**SELENIUM PART-1**

**Selenium Syllabus: -**

* From Basic to Advance Selenium Concepts
* TestNG, ANT, Maven and Jenkins
* Basics of Java
* Different other tools like log4j, cucumber that can be used in conjunction with selenium
* Data Driven Testing
* Interview Questions and Quizzes

Sections 1 and 2 includes: -

* Selenium Introduction
  + - Selenium History
    - Selenium Architecture
* Installation of Java, Eclipse and Selenium Jars download.
  + - Configuration of Selenium Jars into Eclipse.
    - Brush up of Basic Java Concepts

Sections 3 and 4 includes: -

* First WebDriver Program
  + - Different WebDriver methods
    - How to invoke different browsers using WebDriver.
* Different Locators Techniques and tools used
  + - Firebug and Firepath Add-onsrffqff
    - Element identification by different locating mechanism

Section 5 includes: -

* Advance ways of locating webelement\objects
  + - Writing customized xpath and cssselector
    - Tips and Tricks for interview

Section 6 includes: -

* Techniques to automate web UI
  + - Handling Dropdown/Select element
    - Handling checkboxes, radiobuttons
    - Handling Javascript Alert
    - WebElement validation techniques
    - Interview Questions

Sections 7,8 and 9 includes: -

* Techniques to Automate advanced WebUI
  + - Handling Ajax/Mouse interaction
    - Handling window, multiple windows
    - Handling Frames
* Real Time Exercises (end to end Programming)
* Practical problems and Methods to Handle them with Selenium

Sections 10,11,12,13 and 14: -

* Famous Interview Questions
* Cross Browser Testing with Selenium Grid
* Cloud Testing with Selenium-Sauce Labs
* Performance Testing on Selenium Test cases
* Selenium 3.0 -Mobile Automation Testing Basics

Sections 15 to 20: -

* TestNG
* Data Driven Framework using properties file
* ANT build management tool
* Generate xslt report
* Jenkins Continuous Integration tool
* Maven Build Management tool

Sections 21 to 26: -

* Logging Framework – log4j
* PageObject Pattern and PageFactory
* Data Driven Testing using Apache POI
* Database connection to Selenium Test cases
* Java OOPS Basics for Selenium
* Cucumber Behaviour Driven Data –Framework

**Introduction:**

* **Selenium** is an open source automation testing tool. It is used exclusively for web based applications
* You can work on multiple operating systems using selenium
* Platforms Supported by Selenium – Windows, OS X, Linux, Solaris
* Following languages are used with selenium.

- Java  
- C#  
- Ruby  
- Python  
- PHP  
- Perl

* Selenium Browsers Support:

- InternetExplorer

- Firefox

- Chrome

- Safari

**Below few important interview questions:-**

* What makes Selenium Unique from other Automation tools?
* Explain WebDriver Architecture.
* Why not older Version Selenium RC 1.0?
* What are different Versions of Selenium?
* What are different browsers does Webdriver Support?
* In how many languages we can write Selenium code?

**1. Selenium History**

Selenium came into existence in 2004 when GUY named Jason Huggins was testing an internal application at ThoughtWorks. Selenium is a set of different software tools each with a different approach to supporting test automation.

1.1 Selenium Components:

* [Selenium IDE](http://en.wikipedia.org/wiki/Selenium_(software)#Selenium_IDE)
* [Selenium Remote Control](http://en.wikipedia.org/wiki/Selenium_(software)#Selenium_Remote_Control)
* [Selenium WebDriver](http://en.wikipedia.org/wiki/Selenium_(software)#Selenium_WebDriver)
* [Selenium Grid](http://en.wikipedia.org/wiki/Selenium_(software)#Selenium_Grid)

Let’s have a look on each of these components:

[**Selenium IDE**](http://en.wikipedia.org/wiki/Selenium_(software)#Selenium_IDE)

Selenium IDE is a complete [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) for Selenium tests. It is implemented as a [Firefox extension](http://en.wikipedia.org/wiki/Add-on_(Mozilla)), and allows *recording, editing, and debugging* tests. It was previously known as Selenium Recorder.

