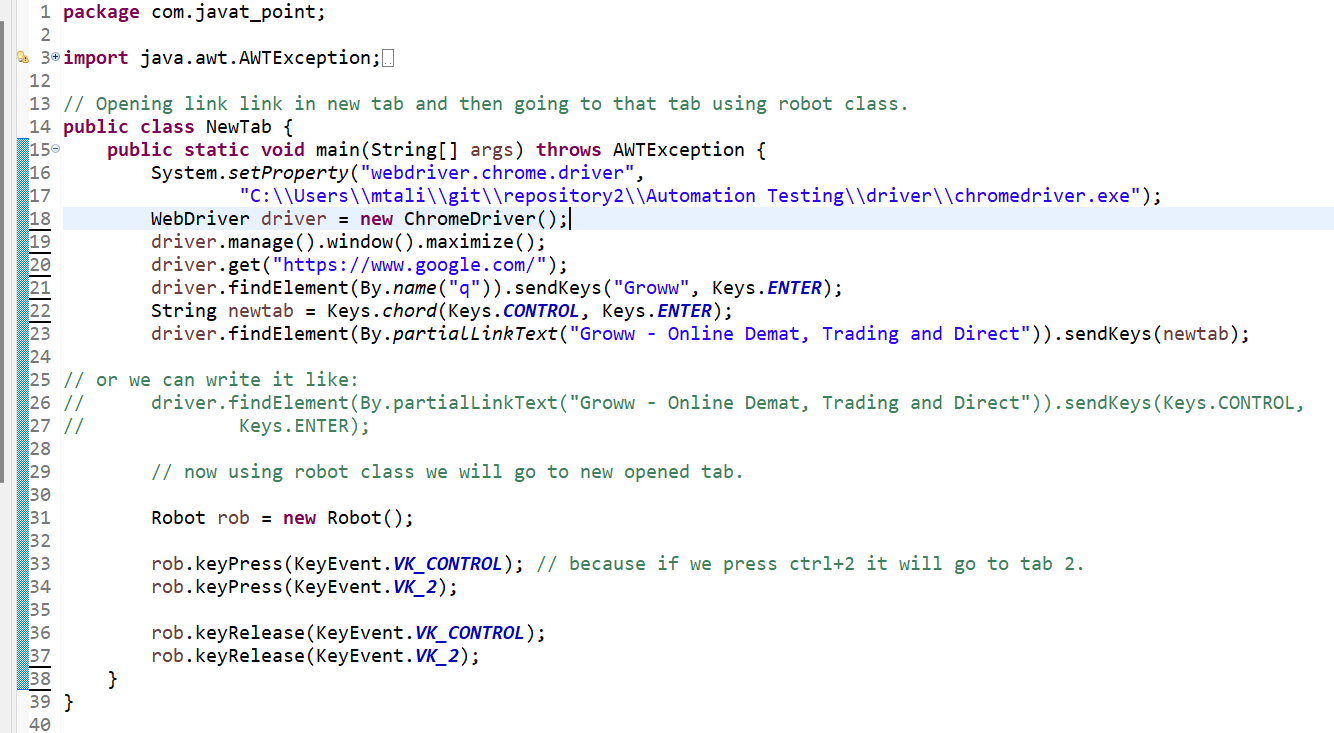
**Opening a link in new Tab:**

We can open a link in the new tab of Chrome browser using Selenium webdriver using the methods Keys.chord and sendKeys. The method Keys.chord is used to send multiple keys simultaneously as parameters.

To open a new tab, the Keys.CONTROL and Keys.ENTER are passed as parameters to the Keys.chord. Finally, the Keys.chord is passed as a parameter to the sendKeys.



Check below example:



**Selenium WebDriver Commands:**

**1. Fetching a web page**

There are two methods to fetch a web page:

Using Get method

driver.get("URL");

Using Navigate method

driver.navigate().to("URL");

**2. Locating forms and sending user inputs**

driver.findElement(By.*xpath*("xpath")).sendKeys("input");

**3. Clearing User inputs**

The clear() method is used to clear the user inputs from the text box.

driver.findElement(By.*name*("q")).clear();

**4. Fetching data over any web element**

Sometimes we need to fetch the text written over a web element for performing some assertions and debugging. We use getText() method to fetch data written over any web element.

driver.findElement(By.*id*("element567")).getText();

**5. Performing Click event**

The click() method is used to perform click operation on any web element.

driver.findElement(By.*id*("btnK")).click();

**6. Navigating backward in browser history**

driver.navigate().back();

**7. Navigating forward in browser history**

driver.navigate().forward();

**8. Refresh/ Reload a web page**

driver.navigate().refresh();

**9. Closing Browser**

driver.close();

**10. Closing Browser and other all other windows associated with the driver**

driver.quit();

**11. Moving between Windows**

driver.switchTo().window("windowName");

**13. Moving between Frames**

driver.switchTo().frame("frameName");

**14. Drag and Drop**

Drag and Drop operation is performed using the Action class.

WebElement element = driver.findElement(By.*name*("source"));

WebElement target = driver.findElement(By.*name*("target"));

Actions act = **new** Actions(driver);

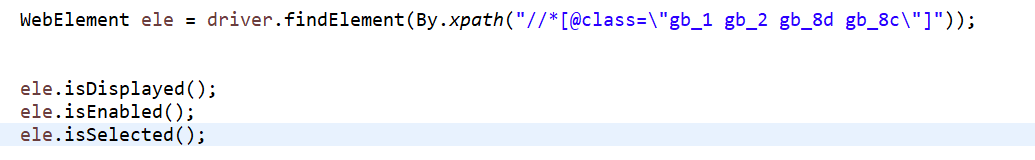
act.dragAndDrop(element, target).perform();

**WebElement Commands:**

The term web element refers to a HTML element. The HTML documents are composed of HTML elements. It consists **a start tag, an end tag** and the **content** in between. For instance, a HTML element is written as: "<tagname> content </tagname>"

In WebDriver, we have several commonly used web element commands and actions. The following screenshot displays the eclipse web element command panel.

Check below example, there are some other like sendkeys, .click etch.

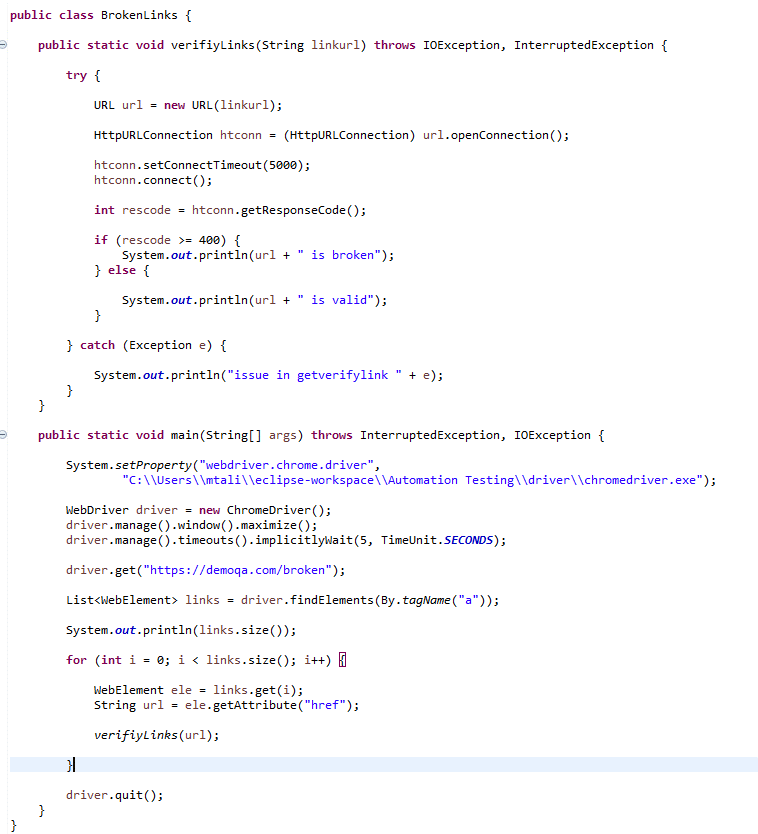


**Broken Links and Image:**

When someone clicks on such a link, sometimes an error message is displayed like a page not found. There may not be any error message at all.  These are essentially invalid HTTP requests and have ***4xx*** and ***5xx*** status code. Some common reasons for a broken link on a webpage can be

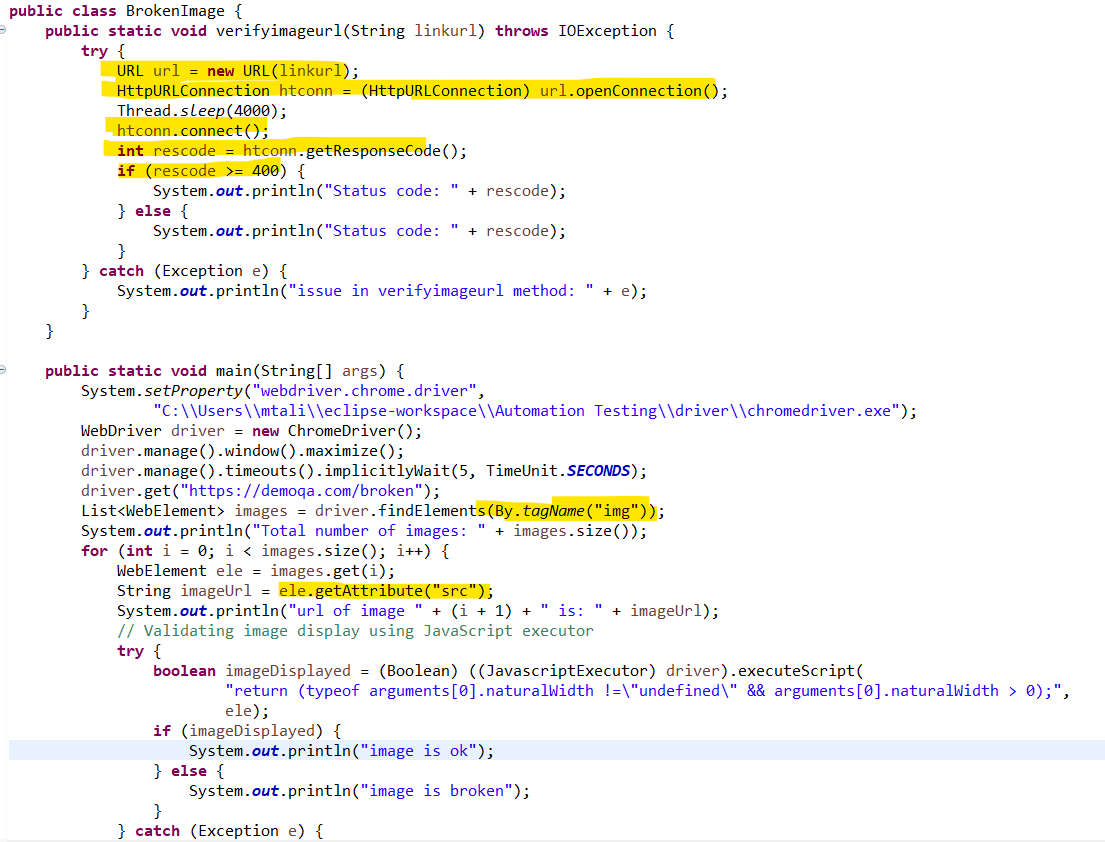
To find broken links, first we have to find all the links on a webpage. The hyperlinks are generally implemented on a web page using the HTML Anchor (*<a>*) tag. So, if you identify and locate all the anchor tags on a web page and then get the corresponding URLs, we will be able to traverse through all the links on the web page.

Check below example: here we have used some keyword like URL, httpconnection etc.



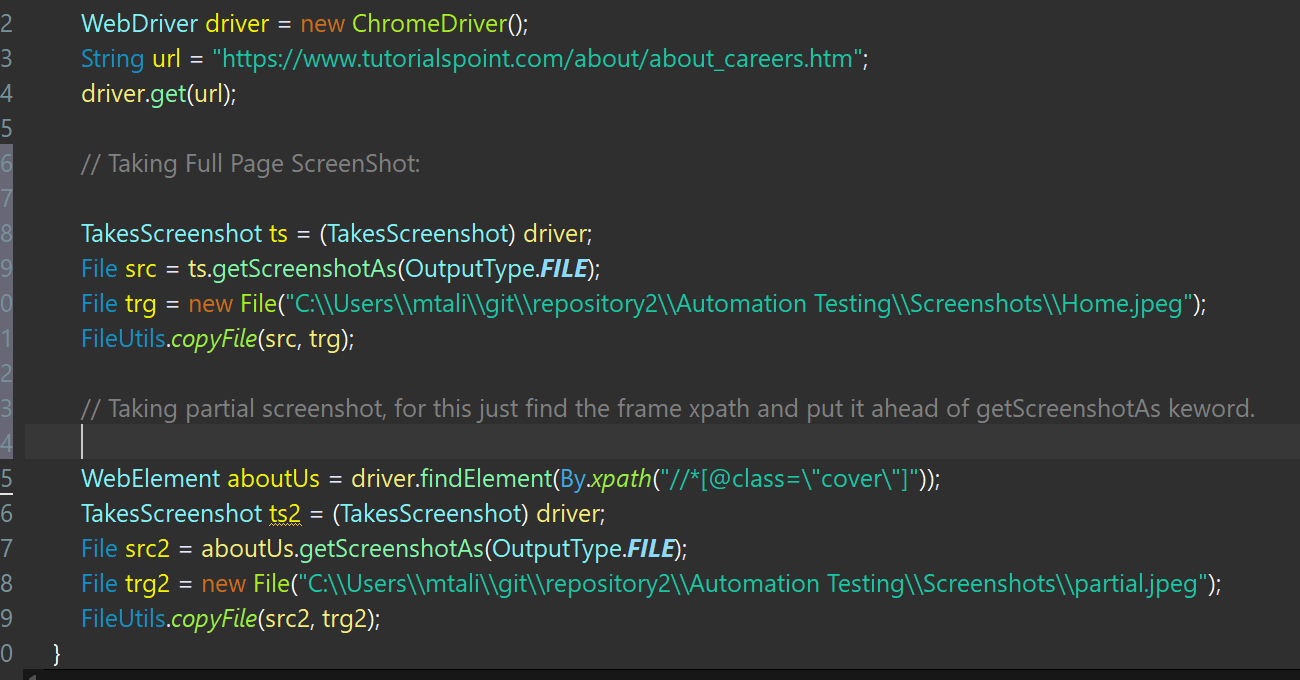
There are cases where an image on the web page does not load properly, and we see ***"Failed to load image"*** or similar error messages. In such cases, the image is either corrupt or the image is not at the specified path. A ***broken image*** on a web page is a link that is associated with the image, and the link is not working.

we will need to validate both the perspective of the image, i.e., the URL of the image should be valid, i.e., should return status code as 200, and the image should render correctly on the browser window, which we can validate using JavaScript. Marker 1 highlights a valid image in the above image, and Marker 2 highlights an invalid/broken image.



**Full and Partial Screenshot:**

Very Simple to find partial screenshot, just need to find the xpath of the frame:

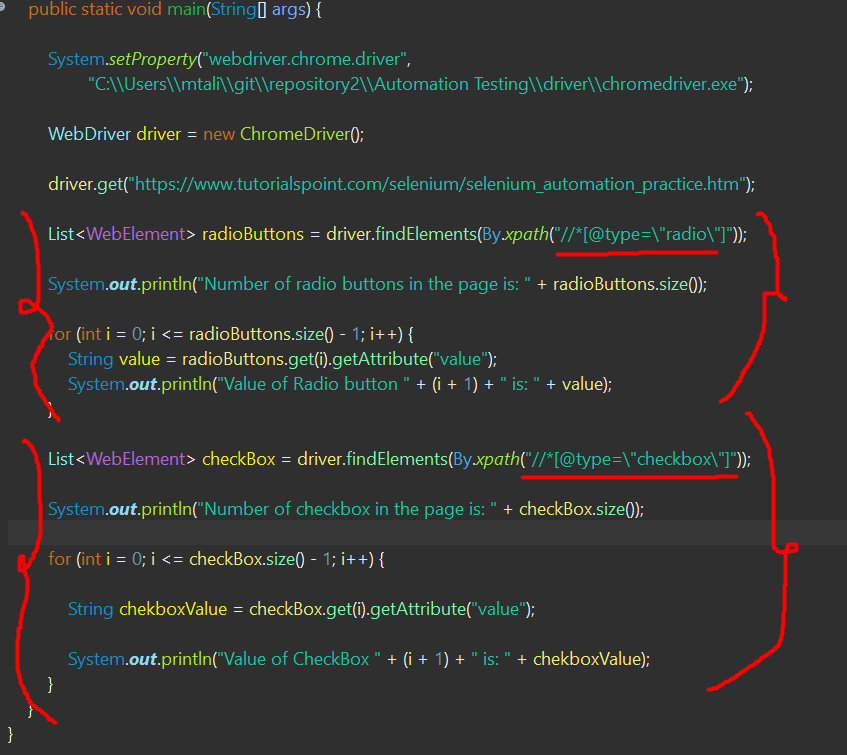


**Get the total number of Radio Button:**

**What is Radio and Checkbox Buttons**: A radio button is used when you want to select only one option out of several available options.

A checkbox allows you to choose one or many options to be selected from a list of options.

To find this is very simple, just find the xpath of all radio by **//\*[@type="radio"]** then put it in List<Webelement> and then we can easily find the number of radio buttons in the webpage.



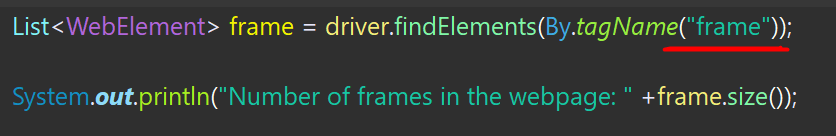
**How to find the all the links in a webpage:**

The total number of links in a page can be counted with the help of findElements() method. The logic is to return a list of web elements with tagname anchor, then getting the size of that list.

Just use below and can easily find the number of links in a webpage.

List<WebElement> links = driver.findElements(By.tagName("a"));

**To find the number of Frames in webpage use:**



Note: If List<Weblement> li = driver.findElements(By.xpath(“”));

Is not able to locate any element it **will return the Value of 0**

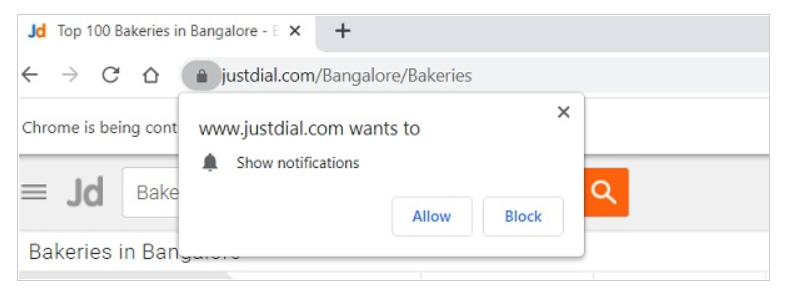
**To find the number of Headers in a webpage:**

The total number of headers in the web table can be counted with the help of findElements() method. The logic is to return a list of web elements with xpath with the help of <th> tag inside the table, then getting the size of that list.



**How to handle chrome notification in Selenium?**

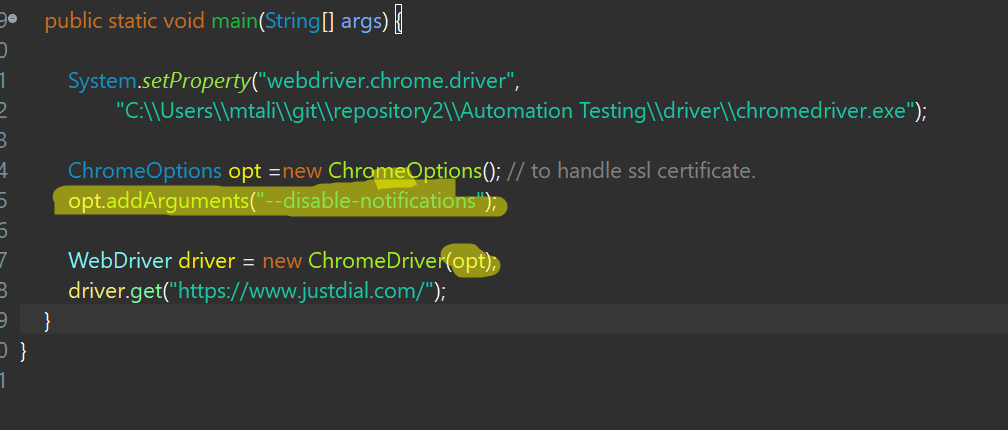
Look at below push notification, to handle these kind of notification we will use chromeoption class.



ChromeOptions o = new ChromeOptions();

o.addArguments("--disable-notifications");

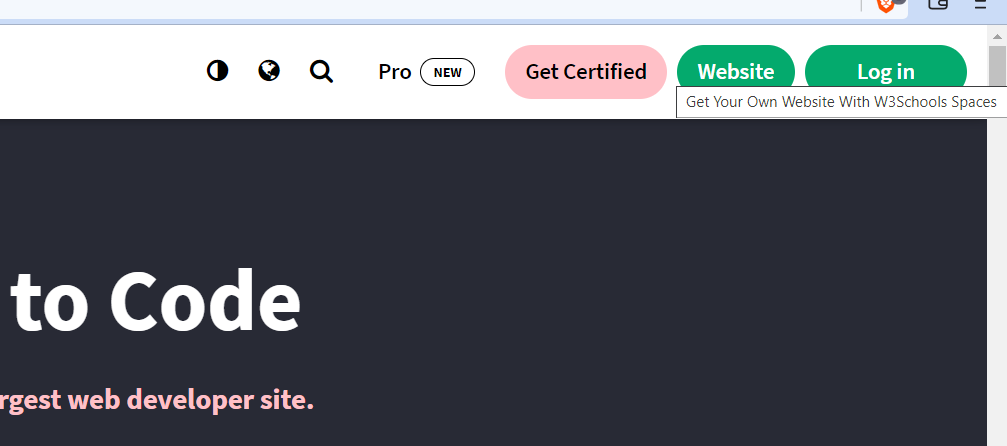
look at below code, to disable push notification:



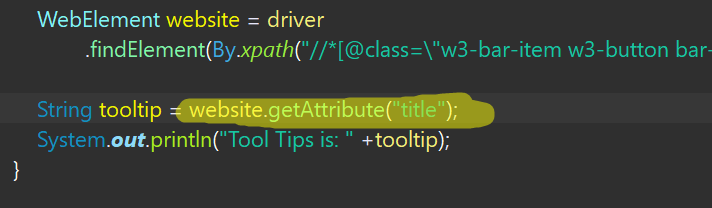
**ToolTips:**

A tooltip text becomes visible from an element as we hover the mouse on it. To obtain the tooltip we have to pass the title as a parameter to the getAttribute method.

Look at below while hovering over mouse on Website page:



To obtain the tooltip we have to pass the title as a parameter to the getAttribute method.



Sometimes, DOM may not have title attributes then we will use Actions class to get the ToolTip.

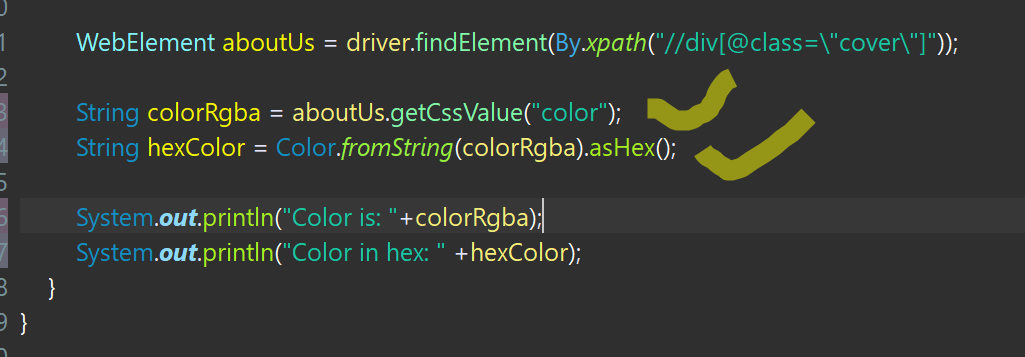
**The xpath of the element keeps changing, how do I find dynamic xpath for this element in Selenium?**

**How to verify colour of a web element in Selenium Web driver?**

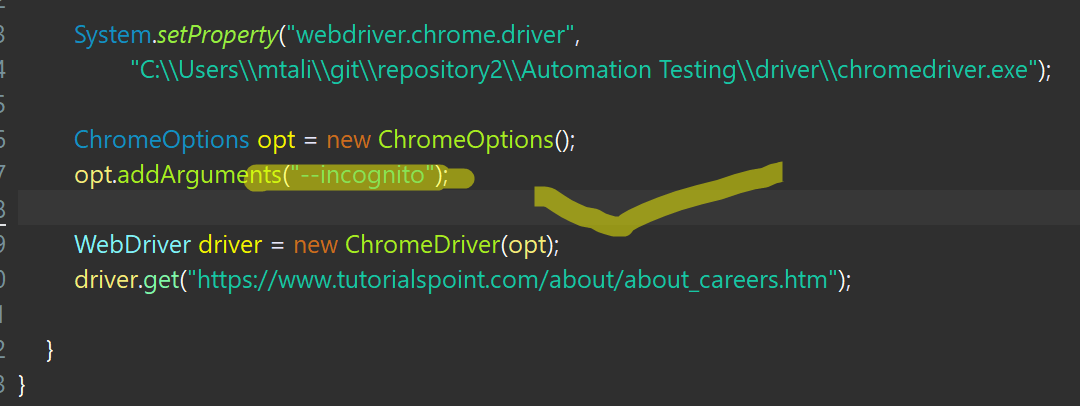
We can verify the color of a WebElement in Selenium web driver using the getCssValue method and then pass color as a parameter to it. This returns the color in rgba() format.

Next, we have to use the class Color to convert the rgba() format to Hexl

Look at below screenshot:



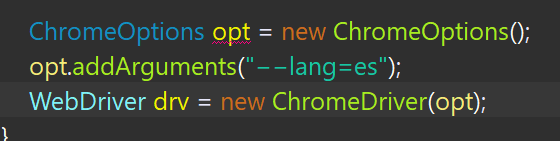
**Running chrome browser in inconginto Mode in Selenium**



**Setting up chrome language:**

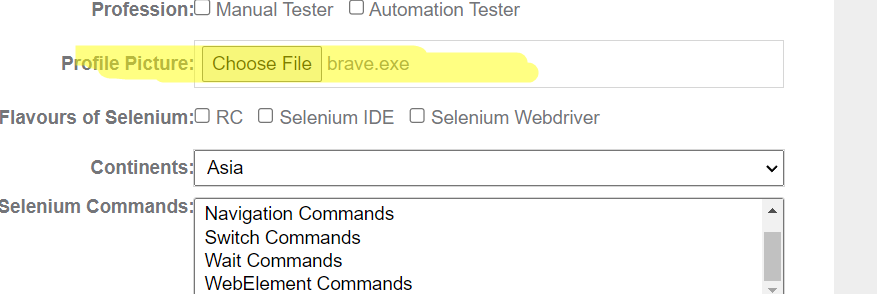
In Selenium, we can modify the language preferences with the help of **ChromeOptions** class for the Chrome browser. We shall create an object of this class and apply **addArguments** method on it.

To modify the language to Spanish, we have to pass **−−lang=es** as a parameter to the addArguments method. This information is then made available to the webdriver object.

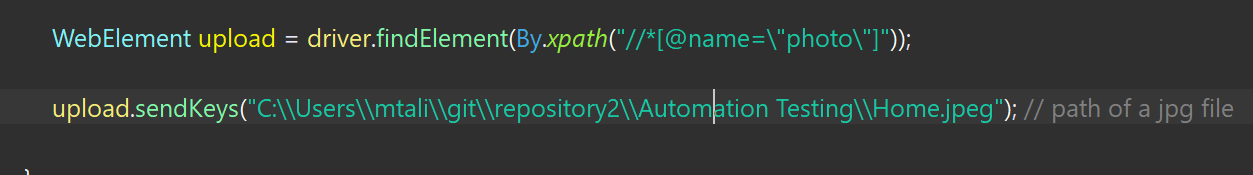


**Uploading file with no text box:**

Look at:

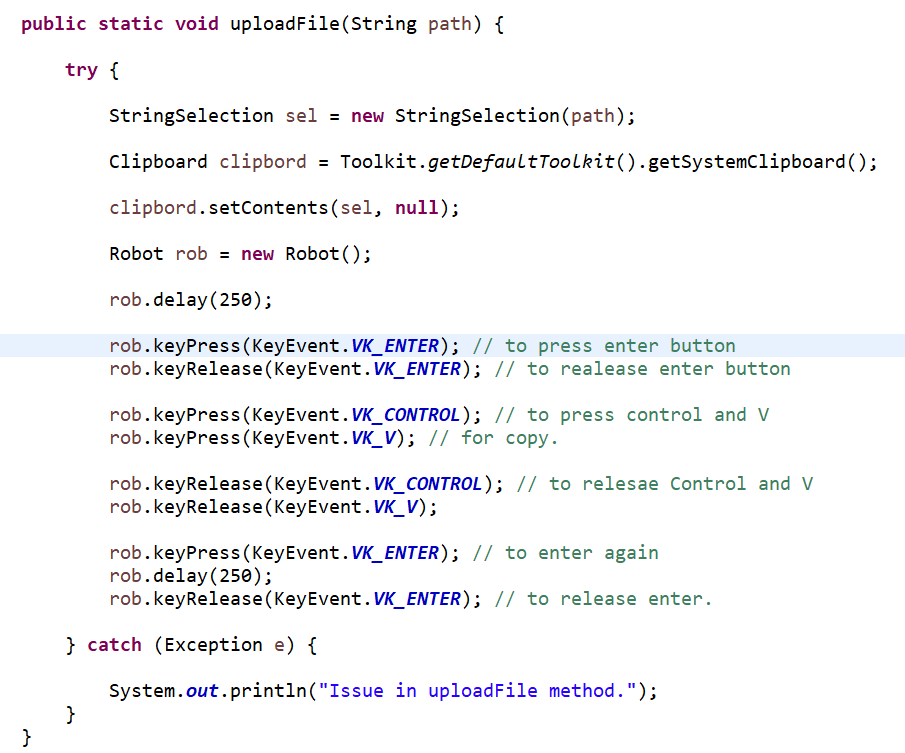


If we want to upload a file here we can use below code here:



Uploading the File:

We have two methods to upload a file. Robot method and AutoIT method. But we prefer Robot Method. As AutoIT method has some complication due to exe files.



implicit wait is simple to use and and applicable to all elements in the program.

Hence we can use implicit wait only once in the program.

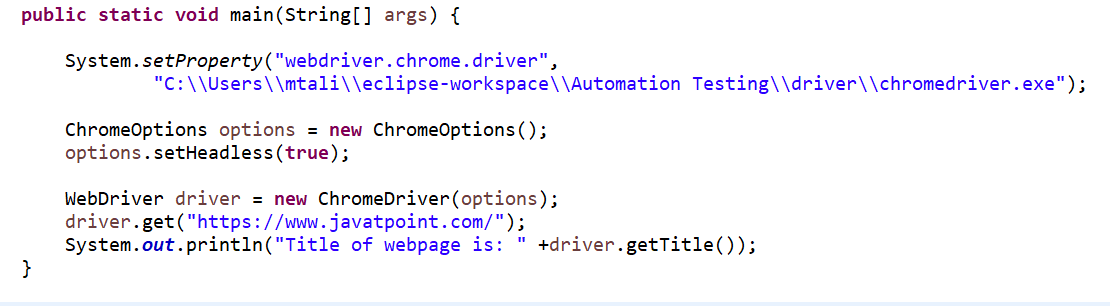
Explicit wait is elements specific, in explicit wait we can wait for any particular elements.