Scripts are recorded in Selenese, a special test scripting language for Selenium. Selenese provides commands for performing actions in a browser (click a link, select an option), and for retrieving data from the resulting pages.

It is not only just play back tool, it can record user actions as they are performed and then export them as reusable script in one of many programming languages that can be later executed.

**Drawbacks:**

As it will come with only Firefox add-in, what if you want to test your application which works only in Internet explorer or some any other browser?

Selenium IDE is not suitable when you want to built robust frameworks.

Selenium IDE doesn’t provide iteration or conditional statements for test scripts.

**Selenium Rc(Selenium 1.0)**

Selenium RC was first component in the market which controls a browser from a language of your choice. The underlying functionality of Selenium Remote Control is that it uses JavaScript library (Selenium core) that could drive interactions with the page, allows us to automatically rerun tests against multiple browsers.

**Selenium RC Architecture**

*There is a selenium-RC server which acts as proxy between our Driver code and Application under Test*

 The client/driver establishes a connection with the selenium-RC server.  
2. Selenium RC server launches a browser (or reuses an old one) with a URL that injects Selenium-Core’s JavaScript into the browser-loaded web page.  
3. The client-driver passes a Selenese command to the server.  
4. The Server interprets the command and then triggers the corresponding JavaScript execution to execute that command within the browser.  
5. Selenium-Core instructs the browser to act on that first instruction, typically opening a page of the AUT.  
6. The browser receives the open request and asks for the website’s content from the Selenium RC server (set as the HTTP proxy for the browser to use).  
7. Selenium RC server communicates with the Web server asking for the page and once it receives it, it sends the page to the browser masking the origin to look like the page comes from the same server as Selenium-Core (this allows Selenium-Core to comply with the Same Origin Policy).  
8. The browser receives the web page and renders it in the frame/window reserved for it

**Drawbacks:**

As it is entirely use JavaScript to talk to browser, it leads to significant weakness. Every browser imposes very strict security rules on the JavaScript being executed to protect the users from malicious scripts.[

There is no support for Android and IOS Platform. Server need to be started every time to run a program. Lot of Limitations when an Application has Rich API with dynamic elements. Native keyboard and mouse events cannot be handled in efficient manner

**Webdriver: (Selenium2.0)**

This brand new automation tool provides all sorts of features, including a more cohesive and object-oriented API as well as an answer to the limitations of the old implementation. It supports the WebDriver API underlying technology, along with the Selenium underlying technology.

The architecture of Selenium Webdriver is entirely different from RC.Unlike RC there is no proxy server between AUT and Code.

It makes direct calls to browser native API to get the things executed. Unlike RC it does not Use any proxy server to talk to browser. WebDriver uses the most appropriate way to access the browser API. If we look at Firefox, it uses JavaScript to access the API. If we look at Internet Explorer, it uses C++. That means that it sometimes needs direct help from browser development team, by this approach we can control browsers in the best possible way but has the downside that new browsers entering the market will not be supported straight away like we can with RC.

As Webdriver directly talks with browser we can overcome the limitations of JavaScript security model which we have faced with Selenium Core in RC

Selenium WebDriver supports the following browsers along with the operating systems these browsers are compatible with.

* + Google Chrome 12.0.712.0+
  + Internet Explorer 6, 7, 8, 9 - 32 and 64-bit where applicable
  + Firefox 3.0, 3.5, 3.6, 4.0, 5.0, 6, 7
  + Opera 11.5+
  + HtmlUnit 2.9
  + Android – 2.3+ for phones and tablets (devices & emulators)
  + iOS 3+ for phones (devices & emulators) and 3.2+ for tablets (devices & emulators)

Languages on which Webdriver Supports:

* Java
* JavaScript
* Ruby
* PHP
* Python
* Perl
* C#

You can use any of the above language to Automate application. There is no rule like as your website is built with C#,so you have to use only C# code in Webdriver to automate your application. The language you write your code is Application independent.

**Why Webdriver??**

* In addition, Selenium 2 still runs Selenium 1’s Selenium RC interface for backwards compatibility
* No server required to start.
* No support for Android and Iphone platform in RC
* Can Handle rich API
* Can handle Mouse movements
* It directly talks to browser. unlike RC it does not use any proxy server
* It supports all the Latest Versions of Firefox
* All future enhancements can be done in Webdriver only.

[**Selenium Grid**](http://en.wikipedia.org/wiki/Selenium_(software)#Selenium_Grid)

Selenium Grid allows you to run your tests in parallel i.e. different tests can run at the same time on different remote machines. Also, if you must run your test suite on multiple environments you can have different remote machines supporting and running your tests in them at the same time. In each case Selenium Grid greatly improves the time it takes to run your suite by making use of parallel processing.

**Interface:**

A Java interface defines a set of methods but does not implement them. A class that implements the interface agrees to implement all of the methods defined in the interface.

Webdriver is the Interface.

All Known Implementing Classes:

[AndroidDriver](http://selenium.googlecode.com/svn/trunk/docs/api/java/org/openqa/selenium/android/AndroidDriver.html), [AndroidWebDriver](http://selenium.googlecode.com/svn/trunk/docs/api/java/org/openqa/selenium/android/library/AndroidWebDriver.html), [ChromeDriver](http://selenium.googlecode.com/svn/trunk/docs/api/java/org/openqa/selenium/chrome/ChromeDriver.html), [EventFiringWebDriver](http://selenium.googlecode.com/svn/trunk/docs/api/java/org/openqa/selenium/support/events/EventFiringWebDriver.html), [FirefoxDriver](http://selenium.googlecode.com/svn/trunk/docs/api/java/org/openqa/selenium/firefox/FirefoxDriver.html), [HtmlUnitDriver](http://selenium.googlecode.com/svn/trunk/docs/api/java/org/openqa/selenium/htmlunit/HtmlUnitDriver.html),  [internetExplorerDriver](http://selenium.googlecode.com/svn/trunk/docs/api/java/org/openqa/selenium/ie/InternetExplorerDriver.html),[IPhoneDriver](http://selenium.googlecode.com/svn/trunk/docs/api/java/org/openqa/selenium/iphone/IPhoneDriver.html), [IPhoneSimulatorDriver](http://selenium.googlecode.com/svn/trunk/docs/api/java/org/openqa/selenium/iphone/IPhoneSimulatorDriver.html), [RemoteWebDriver](http://selenium.googlecode.com/svn/trunk/docs/api/java/org/openqa/selenium/remote/RemoteWebDriver.html), [SafariDriver](http://selenium.googlecode.com/svn/trunk/docs/api/java/org/openqa/selenium/safari/SafariDriver.html)

Steps for creating class object with reference to interface

1. Choose the Browse on which you want to perform Automation and Select the respective class to implement Webdriver

Ex: [FirefoxDriver](http://selenium.googlecode.com/svn/trunk/docs/api/java/org/openqa/selenium/firefox/FirefoxDriver.html)()

2. Create an object for the Class and assign name to it

Selenium=New [FirefoxDriver](http://selenium.googlecode.com/svn/trunk/docs/api/java/org/openqa/selenium/firefox/FirefoxDriver.html)();

1. Then make Class object reference to the Webdriver Interface which you want to implement

Webdriver Selenium=New [FirefoxDriver](http://selenium.googlecode.com/svn/trunk/docs/api/java/org/openqa/selenium/firefox/FirefoxDriver.html)();

* Basis methods of Webdriver:
  + Get();
* Load a new web page in the current browser window
  + getCurentURl();  
    Get a string representing the current URL that the browser is looking at.
  + getTitle()
* The title of the current page
  + getPageSource()
* Get the source of the last loaded page.
  + Quit();
* Quits this driver, closing every associated window.
  + Close();

Close the current window, quitting the browser if it's the last window currently open.