**Headless Browser:**

A headless browser is just like any other browser, the only difference is we cannot see anything on the screen. Here we can say that the program actually runs in the backend and nothing can be viewed on the screen. Thus, it is known to be the one without a Head/GUI.

To achieve this, we will use syntax;

Below test will be executed without opening a browser.



**Actions Class in Selenium**

***Different Methods for performing Mouse Events:***

* ***click():****Clicks at the current mouse location.*
* ***doubleClick():****Performs a double-click at the current mouse location.*
* ***contextClick() :****Performs a context-click at middle of the given element.*
* ***clickAndHold():****Clicks (without releasing) in the middle of the given element.*
* ***dragAndDrop(source, target):****Click-and-hold at the location of the source element, moves to the location of the target element*
* ***dragAndDropBy(source, xOffset, yOffset):****Click-and-hold at the location of the source element, moves by a given offset*
* ***moveByOffset(x-offset, y-offset):****Moves the mouse from its current position (or 0,0) by the given offset*
* ***moveToElement(toElement):****Moves the mouse to the middle of the element*
* ***release():****Releases the depressed left mouse button at the current mouse location*

**Difference between Action and Actions in Selenium**

**Actions** class is based on builder design pattern which builds a composite action with the aggregation of Selenium WebDriver, where webdriver is only used to identify the presence of web elements on web application

 **Action interface** is only used to represent the single user interaction i.e to perform the series of action items build by Actions class.

**Stale Elements Exception:**

A stale element reference exception is thrown in one of two cases, the first being more common than the second:

* The element has been deleted entirely.
* The element is no longer attached to the DOM.

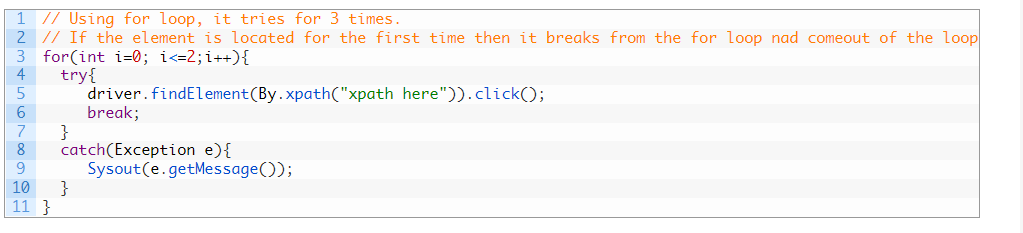
One way to generate stale elements exception is, suppose we are performing some action on a webpage using selenium web driver and trying to locate a specific WebElement. However as soon as page load we will click on some links then new page will be opened, in this case Selenium will not be able to locate the elements that was on previous page and will give stale elements exception.

How to fix stale element exception:

By refreshing the page:



Using Try Catch Block:



Using expectedconditons.refreshed:



Wait in Selenium

**Selenium – Synchronization**

Now a day, we see most of the web applications are developed by using Javacript and Ajax where it might happen that some of the elements may load at distant time intervals, due to which we may encounter “ElementNotVisibleException” or “NoSuchElementException” exceptions. In such scenarios, we use Synchronization/wait to overcome these kinds of exceptions.

Synchronization can be classified into two categories:

1. **Unconditional Synchronization**
2. **Conditional Synchronization**

**Unconditional** synchronization indicates timeout value alone and makes tool to continue further only when specified time is elapsed. One such functions is:

*System.Threading.Thread.Sleep();*

This statement is helpful only when we are interacting with third party systems like interfaces because it is not possible to write conditions to check the status. Now, we want tool to wait for certain amount of time and then perform the next actions sequence.

This statement makes the tool to wait for certain time even though the application is ready, thus increasing the chance of unnecessary wait and hence giving the scope to **Conditional Synchronization.**

**Conditional** Synchronization: By using conditional synchronization we can specify timeout duration along with some desired conditions to check and then throw an error if nothing happens.

**Types of Conditional Synchronization:**

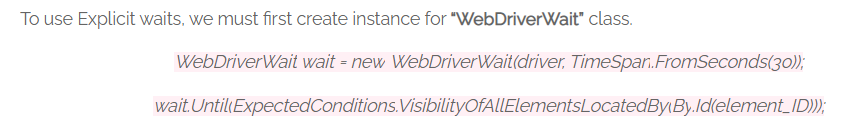
1. **Implicit Wait**
2. **Explicit Wait**

**Implicit Wait**: Implicit Wait tells the WebDriver to Wait until the stated time before throwing the NoSuchElement/ElementNotVisible exception. Waiting time across the test script between each consecutive steps are taken by default. Hence, next testStep will execute only when the specified time is elapsed post executing the previous testStep.

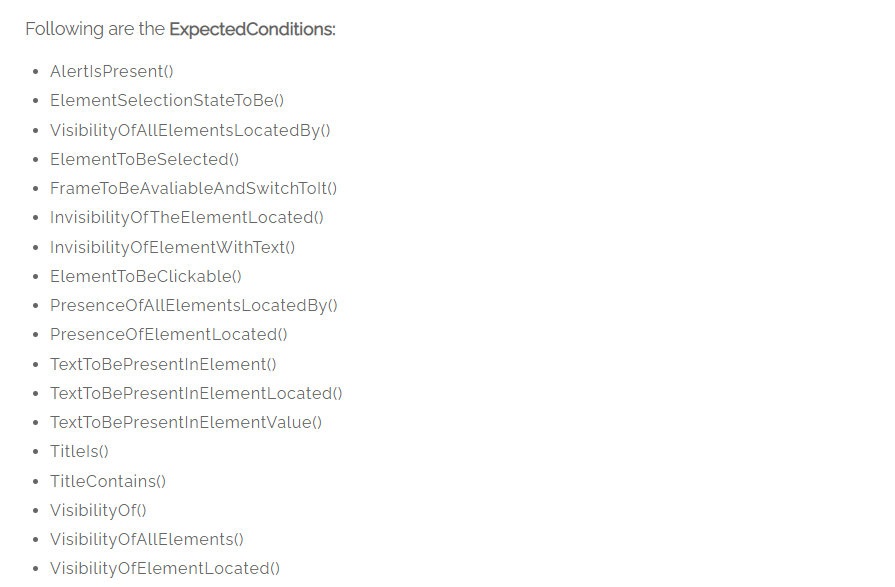


**Eg:** The wait time specified in Implicit Wait command is say 30seconds. The specified time (30seconds) is default for all the succeeding test steps thus giving rise to execution time. To overcome this issue, Explicit waits are introduced to apply waits whenever it is required while executing the test steps.

**Explicit Wait:**Explicit waits are very good to use when page loads dynamically. Explicit Wait tells the WebDriver to Wait until the specified condition is met or maximum time elapses before throwing NoSuchElement (or) ElementNotVisible Exceptions. Explicit waits are applied for the specified testStep in test script.



The reference variable “wait” informs the WebDriver to wait until the Expected condition to occur (or) Wait for the specified time of 30seconds, whichever shows in first place before throwing an exception.



The **Implicit wait** is simple when compared to Explicit wait and can be declared in setup method of the test script in single line of code. However, the Implicit wait has drawbacks despite being easy and simple to apply. Hence rising scope to use **Explicit Waits** because these waits can be applied for a specified testStep in test script.

Wait in selenium we have 3 types of wait.

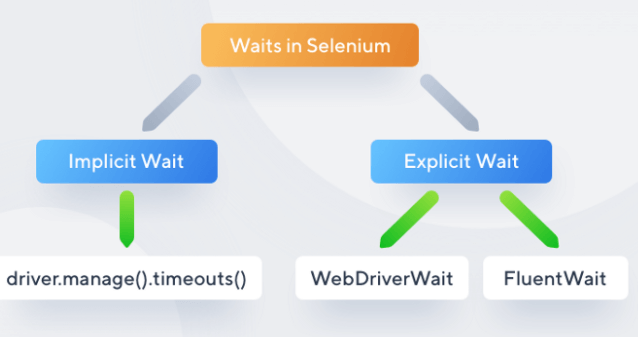
 Implicit wait

 Explicit wait

 Fluent wait

1. **Implicit wait:** Implicit wait we used to wait for the throw out the page. Implicit wait throws the only 1 types of exception like: Time out exception.

2. **Explicit wait**: Explicit wait to use for the different-different conditions, like: element to be visible, element to be clickable, visibility of element, alert is present like that.



Syntax for Implicit wait:

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

The main function of implicit Wait is to tell the web driver to wait for some time before throwing a **"No Such Element Exception".** Its default setting is knocked at zero. Once the time is set, the driver automatically will wait for the amount of time defined by you before throwing the above-given exception.

Advantage of implicit wait is it is simple to use and we can add implicit wait only once which is applicable for all the elements.

The main disadvantage of implicit wait is that it slows down test performance.

Another disadvantage of implicit wait is:

Suppose, you set the waiting limit to be 10 seconds, and the elements appear in the DOM in 11 seconds, your tests will be failed because you told it to wait a maximum of 10 seconds.

implicit wait is simple to use and and applicable to all elements in the program.

Hencce we can use implicit wait only once in the program.

Explicit wait is elements specific, in explicit wait we can wait for any particular elements.

**Please Check Below example both explict and implicit wait are shown:**



**Fluent Wait:**

The **Fluent Wait in Selenium** is used to define maximum time for the web driver to wait for a condition, as well as the frequency with which we want to check the condition before throwing an “ElementNotVisibleException” exception. It checks for the web element at regular intervals until the object is found or timeout happens.

**Frequency:**Setting up a repeat cycle with the time frame to verify/check the condition at the regular interval of time

Let’s consider a scenario where an element is loaded at different intervals of time. The element might load within 10 seconds, 20 seconds or even more then that if we declare an explicit wait of 20 seconds. It will wait till the specified time before throwing an exception. In such scenarios, the fluent wait is the ideal wait to use as this will try to find the element at different frequency until it finds it or the final timer runs out.

**What is DOM:**

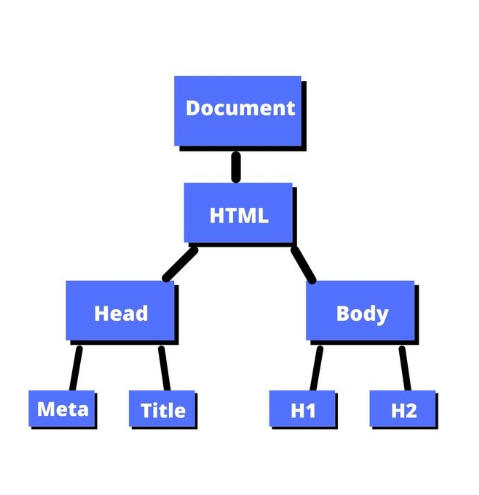
DOM stands for Document Object Model. It is a programming interface that allow us to create, change or remove elements from the document.

Look at below HTML code:



Our document is called the root node and contains one child node which is the <html> element. The <html> element contains two children which are the <head> and <body> elements.

Both the <head> and <body> elements have children of their own.



**Assertion in Selenium WebDriver using TestNG**

Assertion in TestNG framework is used to validate actual and expected result. By assertion if actual and expected will not match then test case will fail and same reason can be obtained from html report.

Actual result is compared with expected result with the help of Assertion. It means verification is done to check if the state of the application is the same to what we are expecting or not. For creating assertion, we are going to use the Assert class provided by TestNG.

Assertion we have 2 types-

* Soft assert/verify
* Hard assert

**Soft assert:**

In case of soft assert, if actual and expected data will not match then test case will not immediately fail, **till assertAll**() statement is not executed.

Soft asserts are just the opposite of hard asserts. In soft asserts, the subsequent assertions keep on running even though one assert validation fails, i.e., the test execution does not stop. ***Soft assert does not include by default in TestNG***. For this, you need to include the package ***org.testng.asserts.Softassert***. So, when should we use soft asserts in TestNG? We use soft asserts when we do not care about the failure of specific validations and want the test execution to proceed and also want to see the exception errors.

***Soft asserts are also known as*** "***Verify***" and hence do not get confused about the same.

**Hard assert:**

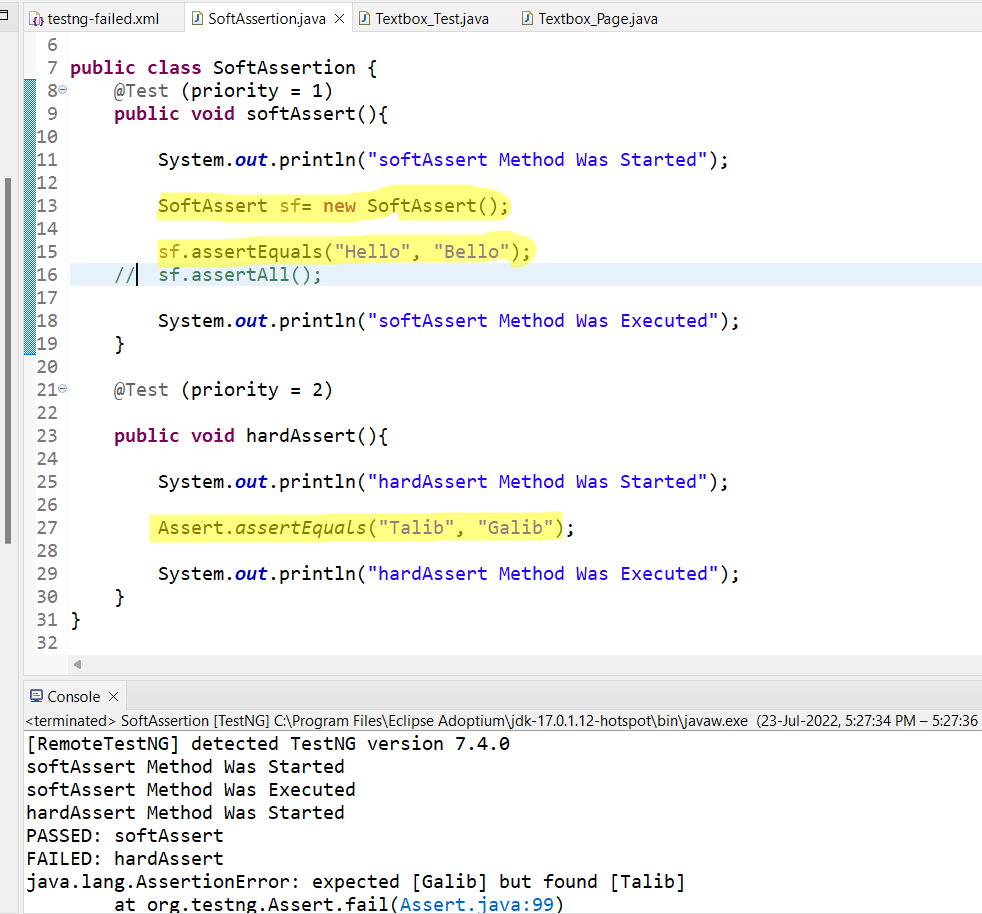
if actual and expected will not match then your test case will immediately fail. Suppose we have two test cases in a suite. The first test case in a suite has an assertion that fails, and if we want to run the second case in a suit, then we need to handle the assertion error.

It plays a vital role in projects where we have an element without whose validation, asserting other elements is useless. One good example in such cases is the login functionality. If I want to see my past orders, for example, then what is the point of checking this test case when the login validation already failed? ***Hard asserts are the default type of asserts in TestNG***, and what we used in the previous section was Hard Assert.

 A Hard Assertion contains the following methods:

* AssertEquals
* AssertNotEquals
* AssertTrue
* AssertFalse
* AssertNull
* AssertNotNull

Look at below example to understand it clearly:



1. We need a reference variable for to use soft assert.
2. If we don’t use assert.All() with soft assert, it will pass the failed test as in above example.
3. Hard assert can be used directly without reference variable.