Below is the first program which gives basics on geckodriver and chromedrive to open firefox & chrome respectively.

**WebdriverMethod:**

**package** basics;

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

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

**public class** BrowserInvocation {

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

System.***out***.println("Welcome to Selenium !!!");

**System.*setProperty*("webdriver.gecko.driver", "E:\\SOFTWARE\\geckodriver-v0.19.1-win64\\geckodriver.exe");**

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

driver.get("https://www.facebook.com/");

System.***out***.println(driver.getTitle()); //it will give the title

//System.out.println(driver.getPageSource()); //To get the page source

System.***out***.println("Webpage is opened now !!!");

System.***out***.println(driver.getCurrentUrl()); // retrieves the current URL that is opened the browser

//driver.close(); //Closes the entire browser

driver.quit(); // only closes the tab that the link is opened

}

}

**This is my first code to send some text in some field by using ID, NAME & LINKTEXT locator strategy:**

**package** basics;

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

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

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

**public class**  Locator {

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

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

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

driver.get("https://www.facebook.com/"); //URL OF THE BROWSER

driver.findElement(By.*id*("email")).sendKeys("9790917775");

driver.findElement(By.*name*("pass")).sendKeys("XXXYYY");

driver.findElement(By.*linkText*("Forgotten account?")).click();

}

}

* ID
* CLASSNAME
* NAME
* LINKTEXT
* XPATH
* CSS

Every object may not have ID, NAME

Alphanumeric IDs will change in every refresh, so don’t use them.

* Confirm the link object anchor a tag.

When we have a class name with spaces, selenium will not read it. So it should be with without spaces. Error: **invalid selector: Compound class names not permitted**

**One complete login attempt:-**

**package** basics;

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

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

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

**public class** Locator\_Salesforce {

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

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

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

driver.get("https://login.salesforce.com/"); //URL OF THE BROWSER

driver.findElement(By.*id*("username")).sendKeys("9790917775");

driver.findElement(By.*name*("pw")).sendKeys("XXXYYY");

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

System.***out***.println("Login attempt completed successfully !!!");

}

}

Selenium scans from top left -> If there is a field id which is same for both , then selenium will scan from top left , and the first element will get the locator and then next one will be skipped.

Multiple values – Selenium identifies the first one.

driver.findElement(By.*className*("inputtext")).sendKeys("9790917775");

driver.findElement(By.*className*("inputtext")).sendKeys("XXXYYY");

In the above case both username and password fields have same className, but it only fills the first field i.e. username. One more point here, if we are sending different data for the same field twice then those gets appended.

***Break: Just a note to self to boost the motivation.***

* *Spaced Strategy - Not Marathon study – consistent short study session*
* *Don’t spend entire session on one topic – one topic for long time*
* *Have notes and ask why and how to yourself and note them down – organize your own ideas and note it down*
* *Use specific concrete examples – Real life examples*
* *Combine verbal materials with visuals*
* *Make it as simple as you can, for yourself and for others whoever you will share this knowledge. Plain and simple way is the way of genius.*

**Identifying xpath& CSS in chrome & Firefox browser:-**

* We can generate Xpath from browser tools.
* We can generate Xpath by ourselves.
* Xpath can be defined in n number of ways
* Right click copy on blue highlighted html code to generate xpath
* Double quotes inside double quotes are not accepted. Normally xpaths will be in double quotes e.g. //\*[@id="email"]. So we need to change it to - //\*[@id=’email’] while using in our selenium code. Or else it’ll throw error.
* We can write xpath in n number of ways. There is specific way of writing it. Each browser may handle it differently.

**package** basics;

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

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

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

**public class** Locator\_Salesforce {

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

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

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

driver.get("https://customer.i-on.in/"); //URL OF THE BROWSER

driver.manage().window().maximize(); //Maximizes the window

driver.findElement(By.*cssSelector*("#user")).sendKeys("GION54074"); //By CSSSELECTOR

driver.findElement(By.*xpath*("//\*[@id='password']")).sendKeys("GION54074"); // BY Xpath

driver.findElement(By.*xpath*("//\*[@id='login']/input")).click(); //BY XPATH

System.***out***.println("Login attempt completed successfully !!!");

}

}

* If a xpath starts with html then we should not use that, it’s not reliable. Check in both chrome and Firefox while taking, check which one is correct.
* There is no direct way to get css chrome. You will find it in tool bar.