**Difference between Hard and Soft assert:**

| **Hard Assert** | **Soft Assert** |
| --- | --- |
| Throws an exception immediately after the assert fails and carries out with the next test case of the suite. | Does not throw an exception immediately when the assertion fails, collects them and carries out with the next validation. |
| This throws an Assert Exception instantly so handled with a catch block. After suite execution is completed the test is made as PASS. | This accumulates the errors in each @Test execution. |

**Difference between Assert and Verify:**

| **l.no.** | **Assert** | **Verify** |
| --- | --- | --- |
| 1 | Verifies if the specified condition is true and false. If the result is true, the next test step will be executed. In case of false condition, the execution would terminate. | Verifies if the specified condition is true and false. If the result is true, the next test step will be executed. In case of false condition, the execution would still continue. |
| 2 | In case of false condition, the next test case of the suite will be executed. | In case of false condition, the next test step of the same test case will continue. |
| 3 | There are two types of assets namely hard and soft asserts. | There are no categories for verification. |

***How do I implement verify in TestNG?***  
*Verify is another name for soft asserts. Soft asserts can implement in the test code with the help of SoftAssert class in TestNG*.

***What does "assert fail" mean in TestNG?***  
*Assert fail refers to the failure of the assertion test method. The conditions for failing depends totally on the assertion methods. When an assertion fails, they throw an exception error onto the console describing the failed test (only in hard asserts)*.

***Can we apply the assert statements inside the if-statements?***  
*Yes, assert statements are just like any other methods that return a Boolean true or false value. A tester can leverage the return value and can use assert statements at any place in the code*.

.

JavaScriptExecutor in Selenium WebDriver

JavaScriptExecutor is an Interface that helps to execute JavaScript through Selenium WebDriver. JavaScriptExecutor provides two methods “executescript” & “executeAsyncScript” to run JavaScript on the selected window or current page.

In case, the locators do not work you can use JavaScriptExecutor. You can use JavaScriptExecutor to perform a desired operation on a web element.

**JavaScriptExecutor Methods**

1. **executeAsyncScript**

With Asynchronous script, your page renders more quickly. Instead of forcing users to wait for a script to download before the page renders. This function will execute an asynchronous piece of JavaScript in the context of the currently selected frame or window in Selenium. The JS so executed is single-threaded with a various callback function which runs synchronously.

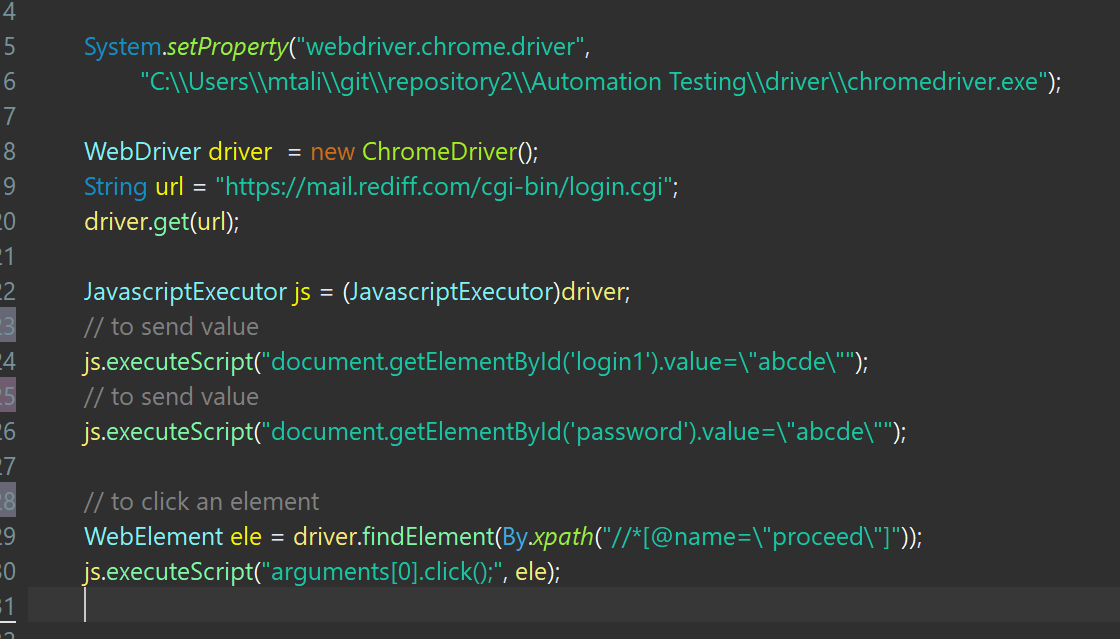
1. **executeScript**

This method executes JavaScript in the context of the currently selected frame or window in Selenium. The script used in this method runs in the body of an anonymous function (a function without a name). We can also pass complicated arguments to it.

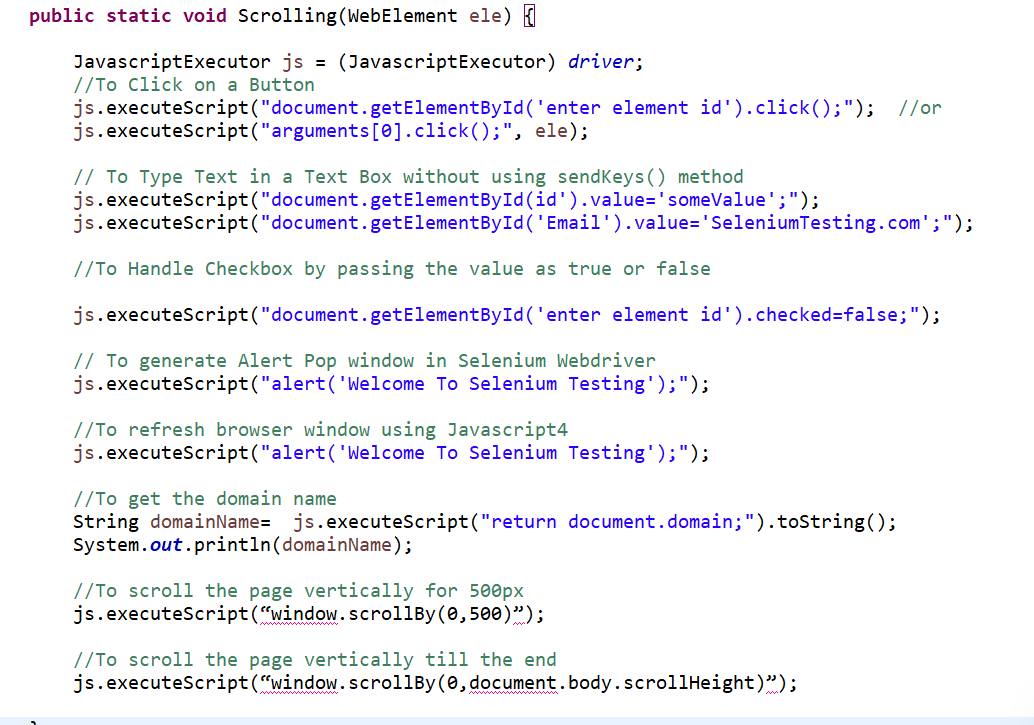
The script can return values. Data types returned are

* Boolean
* Long
* String
* List
* WebElement.

Look at below screenshot:



Some Important example:



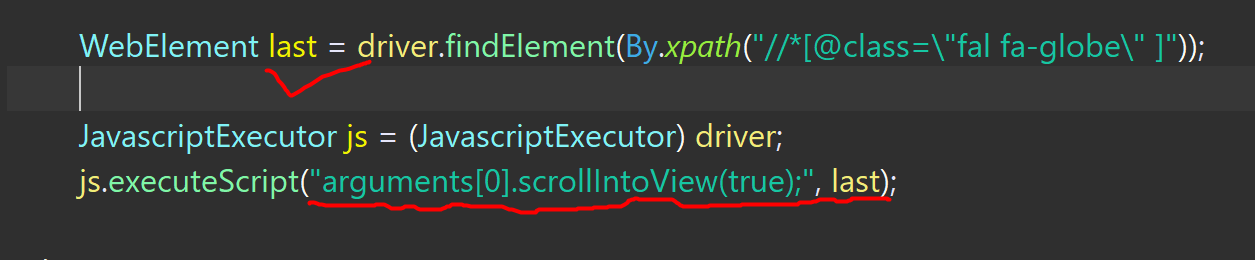
Syntax:

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript(scripts, args);

**Scrolling to element using JavaScript:**

It is very simple first we will find the WebElement where we want to scroll and will use JavaScript executor then, look at below:



**How to handle hidden elements in selenium:**

Hidden elements are the elements which are not displayed on the Web-Pages, however, it is available in the HTML DOM structure.

Elements are hidden on the web-pages due to below-mentioned reasons -

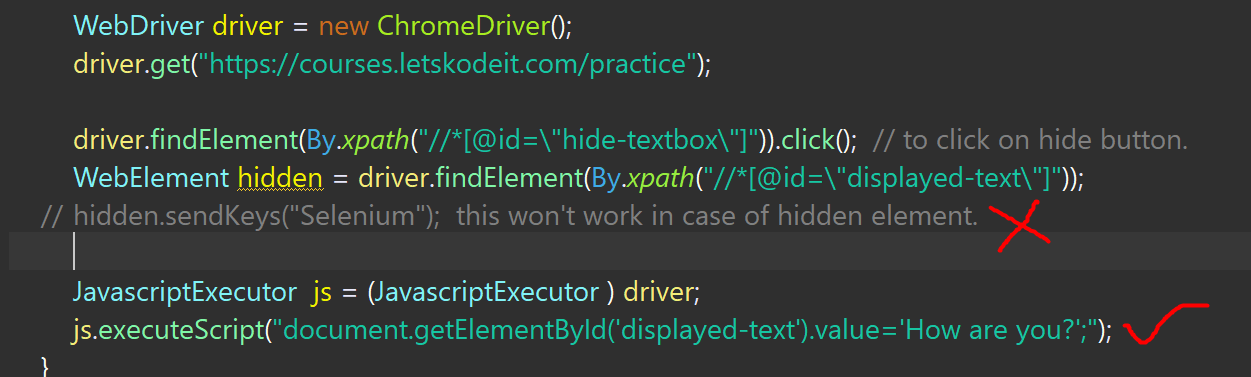
* They have a CSS display value of none.  style=”display: none;”
* They are form elements with type="hidden"
* Their width and height are explicitly set to 0.
* An ancestor element is hidden, so the element is not shown on the page.

Issue - Selenium does not allow to interact with an element which is hidden on the web-page and if we try to do so, then it will throw**''Element Not Visible Exception".**

On url: <https://courses.letskodeit.com/practice> we have option for hidden and show element. If hidden element button is disabled and then we can simply pass the value in text box, However hidden button is enabled we cant pass the value in text box using simple method as it will give an error.

Syntax; **JavascriptExecutor js = (JavascriptExecutor ) driver;**

**js.executeScript("document.getElementById('displayed-text').value='How are you?';");**



**Explanation** -

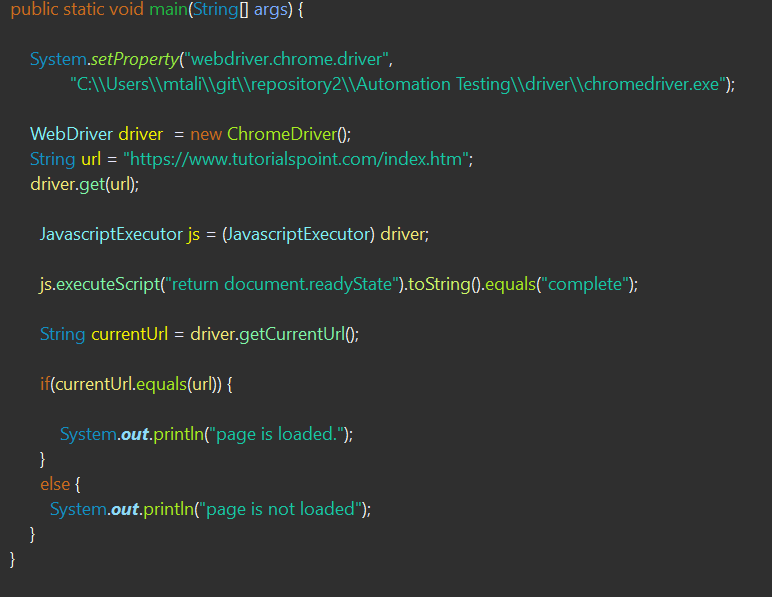
* **JavascriptExecutor**- It is an interface provided by Selenium WebDriver to execute Javascript commands in the Selenium Script.
* **getElementById() -**It is a Javascript method that is used to locate an element on the web-page using **ID**attribute of that element.
* **executeScript()** - It is a method of **JavascriptExecutor**interface to execute the javascript commands in the Selenium Script.

**In a web page, how will you ensure that the page has been loaded completely?**

We can get Selenium to recognize that a page is loaded. We can set the **implicit wait** for this purpose. It shall make the driver to wait for a specific amount of time for an element to be available after page loaded.



**Check below example:**



**TestNG framework:**

**TestNG**is an automation testing framework in which NG stands for “Next Generation”. TestNG is inspired by JUnit which uses the annotations (@). TestNG overcomes the disadvantages of JUnit and is designed to make end-to-end testing easy.

Using TestNG, you can generate a proper report, and you can easily come to know how many test cases are passed, failed, and skipped. You can execute the failed test cases separately.

For example:

* Suppose, you have five test cases, one method is written for each test case (Assume that the program is written using the main method without using **TestNG**. When you run this program first, three methods are executed successfully, and the fourth method is failed. Then correct the errors present in the fourth method, now you want to run only fourth method because first three methods are anyway executed successfully. This is not possible without using TestNG.
* The TestNG in Selenium provides an option, i.e., testng-failed.xml file in test-output folder. If you want to run only failed test cases means you run this XML file. It will execute only failed test cases.

**Why to use TestNG:**

* Generate the report in a proper format including a number of test cases runs, the number of test cases passed, the number of test cases failed, and the number of test cases skipped.
* Multiple test cases can be grouped more easily by converting them into testng.xml file. In which you can make priorities which test case should be executed first.
* The same test case can be executed multiple times without loops just by using keyword called ‘invocation count.’
* Using **TestNG**, you can execute multiple test cases on multiple browsers, i.e., cross browser testing.
* The TestNG framework can be easily integrated with tools like TestNG Maven, Jenkins, etc.
* Annotations used in the testing are very easy to understand ex: @BeforeMethod, @AfterMethod, @BeforeTest, @AfterTest
* WebDriver has no native mechanism for generating reports. TestNG can generate the report in a readable format like the one shown below.
* TestNG simplifies the way the tests are coded. There is no more need for a static main method in our tests. The sequence of actions is regulated by easy-to-understand annotations that do not require methods to be static.
* Uncaught exceptions are automatically handled by TestNG without terminating the test prematurely. These exceptions are reported as failed steps in the report.
* Parallel testing is possible.

TestNG framework work with selenium web driver. TestNG have some pre-define classes, interfaces and methods. Using TestNG we can perform multiple actions like-Parallel test case execution, Cross browser testing, Group by test cases

**What is Annotation in TestNG:**

Annotations, in general, mean "***a comment***" or "***a note***" on a diagram, etc. to denote its meaning. TestNG also uses them for the same reason. TestNG annotations are the code that is written inside your source test code logic to control the flow of the execution of tests. It is essential to annotate your methods in TestNG to run the tests. [***TestNG***](https://testng.org/doc/) will ignore the method which does not contain an annotation since it won't know when to execute this method.

**Annotations in TestNG are lines of code that can control how the method below them will be executed**. They are always preceded by the @ symbol. A very early and quick TestNG Example is the one shown below.