**Validating the accuracy of xpath and Css from browser add-ons: -**

* Right click > inspect element > get the **XPATH** e.g. - //\*[@id="forgot\_password\_link"]
* Go to the console tab> type $x("//\*[@id='forgot\_password\_link']")
* It will give below output > then the xpath is correct

>**$x("//\*[@id='forgot\_password\_link']")**

*[a#forgot\_password\_link.fl.small]*

0:a#forgot\_password\_link.fl.small

length:1

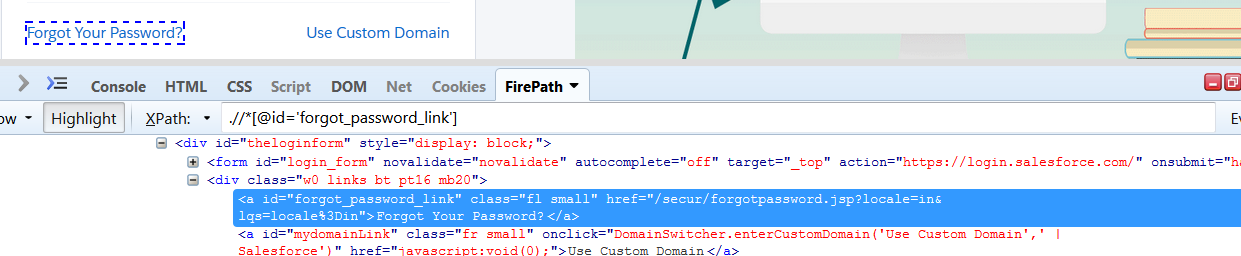
\_\_proto\_\_:Array(0)

* Right click > inspect element > get the **CSS** e.g. - #mydomainLink
* Go to the console tab > type $("#mydomainLink")

>**$("#mydomainLink")**

<a id=​"mydomainLink" class=​"fr small" href=​"javascript:​void(0)​;​" onclick=​"DomainSwitcher.enterCustomDomain('Use Custom Domain',' | Salesforce')​">​Use Custom Domain​</a>​

* In Firepath there is another method of verifying the xpath i.e. Firefox >FirePath tab > Put in the top section and check if it is selecting one. See below screenshot :



**package** basics;

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

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

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

**public class** Locator\_Salesforce {

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

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

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

driver.get("https://login.salesforce.com/"); //URL OF THE BROWSER

driver.manage().window().maximize(); //Maximizes the window

driver.findElement(By.*cssSelector*("#username")).sendKeys("GION54074"); //By CSSSELECTOR

driver.findElement(By.*xpath*("//\*[@id='password']")).sendKeys("GION54074"); // BY Xpath

driver.findElement(By.*xpath*("//\*[@id='Login']")).click(); //BY XPATH

System.***out***.println("Printing the error message below - ");

System.***out***.println(driver.findElement(By.*cssSelector*("#error")).getText());

System.***out***.println("Login attempt completed successfully!!!");

}

}

O/P: -

Dec 18, 2017 7:46:15 AM org.openqa.selenium.remote.ProtocolHandshakecreateSession

INFO: Detected dialect: OSS

Printing the error message below -

Please check your username and password. If you still can't log in, contact your Salesforce administrator.

Login attempt completed successfully!!!