Types of TestNG annotations:

***Types Of TestNG Annotations***

In TestNG, there are ten types of annotations:

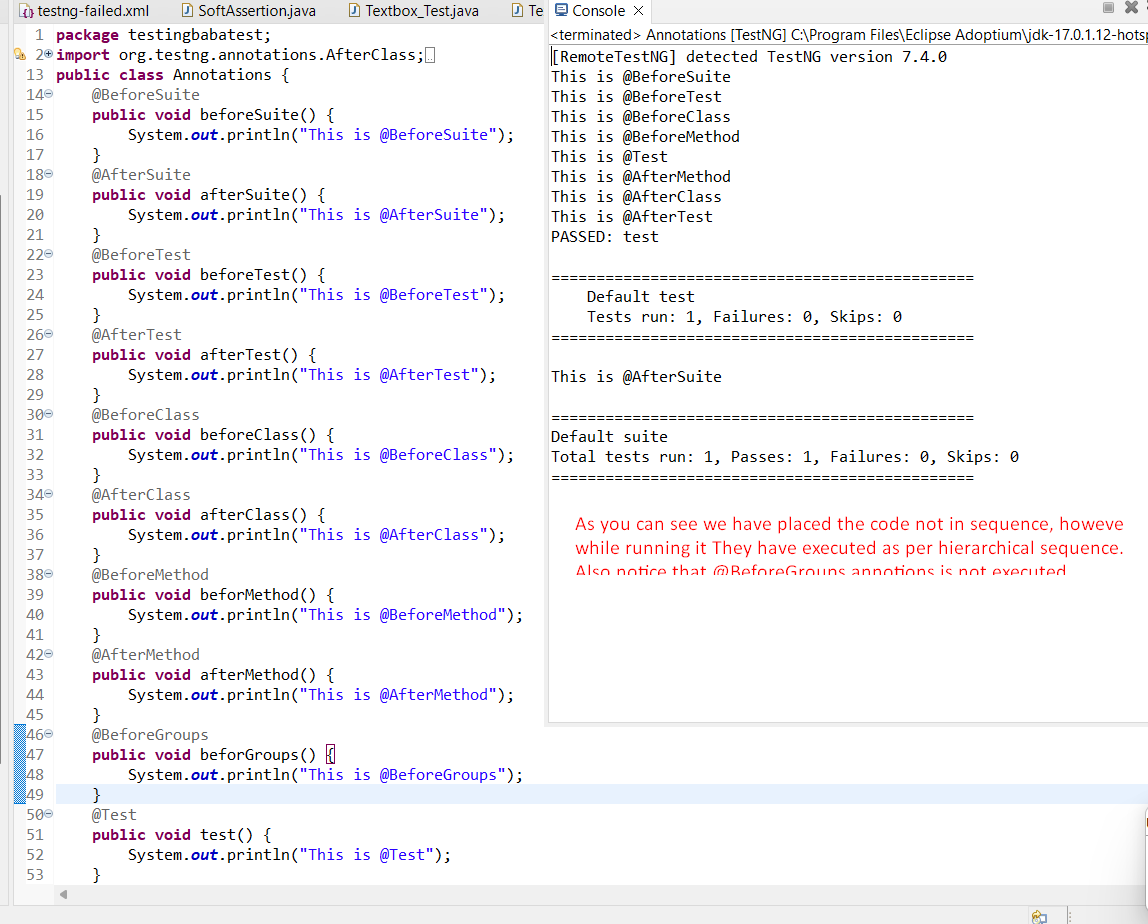
* ***@BeforeSuite*** - *The @BeforeSuite method in TestNG runs before the execution of all other test methods*.
* ***@AfterSuite*** - *The @AfterSuite method in TestNG runs after the execution of all other test methods*.
* ***@BeforeTest*** - *The @BeforeTest method in TestNG runs before the execution of all the test methods that are inside that folder*.
* ***@AfterTest*** - *The @AfterTest method in TestNG executes after the execution of all the test methods that are inside that folder*.
* ***@BeforeClass*** - *The @BeforeClass method in TestNG will run before the first method invokes of the current class*.
* ***@AfterClass*** - *The @AfterClass method in TestNG will execute after all the test methods of the current class execute*.
* ***@BeforeMethod*** - *The @BeforeMethod method in TestNG will execute before each test method*.
* ***@AfterMethod*** - *The @AfterMethod method in TestNG will run after each test method is executed*.
* ***@BeforeGroups*** - *The @BeforeGroups method in TestNG run before the test cases of that group execute. It executes just once*.
* ***@AfterGroups*** - *The @AfterGroups method in TestNG run after the test cases of that group execute. It executes only once*.

**Hierarchy of TestNG suite:**

|  |  |
| --- | --- |
| @BeforeSuite | The @BeforeSuite annotated method will run before the execution of all the test methods in the suite. |
| @BeforeTest | The @BeforeTest annotated method will be executed before the execution of all the test methods of available classes belonging to that folder. |
| @BeforeClass | The @BeforeClass annotated method will be executed before the first method of the current class is invoked |
| @BeforeMethod | The @BeforeMethod annotated method will be executed before each test method will run. |
| @Test |  |
| @AfterMethod | The @AfterMethod annotated method will run after the execution of each test method. |
| @AfterClass | The @AfterClass annotated method will be invoked after the execution of all the test methods of the current class. |
| @AfterTest | The @AfterTest annotated method will be executed after the execution of all the test methods of available classes belonging to that folder. |
| @AfterSuite | The @AfterSuite annotated method will run after the execution of all the test methods in the suite. |

Using these annotations we achieve code reusability, and can easily maintain automation script.

Check below example to understand hierarchy of annotations:



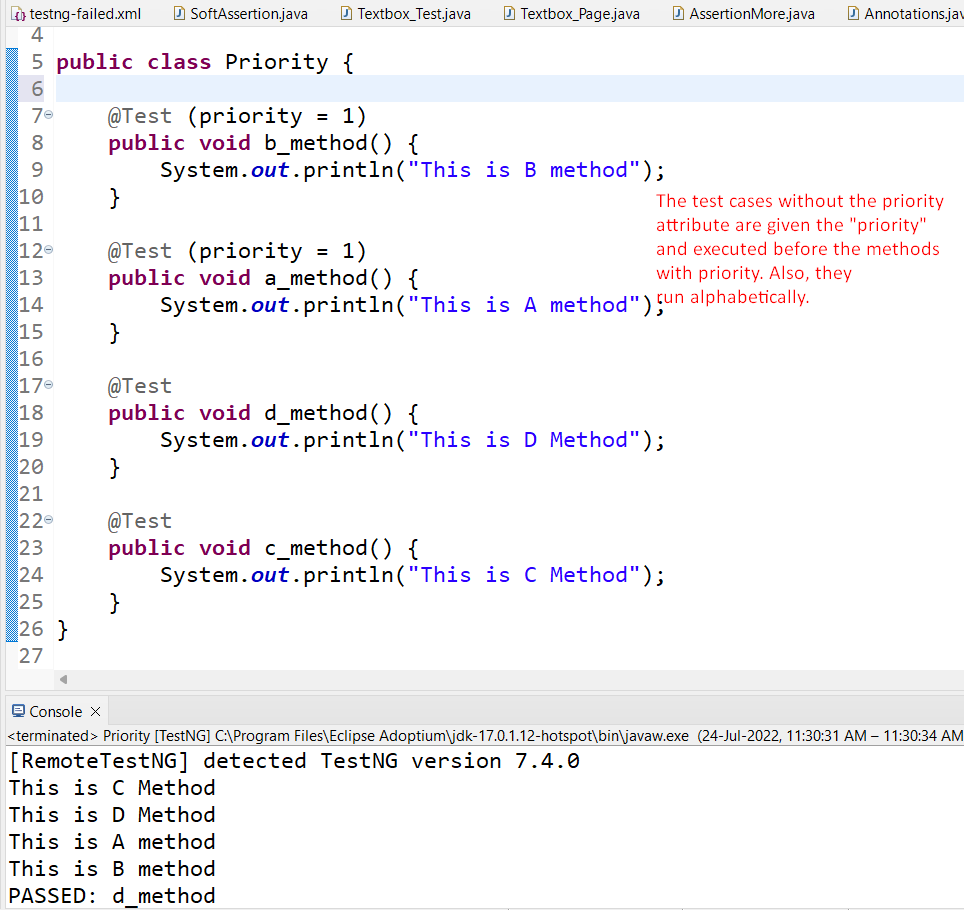
The ability to run tests in parallel is available in TestNG but not in JUnit, so the TestNG framework is more preferred for testers using Selenium Grid.

**@BeforeSuite:** The **@BeforeSuite** annotated method is executed before the execution of all the test cases defined in the folder.

Generally, @BeforeSuite is used when we have different URLs to run your test cases. Environment variables are set in a @BeforeSuite annotated method so that before executing all the test cases, you need to load all the environment variables for your framework, and then it starts executing your test cases.

The @BeforeSuite annotated method is given as the first priority, so it is executed before all the other test methods.

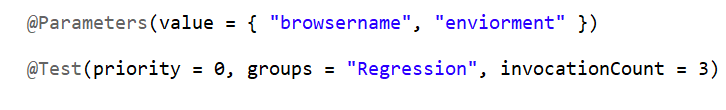
**Check below example to understand priority more clearly:**



**Parameters:**

If you want the methods to be executed in a different order, use the parameter “priority”. **Parameters are keywords that modify the annotation’s function**. TestNG will execute the @Test annotation with the lowest priority value up to the largest. There is no need for your priority values to be consecutive.

**Multiple Parameters:**



**What is TestNG Suite:**

Collection of [***TestNG Tests***](https://www.toolsqa.com/testng/testng-test/) together is called a ***Test Suite***. A test suite can run multiple tests at once by executing the test suite. Additionally, these test cases can be dependent on each other or may have to be executed in a specific order independently. Moreover, running the TestNG test suite gives us the capability to manage our test execution.

***It is important to remember that the TestNG does not let us define the test suites inside the test code or the main testing source code. Hence, we need to create a TestNG XML file for the same and execute this file.***

***To create TestNG XML:***

***Right click on TestNG project, Go to TestNG and click on Convert to TestNG.***

We can also implement with Page object model with Test-NG

**What is the structure of a Framework?**

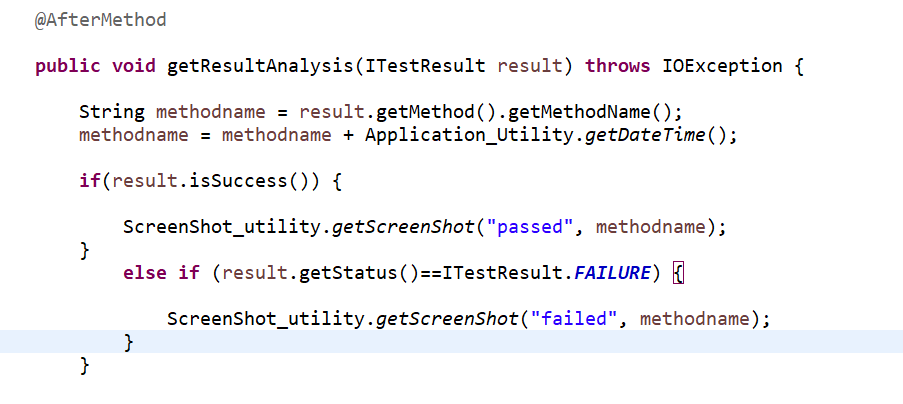
Different projects have different framework structures. But generally, a framework should have:

* A source(src) folder that has original test scripts
* A library(lib) folder that has the entire libraries and common methods
* A class folder that has the entire class file in case of java
* A file with URL, environment together with login information
* A log folder that has the entire log files
* A web element Ids folder

**TestNG Listener:**

TestNG provides the @Listeners annotation which listens to every event that occurs in a selenium code. Listeners are activated either before the test or after the test case. It is an interface that modifies the TestNG behaviour. For example, when you are running a test case either through selenium and suddenly a test case fails. We need a screenshot of the test case that has been failed, to achieve such scenario, TestNG provides a mechanism, i.e., Listeners. When the test case failure occurs, then it is redirected to the new block written for the screenshot.

Check below ss, where we have used Listeners in @AfterMethod to take the screenshot of failed and passed method.



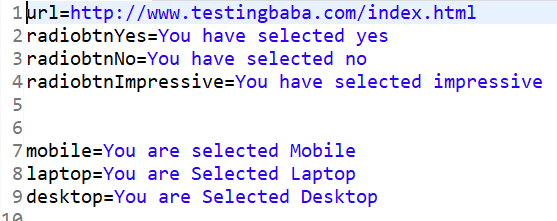
**config.properties file in selenium**

1. Properties are used to externalize the data which is configurable and if you put that data in your code (test script) you have to build the code each time you want to change the value of the property. The main advantage of properties is that they are outside your source code and you can change them anytime.
2. Each parameter is stored as a pair of strings, one storing the name of the parameter (called the key), and the other storing the value.
3. Whatever constant and generic entities are there in our project, we can define in config.properties file

**To create config.properties:**

1. Open eclipse. Right click on the project, select New→ file→ give file name as “config.properties”→ Finish. We cannot write java code in this. It is simple text file. Make sure to use extension as “properties” and not “property“.

**Key Value**

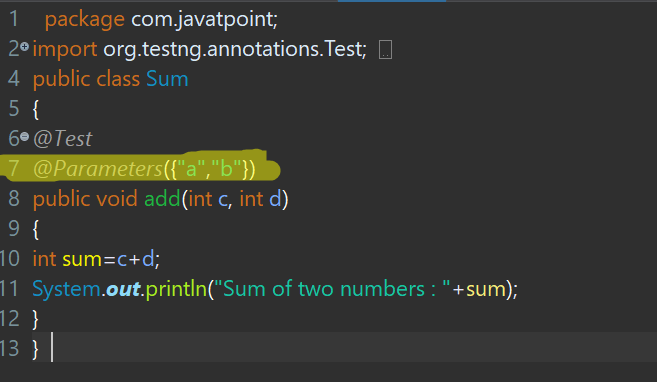


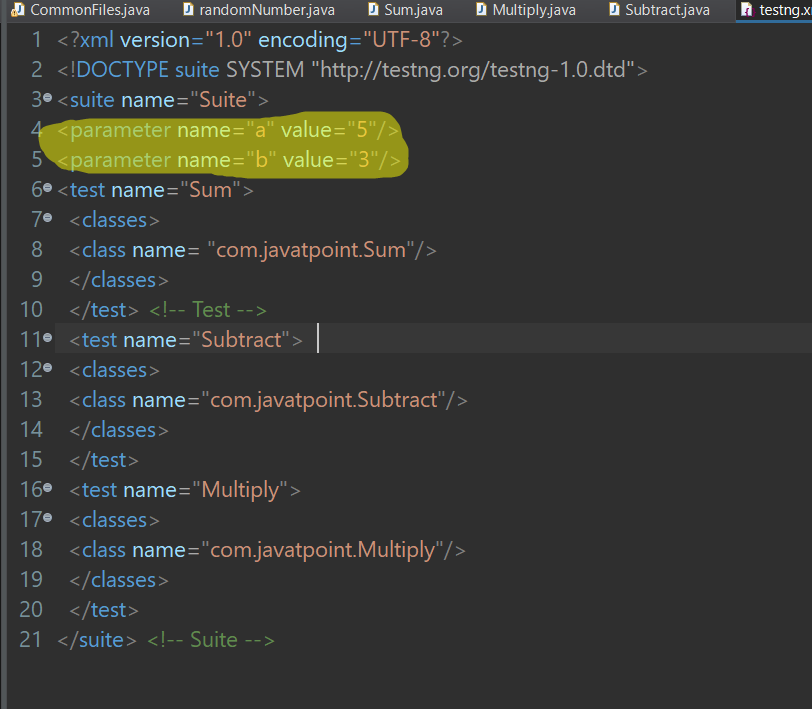
**TestNG Parameters:**

Suppose we want to set the global variables such url settings, username, password or API Keys, there are some values which are constant in all the test cases, in such case we use the TestNG Parameters.

TestNG Parameters are present in the xml file. They can be applied either inside the tag or tag. If we want to apply the parameters to all the test cases, then the parameters are applied inside the tag. If the parameter is specific to a particular folder, then the parameter is applied within a tag.

Look at below example, Parameters variable a and b are crated under @Test annotations and value of the parameters a and b is defined under Testng.xml file.





**TestNG DataProviders**:

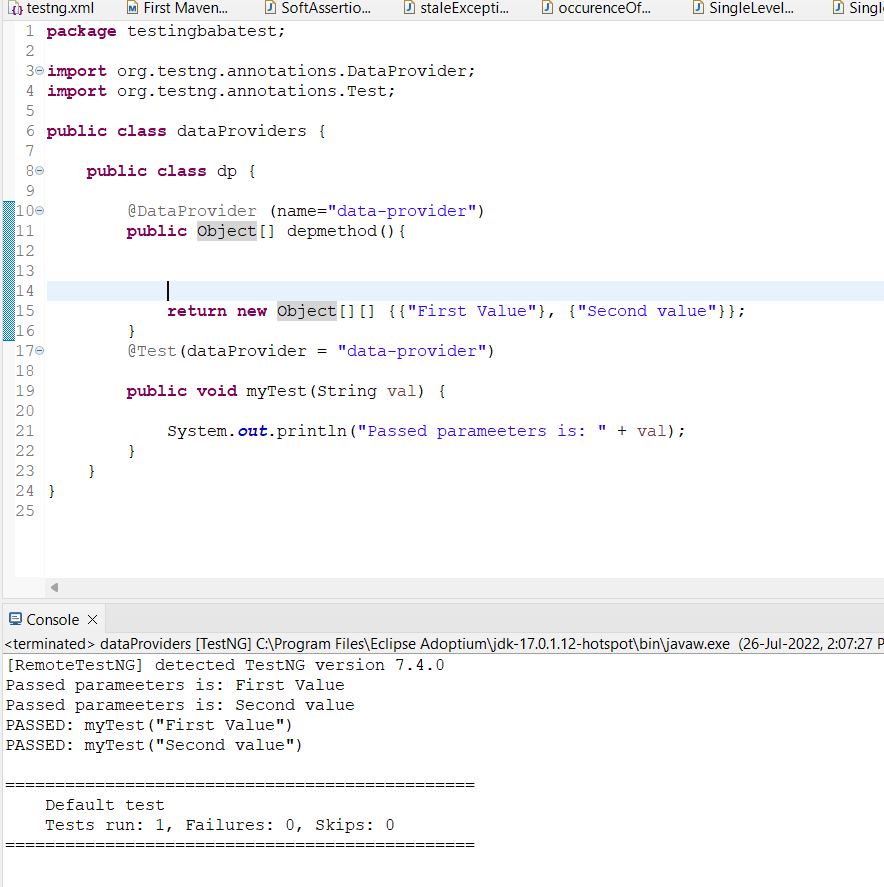
DataProviders annotation in TestNG, are another way to pass the parameters in test function() , the other one being TestNG parameters.

**Data Provider in TestNG** is a method used when a user needs to pass complex parameters. Complex Parameters need to be created from Java such as complex objects, objects from property files or from a database can be passed by the data provider method. The method is annotated by @DataProvider and it returns an array of objects.parameters in different ways.

Look at below syntax;



Look at below example:



Note below points:

TestNG DataProviders contains only one single attribute, which is its name, it is always a string type in nature. For example, “data-proivder”, as mentioned above.

DataProviders are not declared on top of function like TestNG parameters but have a method of their own, which in regular speaking terms called a DataProviders method. For example dpmethod here.

If tester has not specified the name of the DataProviders, then the method name becomes the DataProviders name by default.

TestNG DataProviders returns a 2d list of objects.

This method then performs a data-driven test for each value that you have specified.

The DataProvider name calls the DataProviders method is the default name used in the receiving @Test case.

***How do you give parameters in TestNG?***  
*There are two ways to pass the parameters in TestNG*:

* *TestNG Parameters*
* *TestNG DataProviders*

***What is the difference between DataProvider and Parameter in TestNG?***  
*DataProviders pass the different parameters on a single test in a single execution, whereas parameters pass the parameters just once per execution in*[***TestNG***](https://testng.org/doc/).

**Page Object Model and Page Factory in Selenium**

Page Object Model, also known as POM, is a design pattern in Selenium that creates an object repository for storing all web elements. It is useful in reducing code duplication and improves test case maintenance.

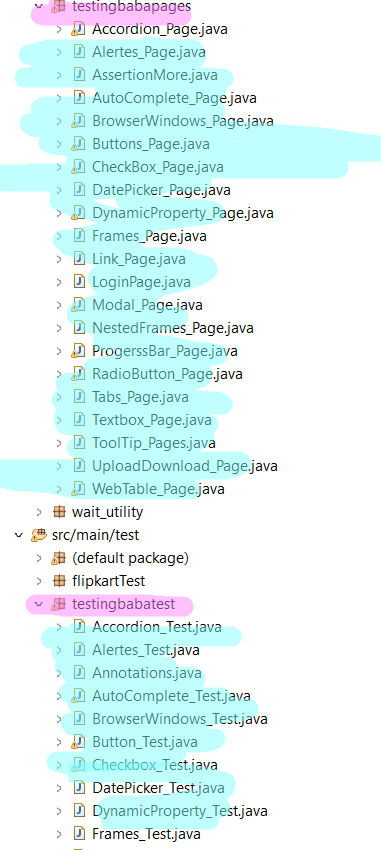
**Benefits of POM:**

* + Easy maintenance.
  + Help with reusing code.
  + Readability and reliability of scripts.

In Page Object Model, consider each web page of an application as a class file. Each class file will contain only corresponding web page elements. Using these elements, testers can perform operations on the website under test.

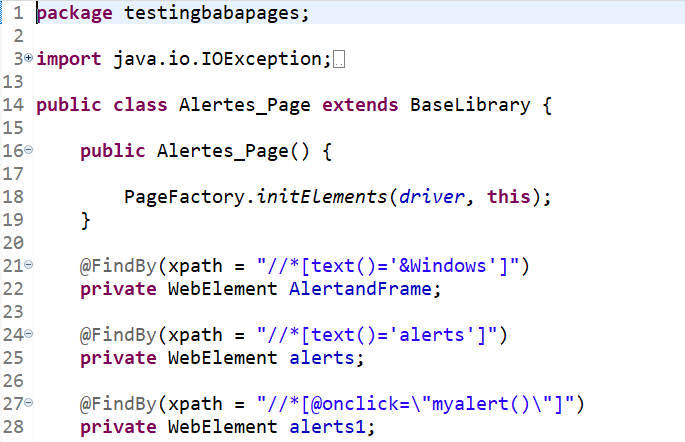
Under this model, for each web page in the application, there should be a corresponding Page Class. This Page class will identify the WebElements of that web page and also contains Page methods which perform operations on those WebElements. Name of these methods should be given as per the task they are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

As we have created a framework testing baba:



**Page Factory in Selenium?**

**Page Factory in Selenium** is an inbuilt Page Object Model framework concept for Selenium WebDriver but it is very optimized. It is used for initialization of Page objects or to instantiate the Page object itself. It is also used to initialize Page class elements without using “FindElement/s.”



**Is Page object model a framework?**

A) Page Object Model is a Design Pattern which has become popular in Selenium Test Automation. Page object model (POM) can be used in any kind of framework such as modular, data-driven, keyword driven, hybrid framework etc.

**Explain the Automation Test Framework in your project:**

If asked in interview, we can explain our project as mentioned below:

**Language:** In our project framework we are using Java language because most of automation tester have knowledge of Selenium with Java.

**Type of Framework**: In our project we are using **Data driven Framework** by using Page Object Model design pattern with Page Factory.

**POM**: As per the POM, we have maintained a class for every web page. Each web page has a separate class and hold the functionality and member of the webpage. With separate class we have corresponding test classes in page object model.

**Packages**: We have separate packages for **Pages** and **Tests**. All the web page related classes come under the **Pages** and all the tests related class come under the **Tests** packages.

For example, Homepage and Login Page have separate class to store element locators. For the login test, there would be a separate class which calls the methods from the Home Page class and Login Page class.

**Base**: Class: In our project Base class have all the common functions used by all the pages. This class is responsible for initializing the Web Driver, Explicitly Waits, Maximizing the browser.

**Utility Class**: In utility class we will store the functions which are repetitive in nature, such as waits, mouse actions, accessing excels files, capturing the screenshot.

So any functionality that can be used across the entire frameworks can be stored in Utility class.

**Properties File**: Properties file (config. properties) is stored as a pair of strings, in key value pair. With the help of properties file, we don’t need to use hard coded value in our framework. If any information is changed from the properties file, we can change that value without affecting the class.

For example, we keep application URL in the property file, so in case you want to run test from on other test environment, just change the URL in a property file and that's it. You do not require to build the whole project again.

**Screenshots:** Screenshots will be captured and stored in a separate folder and also the screenshots for failed test cases will be stored. To store the screenshots of failed test cases we have used ITEST listener under @AfterMethod annotations in base library.

**Test Data:** All the historical test data will be kept in excel sheet and we can access this data by a separate call using Apache POI.

**TestNG**: We are using TestNG for grouping, Assertions, Reports, Parallel executions etc.

**Maven**: We are using Maven for build, execution, and dependency purpose, Integrating the dependency in POM.xml file and running this POM.xml using Jenkins.

**Version Control Tool**: We use git repository to store our test scripts.

**Jenkins**: By using Jenkins CI tool, we execute test cases on a daily basis and also for nightly execution.

**Top Selenium Exception**

Though there are many Exception classes under WebDriverException, we commonly see the below ones.

* *NoSuchElementException*
* *NoSuchWindowException*
* *NoSuchFrameException*
* *NoAlertPresentException*
* *InvalidSelectorException*
* *ElementNotVisibleException*
* *ElementNotSelectableException*
* *TimeoutException*
* *NoSuchSessionException*
* *StaleElementReferenceException*

**NoSuchElementException:** The exception occurs when WebDriver is unable to find and locate elements. Usually, this happens when tester writes incorrect element locator in the findElement(By, by) method.

**NoSuchWindowException**: This is thrown when WebDriver tries to switch to an invalid window.

The below code can throw org.openqa.selenium.NoSuchWindowException if the window handle doesn’t exist or is not available to switch.

driver.switchTo().window(handle\_1);

**NoSuchFrameException:** When WebDriver is trying to switch to an invalid frame, NoSuchFrameException under NotFoundException class is thrown.

The below code can throw org.openqa.selenium.NoSuchFrameException if a frame “frame\_11” doesn’t exist or is not available.

driver.switchTo().frame(“frame\_11”);

**NoAlertPresentException:** when WebDriver tries to switch to an alert, which is not available.

org.openqa.selenium.NoAlertPresentException will be thrown If below automation code calls accept() operation on Alert() class when an alert is not yet on the screen.

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

**InvalidSelectorException:** when a selector is incorrect or syntactically invalid. This exception occurs commonly when XPATH locator is used.

**Consider the below example:**

clickXPathButtonAndWait(“//button[@type=’button’][100]”);

This would throw an InvalidSelectorExeption because the XPATH syntax is incorrect.

**ElementNotVisibleException:** This exception is thrown when WebDriver tries to perform an action on an invisible web element, which cannot be interacted with. That is, the web element is in a hidden state.

**ElementNotSelectableException:** ElementNotSelectableException indicates that the web element is present in the web page but cannot be selected.

**For example,** the below code can throw a ElementNotSelectableException if the id “swift” is disabled.

Select dropdown = new Select(driver.findElement(By.id(“swift”)));

**NoSuchSessionException:** This exception is thrown when a method is called after quitting the browser by WebDriver.quit(). This can also happen due to web browser issues like crashes and WebDriver cannot execute any command using the driver instance.

To see this exception, the code below can be executed.

driver.quit()

Select dropdown = new Select(driver.findElement(By.id(“swift”)));

**StaleElementRefernceException:** his exception says that a web element is no longer present in the web page.

This error is not the same as ElementNotVisibleException.

StaleElementReferenceException is thrown when an object for a particular web element was created in the program without any problem and however; this element is no longer present in the window. This can happen if there was a

* Navigation to another page
* DOM has refreshed
* A frame or window switch

WebElement firstName = driver.findElement(By.id(“firstname”));

driver.switchTo().window(Child\_Window);

element.sendKeys(“Aaron”);

In the code above, object firstName was created and then the window was switched. Then, WebDriver tries to type ‘Aaron’ in the form field. In this case StaleElementReferenceException is thrown.

**More About StaleElementReferenceException:**

Stale means old or we can say the element no longer appears on the DOM of web page.

eg: Let's assure that there is an login page where you are entering invalid credentials and clicking on login button and it will throw "invalid user name and password" with login page as previous. Same page is showed again but the address of web elements get changed. But we are trying to use the old reference and getting StaleElementReferenceException.

**Possible Causes for an Stale Element Exception:**

1. The element has been deleted entirely.
2. The element is no longer attached with the DOM on web page

**Solution 1: (Handle it using POM "@FindBy")**

If we use Page Object Model (or) Page Object design pattern you will never get Stale Element Reference Exception. When we use FindBy annotation in POM, webdriver locates the elements and update the reference (address of element) every time before performing any action on that. This is the major advantage of using Page Object Model.

**Solution 2: (Handle it using "try-catch block" within "for loop")**

If an element is not attached to the DOM then you could try using "try-catch block" within "for loop". Not only for loop we can use depends on our convenient do-while or while.

**Solution 3: (Handle it using "Explicit wait")**

Wait for the element until it gets available in DOM. Use ExpectedConditions.

wait.until(ExpectedConditions.presenceOfElementLocated(By.id("table")));

**Solution 4: (Handle it using "JavaScriptExecutor")**

**Maven**

Maven is a powerful *project management tool* that is based on POM (project object model). It is used for projects build, dependency and documentation.

It simplifies the build process.

For a smaller project with few functionalities or modules, a user can manually download JAR’s or dependencies and can add in the project. However, if a user is working on a medium or large project with many modules and a big team, then Maven plays a vital role in project management. It is a challenging task to ensure that all the team members are using the same dependency version. This issue can be solved by standardizing the dependency version for all.

Maven is widely used for dependency management in Java. It also provides a predefined folder structure to write the code. We can add different plugins and JARs in our project

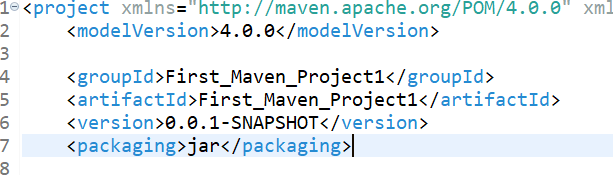
Without maven we may face issue like: Adding sets of jar in every project, crating project framework (folder etc)

Benefits of Maven:

* One can easily configure dependencies required for building, testing and running code using pom.xml
* Maven downloads the required file automatically from the repository
* Using Maven, one can manage the entire life-cycle of a test project

**POM** is an acronym for **Project Object Model**. The pom.xml file contains information of project and configuration information for the maven to build the project such as dependencies, build directory, source directory, test source directory, plugin, goals etc.

Maven reads the pom.xml file, then executes the goal.



|  |
| --- |
| **Node & Description** |
| **Project root**  This is project root tag. You need to specify the basic schema settings such as apache schema and w3.org specification. |
| **Model version**  Model version should be 4.0.0. |
| **groupId**  This is an Id of project's group. This is generally unique amongst an organization or a project. For example, a banking group com.company.bank has all bank related projects. |
| **artifactId**  This is an Id of the project. This is generally name of the project. For example, consumer-banking. Along with the groupId, the artifactId defines the artifact's location within the repository. |
| **version**  This is the version of the project. Along with the groupId, It is used within an artifact's repository to separate versions from each other. For example −  **com.company.bank:consumer-banking:1.0**  **com.company.bank:consumer-banking:1.1.** |

**Explain what is Maven artifact?**

Usually, an artifact is a JAR file which gets arrayed to a Maven repository. One or more artifacts a maven build produces such as compiled JAR and a sources JAR.