**Generating customized Xpath from html attributes:-**

**//tagname[@attribute=’value’]** – **Customized Xpath**

Exact html link facebooklogin :<input value="Log In" tabindex="4" data-testid="royal\_login\_button" type="submit" id="u\_0\_2">

So the customized Xpath is - //input[@value='Log In']

Example **of logging in facebook account and getting profile name**:-

**package** basics;

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

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

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

**public class** FacebookLogin {

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

//URL OF THE BROWSER

driver.manage().window().maximize(); //Maximizes the window

driver.findElement(By.*cssSelector*("#email")).sendKeys("9790917775"); //By CSSSELECTOR

driver.findElement(By.*xpath*("//\*[@id='pass']")).sendKeys("XXXXXXX"); // BY Xpath

driver.findElement(By.*xpath***("//input[@value='Log In']")).**click(); //**BY Customized XPATH**

System.***out***.println("Printing the profilename below - ");

System.***out***.println(driver.findElement(By.*cssSelector*("#navItem\_100001318351560 > a > div")).getText());

System.***out***.println("Login attempt completed successfully !!!");

}

}

O/P:-

Dec 18, 2017 8:12:06 AM org.openqa.selenium.remote.ProtocolHandshakecreateSession

INFO: Detected dialect: OSS

Printing the profilename below -

Chittaranjan Swain

Login attempt completed successfully !!!

If you think tag name doesn’t remain same, it keeps on changing then, you can give the xpath as below :

**$x("//\*[@type='email']")- \* can be given in the place of tag name**

**Logging in Facebook only using customized XPATH:-**

**package** basics;

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

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

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

**public class** FacebookXpath {

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

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

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

driver.get("https://www.facebook.com");

driver.manage().window().maximize(); //Maximizes the window

driver.findElement(By.*xpath*("//input[@type='email']")).sendKeys("9790917775");

driver.findElement(By.*xpath*("//input[@type='password']")).sendKeys("XXXXXX");

driver.findElement(By.*xpath*("//input[@value='Log In']")).click();

}

}

**Generating customized CSS from html attributes:-**

**tagName[attribute=’value’]** - **Customized CSS**

**Logging in Facebook only using customized CSS selector:-**

**package** basics;

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

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

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

**public class** FacebookXpath {

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

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

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

driver.get("https://www.facebook.com");

driver.manage().window().maximize(); //Maximizes the window

driver.findElement(By.*cssSelector*("input[name='email']")).sendKeys("9790917775");

driver.findElement(By.*cssSelector*("input[id='pass']")).sendKeys("XXXXXX");

driver.findElement(By.*cssSelector*("input[value='Log In']")).click();

}

}

**Few more ways to write CSS:-**

**tagName#id** – CSS

input#pass – To recognize password field in facebook now. So below is the syntax to give password

driver.findElement(By.cssSelector("**input#pass**")).sendKeys("XXXXXXX");

**#id** - CSS

#pass - To recognize password field in facebook now. So below is the syntax to give password

driver.findElement(By.cssSelector("**#pass**")).sendKeys("XXXXXXX");

**tagName.className**– CSS

input.inputtext - To recognize Username field in facebook now. So below is the syntax to give password

driver.findElement(By.cssSelector("**input.inputtext**")).sendKeys("XXXXXXX");

**Regular Expression:-**

* Suppose a tag is alphanumeric. Means a part of the tag is static and the other part of the tag is dynamic, then how to handle this

<input name=”username123”> - Example is given below

**//tagName[contains(@attribute,’value’)]**– **XPATH REGULAR EXPRESSION SYNTAX**

e.g. //input[contains(@name,’username’)]

**tagName[attribute\*=’value’]** – **CSS REGULAR EXPRESSION SYNTAX**

e.g. input[name\*=username]

* This above trick can be used when the tag names are very lengthy.