Each artifact includes a group ID, an artifact ID and a version string.

**Explain what is Maven Repository? What are their types?**

A Maven repository is a location where all the project jars, library jars, plugins or any other particular project related artifacts are stored and can be easily used by Maven.

Their types are local, central and remote

**7) Why Maven Plugins are used?**

Maven plugins are used to

* Create a jar file
* Create war file
* Compile code files
* Unit testing of code
* Documenting projects
* Reporting

**For POM what are the minimum required elements?**

The minimum required elements for POM are project root, modelVersion, groupID, artifactID and version.

**Explain how to run test classes in Maven?**

To run test classes in Maven, you need surefire plugin, check and configure your settings in setting.xml and pom.xml for a property named “test.”

**What are the advantages of Maven?**

* No need to add jar file in each project
* Creates right directory structure
* Builds and deploys the project

**What is the use of the exclusion element?**

The element is used to exclude dependencies.

**What is SNAPSHOT?**

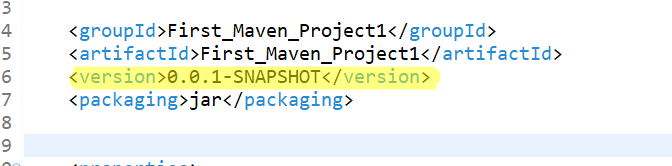
If multiple teams are working on different module of same application, it may happen that team 1 may release the library to repository very frequently. Then below problem may occurs:

* Team 1 should tell team 2 every time that they have released an updated code.
* Team 2 is required to update their pom.xml regularly to get the updated version.

To handle such situation SNAPSHOT concept comes into the picture.

SNAPSHOT is a special version that indicates a current development copy. Unlike regular versions, Maven checks for a new SNAPSHOT version in a remote repository for every build.

Now Team 1 team will release SNAPSHOT of its updated code every time to repository, say data-service: 0.0.1-SNAPSHOT, replacing an older SNAPSHOT jar.



**Snapshot vs Version**

In case of Version, if Maven once downloaded the mentioned version, say data-service:1.0, it will never try to download a newer 1.0 available in repository. To download the updated code, data-service version is be upgraded to 1.1.

In case of SNAPSHOT, Maven will automatically fetch the latest SNAPSHOT (data-service:1.0-SNAPSHOT) every time team 1 team build their project.

TDD Vs BDD

TDD or **Test Driven Development** and BDD or **Behaviour Driven Development** are the two software development techniques.

TDD stands for Test Driven Development. In this software development technique, we create the test cases first and then write the code underlying those test cases. Although TDD is a development technique, it can also be used for automation testing development.

The teams that implement TDD, take more time for development however, they tend to find very few defects. TDD results in improved quality of code and the code that is more reusable and flexible.

TDD also helps in achieving high test coverage of about 90-100%. The most challenging thing for developers following TDD is to write their test cases before writing the code.

TDD we will use selenium

BDD stands for Behavior Driven Development. BDD is an extension to TDD where instead of writing the test cases, we start by writing a behaviour. Later, we develop the code which is required for our application to perform the behaviour

| TDD | BDD |
| --- | --- |
| Stands for Test Driven Development. | Stands for Behaviour Driven Development. |
| The process starts by writing a test case. | The process starts by writing a scenario as per the expected behavior. |
| TDD focuses on how the functionality is implemented. | BDD focuses on the behavior of an application for the end user. |
| Test cases are written in a programming language. | Scenarios are more readable when compared to TDD as they are written in simple English format. |
| Changes in how the application functions impact a lot on the test cases in TDD. | BDD scenarios are not much impacted by the functionality changes. |
| Collaboration is required only between the developers. | Collaboration is required between all the stakeholders. |
| Might be a better approach for projects which involve API and third-party tools. | Might be a better approach for projects which are driven by user actions. For eg: e-commerce website, application system, etc. |
| Some of the tools which support TDD are: JUnit, TestNG, NUnit, etc. | Some of the tools which support BDD are SpecFlow, Cucumber, MSpec, etc. |
| Tests in TDD can only be understood by people with programming knowledge, | Tests in BDD can be understood by any person including the ones without any programming knowledge. |
| TDD reduces the likelihood of having bugs in your tests. | Bugs in tests are difficult to track when compared to TDD. |

**Jenkins**

Jenkins is an open-source automation tools that allow continuous integration. Jenkins build and test our software project.

Jenkins is very useful in integrating large number of test cases.

With the help of Jenkins we can achieve continuous integration (CI) with the help of plugins.

Suppose a organization is developing a project, then Jenkins will continuously test the project build and will detect the error in early stage.

**Jenkins with Selenium:**

* Running Selenium tests in Jenkins allows you to run your tests every time your software changes and deploy the software to a new environment when the tests pass.
* Jenkins can schedule your tests to run at specific time.
* You can save the execution history and Test Reports.
* Jenkins supports Maven for building and Testing a project in continuous integration.

Before proceeding read below three terms:

**What is Continuous Integration:**

Continuous integration is a software development method where members of the team can integrate their work at least once a day. In this method, every integration is checked by an automated build to search the error.

In continuous integration after a code commit, the software is built and tested immediately. In a large project with many developers, commits are made many times during a day. With each commit code is built and tested. If the test is passed, build is tested for Deployment. If the Deployment is a success, the code is pushed to production. This commit, build, test, and deploy is a continuous process, and hence the name continuous integration/deployment.

Working of Continuous Integration:

* Developers input code into their private terminals.
* After that is done, they commit the changes to the shared repository.
* The CI server monitors the repository and analyses changes as they occur.
* Continuous Integration builds the system and runs unit and integration tests.
* The Continuous Integration server gives the team reports of the successful build.
* If the build or tests fail, the server alerts the development team.
* The team will fix the issues as soon as is possible.
* It continues to integrate and run tests throughout the entire project.

**What is Continuous Delivery:**

Continuous delivery is a software engineering method in which a team develops software products in a short cycle. It ensures that software can be easily released at any time.

The main aim of continuous delivery is to build, test, and release software with good speed and frequency. It helps you to reduce the cost time and risk of delivering changes by allowing for frequent updates in production.

**What is Continuous Deployment:**

Continuous deployment is a software engineering process in which product functionalities are delivered using automatic deployment. It helps testers to validate whether the codebase changes are correct and stable or not.

The team can achieve continuous deployment by relying on infrastructure that automates different testing steps. Once each integration meets this release criteria, the application is updated with a new code.

**What is the difference between CI and CD**

Continuous integration (CI) is practice that involves developers making small changes and checks to their code. Due to the scale of requirements and the number of steps involved, this process is automated to ensure that teams can build, test, and package their applications in a reliable and repeatable way. [CI](https://www.synopsys.com/glossary/what-is-continuous-integration.html) helps streamline code changes, thereby increasing time for developers to make changes and contribute to improved software.

[Continuous delivery](https://www.synopsys.com/glossary/what-is-continuous-delivery.html) (CD) is the automated delivery of completed code to environments like testing and development. CD provides an automated and consistent way for code to be delivered to these environments.

[Continuous deployment](https://www.synopsys.com/glossary/what-is-continuous-development.html) is the next step of continuous delivery. Every change that passes the automated tests is automatically placed in production, resulting in many production deployments.

Continuous deployment should be the goal of most companies that are not constrained by regulatory or other requirements.

In short, CI is a set of practices performed as developers are writing code, and CD is a set of practices performed after the code is completed.

**To understand more about Jenkins read below text:**

Let's consider a scenario where the complete source code of the application was built and then deployed on test server for testing. It sounds like a perfect way to *develop software*, but this process has many problems.

* Developer teams have to wait till the complete software is developed for the test results.
* There is a high prospect that the test results might show multiple bugs. It was tough for developers to locate those bugs because they have to check the entire source code of the applications
* It slows the software delivery process.
* Continuous feedback pertaining to things like architectural or coding issues, build failures, test status and file release uploads was missing due to which the quality of software can go down.
* The whole process was manual which increases the threat of frequent failure.

It is obvious from the above stated problems that not only the software delivery process became slow but the quality of software also went down. This leads to customer dissatisfaction.

So to overcome such problem there was a need for a system to exist where developers can continuously trigger a build and test for every change made in the source code.

This is what Continuous Integration (CI) is all about. Jenkins is the most mature Continuous Integration tool available so let us see how Continuous Integration with Jenkins overcame the above shortcomings.

Let's see a generic flow diagram of Continuous Integration with Jenkins:



**Let's see how Jenkins works**. The above diagram is representing the following functions:

* First of all, a developer commits the code to the source code repository. Meanwhile, the Jenkins checks the repository at regular intervals for changes.
* Soon after a commit occurs, the Jenkins server finds the changes that have occurred in the source code repository. Jenkins will draw those changes and will start preparing a new build.
* If the build fails, then the concerned team will be notified.
* If built is successful, then Jenkins server deploys the built in the test server.
* After testing, Jenkins server generates a feedback and then notifies the developers about the build and test results.
* It will continue to verify the source code repository for changes made in the source code and the whole process keeps on repeating.

**Scheduling a Job in Jenkins Timely:**

First click on job and go to configure now, where you will see the option named as build periodically. Here in text box, you can pass the value in proper format.

**1 — Minute (0–59)**

**2 — Hours (0–23)**

**3 — Day in a month (1–31 or or JAN–DEC), the abbreviations are not case-sensitive**

**4 — Month (1–12 or), the abbreviations are not case-sensitive**

**5 — Day in week (0–6) where 0 means Sunday and 6 is equal to Saturday. ‘7’ can also be referred to Sunday.**

**6 — Year (2020–2022), it is not a required field but can be used for scheduling particular range of year or a specific year**

**Commas ( , ) are used to separate items of a list. For example, using "MON,WED,FRI" in the 5th field (day of week) means Mondays, Wednesdays and Fridays.**

**Hyphen ( - ) define ranges. For example, 2000–2010 indicates every year between 2000 and 2010, inclusive.**

For example:

* **20 7 \* \* \* means that it will run every day at exact 7:20 am.**
* **Here only the first two parameters are specifically defined. According to the key defined above, first parameter denotes to the minute of an hour and second parameter to the hour. In the case of 3rd, 4th and 5th parameters asterisks ‘\*’ are used which means that no specific day, month and weekday is defined so it would trigger every day for whole month.**
* **0 4 \* \* 1,3 means that it will run on Monday and Wednesday at exact 4:00 AM. Here comma separated week days are mentioned. Using ’,’ for any parameter means that user wants to mention multiple values. So “1,3” means it will trigger on Monday and Wednesday.**
* **\* 10 10 \* \* means every minute, between 10:00 AM and 10:59 AM, on day 10 of every month.**
* **15 2 \* 12 \* means at 02:15 AM every day of the month December. Here the month parameter is used to tell that the user only wants to run this job in December.**
* **0 10 \* JUN-DEC \* 2020,2022 means at 10:00 AM, June through December, only in 2020 and 2022**

To schedule your build every 5 minutes, this will do the job : **H/5 \* \* \* \***

To the job 5th Minute of every Hour 5 \* \* \* \*

To schedule your build at 6:00PM and 1 AM every day – H 1,18 \* \* \*

To schedule your build start daily at morning – H 09 \* \* 1-5

**Git and GitHub**

**What is Version Control:**

**A** version control is a system that **records changes to a file** or set of files over time so that you can recall specific version later.

**Git:**

**Git** is an **open-source distributed version control system**. It is designed to handle minor to major projects with high speed and efficiency. It is developed to co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace.

Git helps you keep track of the changes you make to your code. If at any point while coding you hit a fatal error and don’t know what is causing it you can always revert back to stable state.

Git also helps you **synchronise code** between multiple people. So imagine you and your friend are collaborating on a project. You both are working on the same project files. Now Git takes those changes you and your friend made independently and merges them to a single “**Master**” repository.

**Features of Git:**

* **Open Source:**
* **Scalable:**
* **Secure:**
* **Branching and Merging:**

**GitHub:**

GitHub is a Git repository hosting service that provides a web-based graphical interface.

GitHub is an immense platform for code hosting. It supports version controlling and collaboration and allows developers to work together on projects. It offers both distributed version control and source code management (SCM) functionality of Git. It also facilitates collaboration features such as bug tracking, feature requests, task management for every project.

Essential components of the GitHub are:

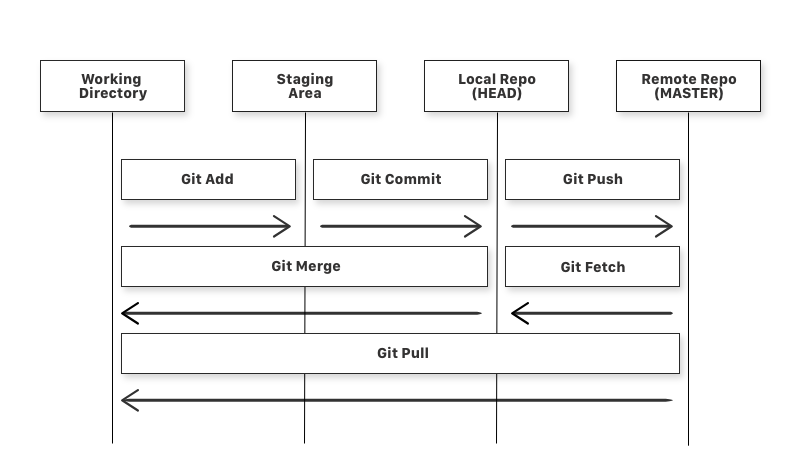
* Repositories
* Branches
* Commits
* Pull Requests
* Git (the version control tool GitHub is built on)

**Few keyword used in Git:**

**What is a Repository?**

A **repository** a.k.a. **repo** is nothing but a collection of source code.

**Work flow of Git**



If you consider a file in your working directory, it can be in three stages

1. **It can be staged:** Which means the file with the updated changes are marked to be committed but not yet committed.
2. **It can be modified**: Which means the file with the updated changes are not yet stored in the local repository.
3. **It can be committed:** Which means the changes that you made to your local file are safely stored in the local repository.

* **git add** is a command used to add a file that is in the working directory to the staging area.
* **git commit**is a command used to add all files that are staged to the local repository.
* **git push** is a command used to add all committed files in the local repository to the remote repository. So in the remote repository, all files and changes will be visible to anyone with access to the remote repository.
* **git fetch** is a command used to get files from the remote repository to the local repository but not into the working directory.
* **git merge** is a command used to get the files from the local repository into the working directory.
* **git pull** is command used to get files from the remote repository directly into the working directory. It is equivalent to a git fetch and a git merge .

**Difference between commit and push:**

**The main difference between commit and push is that the changes to be made in the process of the commit in a coding file is always to be under the limits of the local repository, and on the other hand, the changes to be made in the process of push which comes after commit in a coding file is always to be under the limits of remote respiratory.**

**Changes made in commit goes to local depository.**

**Changes made in push goes to remote repository.**

**Branch**:

**A branch is a version of the repository that diverges from the main working project.** It is an essential feature available in most modern version control systems. A Git project can have more than one branch. We can perform many operations on Git branch-like rename, list, delete, etc.

**Master Branch:**

Master Branch is repository's "default" branch.

**Clone:**

The **git clone** is a Git command-line utility. It is used to make a copy of the target repository or clone it. If I want a local copy of my repository from GitHub, this tool allows creating a local copy of that repository on your local directory from the repository URL.

**Fetch:**

It is used to fetch branches and tags from one or more other repositories, along with the objects necessary to complete their histories. It updates the remote-tracking branches.

**Master:**

Master is a naming convention for Git branch. It's a default branch of Git. After cloning a project from a remote server, the resulting local repository contains only a single local branch. This branch is called a "master" branch. It means that "master" is a repository's "default" branch.

**Merge:**

Merging is a process to put a forked history back together. The git merge command facilitates you to take the data created by git branch and integrate them into a single branch.

**Pull//Pull Request:**

The term Pull is used to receive data from GitHub. It fetches and merges changes on the remote server to your working directory. The **git pull command** is used to make a Git pull.

Pull requests are a process for a developer to notify team members that they have completed a feature. Once their feature branch is ready, the developer files a pull request via their remote server account. Pull request announces all the team members that they need to review the code and merge it into the master branch.

**Push**

The push term refers to upload local repository content to a remote repository. Pushing is an act of transfer commits from your local repository to a remote repository. Pushing is capable of overwriting changes; caution should be taken when pushing.

**Repository:**

In Git, Repository is like a data structure used by VCS to store metadata for a set of files and directories. It contains the collection of the file as well as the history of changes made to those files. Repositories in Git is considered as your project folder. A repository has all the project-related data. Distinct projects have distinct repositories.

**Stashing:**

Sometimes you want to switch the branches, but you are working on an incomplete part of your current project. You don't want to make a commit of half-done work. Git stashing allows you to do so. The **git stash command** enables you to switch branch without committing the current branch.

**Git Revert**

In Git, the term revert is used to revert some commit. To revert a commit, **git revert** command is used. It is an undo type command. However, it is not a traditional undo alternative.

**Git Fork:**

A fork is a rough copy of a repository. Forking a repository allows you to freely test and debug with changes without affecting the original project.

Great use of using forks to propose changes for bug fixes. To resolve an issue for a bug that you found, you can:

* Fork the repository.
* Make the fix.
* Forward a pull request to the project owner.

**Some useful Git Commands:**

how to commit

- git status

- git add "pageclass1"

- git commit -m "fixing of bugs in any module"

- git push origin xyzbranch

**git init [repository name]:** This command is used to start a new repository.

**Usage: git clone [url]**: This command is used to obtain a repository from an existing URL.

**git add [file]:** This command adds a file to the staging area.

**Usage: git add \***: This command adds one or more to the staging area.

**git commit -m “[ Type in the commit message]”**: This command records or snapshots the file permanently in the version history.



**git commit -a**: This command commits any files you’ve added with the git add command and also commits any files you’ve changed since then.

**git diff**: This command shows the file differences which are not yet staged.

**git diff –staged: Th**is command shows the differences between the files in the staging area and the latest version present.

**git diff [first branch] [second branch]:** This command shows the differences between the two branches mentioned.

**git reset [file]:** This command unstages the file, but it preserves the file contents.

**git reset [commit]:** This command undoes all the commits after the specified commit and preserves the changes locally.

**git reset –hard [commit]** This command discards all history and goes back to the specified commit.

**Usage: git status:** This command lists all the files that have to be committed.



**Usage: git rm [file]:** This command deletes the file from your working directory and stages the deletion.

Git Rm Command - Git Commands - Edureka

**git log:** This command is used to list the version history for the current branch.

**git log –follow[file]:** This command lists version history for a file, including the renaming of files also.

**git show [commit]**: This command shows the metadata and content changes of the specified commit.

**git branch**

This command lists all the local branches in the current repository.

Git Branch Command - Git Commands - Edureka

**git branch [branch name]**: This command creates a new branch.

Git Branch Command - Git Commands - Edureka

**git merge [branch name]**

This command merges the specified branch’s history into the current branch.

Git Merge Command - Git Commands - Edureka

**git remote add [variable name] [Remote Server Link]**

This command is used to connect your local repository to the remote server.

Git Remote Command - Git Commands - Edureka

**git push [variable name] master**

This command sends the committed changes of master branch to your remote repository.



**Usage: git push [variable name] [branch]**

This command sends the branch commits to your remote repository.



**Usage: git push –all [variable name]**

This command pushes all branches to your remote repository.



**Usage: git push [variable name] :[branch name]**

This command deletes a branch on your remote repository.



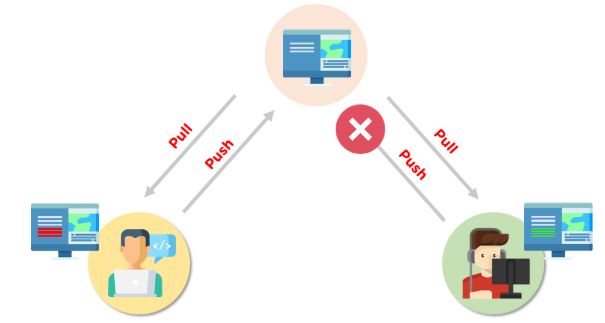
**git pull [Repository Link]**

This command fetches and merges changes on the remote server to your working directory.

**What is a Git Merge Conflict?**

A merge conflict is an event that takes place when Git is unable to automatically resolve differences in code between two commits. Git can merge the changes automatically only if the commits are on different lines or branches.

The following is an example of how a Git merge conflict works:



Let’s assume there are two developers: Developer A and Developer B. Both of them pull the same code file from the remote repository and try to make various amendments in that file. After making the changes, Developer A pushes the file back to the remote repository from his local repository. Now, when Developer B tries to push that file after making the changes from his end, he is unable to do so, as the file has already been changed in the remote repository.

To prevent such conflicts, developers work in separate isolated branches. The Git merge command combines separate branches and resolves any conflicting edits.

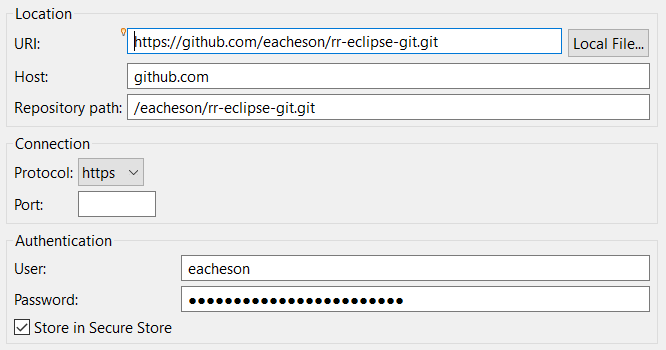
here are a few steps that could reduce the steps needed to resolve merge conflicts in Git.

1. The easiest way to resolve a conflicted file is to open it and make any necessary changes
2. After editing the file, we can use the git add a command to stage the new merged content
3. The final step is to create a new commit with the help of the git commit command
4. Git will create a new merge commit to finalize the merge

Let us now look into the Git commands that may play a significant role in resolving conflicts.

**Import your repository into Eclipse**

* Open Eclipse, with EGit, installed
* In Eclipse, choose File, then Import
* In the dialogue that opens, choose Git > Projects from Git and click Next
* Click on Clone URI, then click Next
* Here the details should already be filled out for you, as shown below. Ensure that your GitHub username and password are entered under Authentication, and click Next.



Now that you've got a local copy and a copy on your GitHub account, there are four things that you'll need to know how to do in order to collaborate with SparkFun:

* **Commit** - committing is the process which records changes in the repository. Think of it as a snapshot of the current status of the project. Commits are done locally.
* **Push** - pushing sends the recent commit history from your local repository up to GitHub. If you're the only one working on a repository, pushing is fairly simple. If there are others accessing the repository, you may need to pull before you can push.
* **Pull** - a pull grabs any changes from the GitHub repository and merges them into your local repository.
* **Sync** - syncing is like pulling, but instead of connecting to your GitHub copy of the forked repo, it goes back to the original repository and brings in any changes. Once you've synced your repository, you need to push those changes back to your GitHub account.

Master Branch: We can’t merge error code in master branch, first we need work on local branch,

Suppose user A, made a branch X, first A will work on local and commit to branch X(local repository)

Then this code will be reviewed, and everything is fine will merge Master Branch.

Automation Testing interview question:

**Where have you applied OOPS in Automation Framework:**

**Abstraction:**

In Page Object Model design pattern, we write locators (such as id, name, Xpath etc.,) and the methods in a Page Class. We utilize these locators in tests but we can’t see the implementation of the methods. Literally we hide the implementations of the locators from the tests.

**Interface:**

statement we know in Selenium WebDriver **driver = new FirefoxDriver();**

WebDriver itself is an Interface. So based on the above statement **WebDriver driver = new FirefoxDriver();** we are initializing Firefox browser using Selenium WebDriver. It means we are creating a *reference* variable (driver) of the interface (WebDriver) and creating an Object. Here WebDriver is an Interface as mentioned earlier and FirefoxDriver is a class.

**Inheritance:**

We create a Base Class in the Automation Framework to initialize WebDriver interface, WebDriver waits, Property files, Excels, etc., in the Base Class.

We extend the Base Class in other classes such as Tests and Utility Class.

Here we extend one class (Base Class like WebDriver Interface) into other class (like Tests, Utility Class) is known as Inheritance.

**Polymorphism:**

Polymorphism allows us to perform a task in multiple ways.

* **Method Overloading:**

A class having multiple methods with same name but different parameters is called Method Overloading.

**Implicit wait** in Selenium. Implicit wait is an example of overloading. In Implicit wait we use different time stamps such as SECONDS, MINUTES, HOURS etc.,

**Action class** in TestNG is also an example of overloading.

**Assert class** in TestNG is also an example of overloading.

* **Method Overriding:**

We use a method which was already implemented in another class by changing its parameters. Declaring a method in child class which is already present in the parent class is called Method Overriding. Examples are **get**and **navigate** methods of different drivers in Selenium.

**Encapsulation:**

All the classes in a framework are an example of Encapsulation. In POM classes, we declare the data members using **@FindBy** and initialization of data members will be done using Constructor to utilize those in methods.

**List out the test types that are supported by Selenium**

For web-based application testing selenium can be used for below testing:

a) Functional,

b) Regression

**Explain what is assertion in Selenium and what are the types of assertion?**

Assertion is used as a verification point. It verifies that the state of the application conforms to what is expected.  The types of assertion are “assert” , “verify” and “waitFor”.

**Explain the difference between single and double slash in X-path?**

**Single slash ‘/ ’**

* Single slash ( / ) start selection from the document node
* It allows you to create ‘absolute’ path expressions

**Double Slash ‘// ’**

* Double slash ( // ) start selection matching anywhere in the document
* It enables to create ‘relative’ path expressions

**What is the difference between verify and assert commands?**

**Assert:**  Assert allows to check whether an element is on the page or not. The test will stop on the step failed, if the asserted element is not available. In other words, the test will terminate at the point where check fails.

**Verify:** Verify command will check whether the element is on the page, if it is not then the test will carry on executing.  In verification, all the commands are going to run guaranteed even if any of test fails.

**Difference between CSS selector and Xpath:**

Xpath allows bidirectional flow which means the traversal can be both ways from parent to child and child to parent as well. Css allows only one directional flow which means the traversal is from parent to child only.

Xpath is slower in terms of performance and speed. Css has better performance and speed than xpath.

There are two types of xpath – absolute and relative. But css has no such types.

**While using click command can you use screen coordinate?**

To click on specific part of element, you would need to use clickAT command.  ClickAt command accepts element locator and x, y co-ordinates as arguments-

clickAt (locator, cordString)

**What is the difference between setSpeed() and sleep() methods?**

Both will delay the speed of execution.

**Thread.sleep ()** :  It will stop the current (java) thread for the specified period of time.  Its done only once

* It takes a single argument in integer format

Ex: thread.sleep(2000)- It will wait for 2 seconds

* It waits only once at the command given at sleep

**SetSpeed ()** :  For specific amount of time it will stop the execution for every selenium command.

* It takes a single argument in integer format

Ex: selenium.setSpeed(“2000”)- It will wait for 2 seconds

* Runs each command  after setSpeed delay by the number of milliseconds mentioned in set Speed

This command is useful for demonstration purpose or if you are using a slow web application

**Mention what is the difference between Implicit wait and Explicit wait?**

**Implicit Wait**: Sets a timeout for all successive Web Element searches. For the specified amount of time it will try looking for element again and again before throwing a NoSuchElementException.  It waits for elements to show up.

**Explicit Wait :**  It is a one-timer, used for a particular search.

**Explain what is the difference between find elements () and find element () ?**

**find element ():** It finds the first element within the current page using the given “locating mechanism”.  It returns a single WebElement

**findElements ():** Using the given “locating mechanism” find all the elements within the current page.  It returns a list of web elements.

**Explain what is Data driven framework and Keyword driven framework?**

**Data driven framework:**  In this framework, the test data is separated and kept outside the Test Scripts, while Test Case logic resides in Test Scripts.  Test data is read from the external files (Excel Files) and are loaded into the variables inside the Test Script.  Variables are used for both for input values and for verification values.

**Keyword driven framework:** The keyword driven frameworks require the development of data tables and keywords, independent of the test automation.  In a keyword driven test, the functionality of the application under test is documented in a table as well as step by step instructions for each test.

**What is Object Repository?**

An object repository is an essential entity in any UI automations which allows a tester to store all object that will be used in the scripts in one or more centralized locations rather than scattered all over the test scripts.

**Explain how you can find broken images in a page using Selenium Web driver.**

To find the broken images in a page using Selenium web driver is

* Get XPath and get all the links in the page using tag name
* In the page click on each and every link
* Look for 404/500 in the target page title

**Explain how you can handle colors in web driver?**

Use getCssValue(arg0) function to get the colors by sending ‘color’ string as an argument

**Explain how you can switch between frames?**

To switch between frames webdrivers **[ driver.switchTo().frame() ]** method takes one of the three possible arguments

* A number:  It selects the number by its (zero-based) index
* A name or ID: Select a frame by its name or ID
* Previously found WebElement: Using its previously located WebElement select a frame

**Mention 5 different exceptions you had in Selenium web driver?**

* NoSuchWindowException
* NoSuchFrameException
* NoSuchElementException
* NoAlertPresentException
* InvalidSelectorException
* TimeoutException
* ElementNotVisibleException
* ElementNotSelectableException
* NoSuchSessionException
* StaleElementReferenceException

**Explain using Webdriver how you can perform double click ?**

You can perform double click by using

**Syntax- Actions act = new Actions (driver);**

**act.doubleClick(webelement);**

**Which web driver implementation is fastest?**

HTMLUnit Driver implementation is fastest, HTMLUnitDriver does not execute tests on browser but plain http request, which is far quick than launching a browser and executing tests

**What is the difference between getWindowhandles() and getwindowhandle()?**

getwindowhandles(): It is used to get the address of all the open browser and its return type is Set<String>

getwindowhandle(): It is used to get the address of the current browser where the control is and return type is string

**In Selenium what are Breakpoints and Startpoints?**

* **Breakpoints:**When you implement a breakpoint in your code, the execution will stop right there. This helps you to verify that your code is working as expected.
* **Startpoints:** Startpoint indicates the point from where the execution should begin. Startpoint can be used when you want to run the testscript from the middle of the code or a breakpoint.

**Mention what is desired capability? How is it useful in terms of Selenium?**

The desired capability is a series of key/value pairs that stores the browser properties like browser name, browser version, the path of the browser driver in the system, etc. to determine the behavior of the browser at run time.

For Selenium,

* It can be used to configure the driver instance of Selenium WebDriver.
* When you want to run the test cases on a different browser with different operating systems and versions.

**Mention why do you need Session Handling while working with Selenium?**

While working with Selenium, you need Session Handling. This is because, during test execution, the Selenium WebDriver has to interact with the browser all the time to execute given commands. At the time of execution, it is also possible that, before current execution completes, someone else starts execution of another script, in the same machine and in the same type of browser. So to avoid such situation you need Session Handling.

**Mention what are the advantages of Using Git Hub for Selenium?**

* Multiple people when they work on the same project, they can update project details and inform other team members simultaneously.
* Jenkins can help you to build the project from the remote repository regularly. This helps you to keep track of failed builds.

**When Would You Prefer Manual Testing Over Automation Testing?**

I prefer manual testing rather than automation testing when the project is short-term, and if it will be costly and time-consuming to craft scripts compared to manual. I would also use manual testing if the module/application has just been developed, thus has no previous test cases. Additionally, I will use it when I need to perform usability, Ad-hoc, or explanatory testing and when the project needs to be flexible.

The differences between get() and navigate() methods are listed below.

| **get()** | **navigate()** |
| --- | --- |
| It is responsible for loading the page and waits until the page has finished loading. | It is only responsible for redirecting the page and then returning immediately. |
| It cannot track the history of the browser. | It tracks the browser history and can perform back and forth in the browser. |