**E2E login of reddif website using whatever we learned yet :-**

**package** basics;

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

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

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

**public class** Rediff\_Regular\_Exp {

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

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

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

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

driver.findElement(By.*cssSelector*("a[title\*='Sign in']")).click(); //searches for 'Sign in' keyword

// & finds the title which matches it first.

driver.findElement(By.*xpath*("//input[@id='login1']")).sendKeys("Hello");

//driver.findElement(By.xpath("//input[@id='password']")).sendKeys("Hello");// filling password field using Xpath

driver.findElement(By.*cssSelector*("#password")).sendKeys("Hello"); // filling password field using CSS

//it can be input#passowrd (tagname#id) or #password (#id)

driver.findElement(By.*xpath*("//input[contains(@name,'procee')]")).click();//Using regular expression in xpath

}

}

***Cheat Sheet Till now:-***

|  |
| --- |
| * Every object may not have ID, className or name- Xpath and CSS Preferred |
| * Alpha numeric id may vary on every refresh- check |
| * Confirm the link object with anchor "a" tag |
| * Classes should not have spaces- Compound classes cannot be accepted |
| * Multiple values - Selenium identifies the first one- Scans from top left |
| * Double quotes inside double quotes are not accepted |
| * Xpath/CSS can be defined in n number of ways |
| * Rightclick copy on blue highlighted html code to generate xpath |
| * Firepath depreciated from firefox- |
| * when xpath starts with html-Not reliable- Switch browser to get another one |
| * There is no direct way to get CSS in chrome. You will find it in tool bar |
| * Degrade browser to less firefox 55 to geFirepath |
| * $("") - for css , $x("") or xpath |
| * //tagName[@attribute='value'] -xpath syntax |
| * tagName[attribute='value'] -CSS tagName#id- CSS tagname.classname- CSS |
| * //tagName[contains(@attribute,'value')] -xpath regular expressions |
| * tagName[Atrribute\*='value'] - Css regular expression   **Identify XPATH with Parent child Traverse relation:-**   * When you don’t have any static locators * Define XPATH for the parent   //div[@class='row']/div/div/div/h3 – This is for the XPATH for the data used for I-ON webpage  $x("//div[@class='col-md-3']/div[1]/div[1]/h3[0]")  Google search box xpath :  //div[@class='lst-c']/div/div[2]/div/input  **package** basics;  **import org**.openqa.selenium.By;  **import org**.openqa.selenium.WebDriver;  **import org**.openqa.selenium.chrome.ChromeDriver;  **public class** Google\_Parent\_Child\_Xpath {  **public static void** main(String[] args) {  System.*setProperty*("webdriver.chrome.driver", "E:\\SOFTWARE\\chromedriver.exe");  WebDriver driver = **new ChromeDriver**();  driver.get("https://www.google.co.in/");  driver.findElement(By.*xpath*("//div[@class='gb\_jegb\_Rgb\_Dggb\_ug']/div[2]/a")).click();  //Above one will search image tab and gets the xpath from parent to child  driver.findElement(By.*xpath*("//div[@class='lst-c']/div/div[2]/div/input")).sendKeys("Chittaranjan");  //Above one will get the xpath for search box using parent to child traversal  //driver.findElement(By.xpath("//div[@class='sbibod']/button/span/svg")).click();  }  } |
| **Chropath – chrome**  **1. What is the difference between relative & absolute xpath?**   * If we are jumping to the tag directly without taking its parent into consideration * When we find a tag directly without depending upon its parent then the xpath is called as **relative xpath.**   e.g. //\*[@id='tablist1-tab2'] - selenium tab from http://www.qaclickacademy.com/interview.php  Relative – Doesn’t depend on parent node.  When we traverse desired tag from its parent and the total xpath is known as **Absolute Xpath**  e.g. //section/div/div/div/div/ul/li[2] - selenium tab from <http://www.qaclickacademy.com/interview.php>  Absolute – Depends on parent child relation. Parent/child   * Relative is preferred because absolute has more chance of getting modified by developers and one modification can impact the entire xpath   **How to move from one sibling to another sibling?**  **How to identify siblings and traverse back to parent?**  **.//\*[@id='tablist1-tab1']/following-sibling::li[1]** – Copy &paste this in Xpath section of firepath (firefox) and check , how it works.  **.//\*[@id='tablist1-tab2']/preceding-sibling::li[1]** – It will go the previous sibling from the current one.  **package** basics;  **import org**.openqa.selenium.By;  **import org**.openqa.selenium.WebDriver;  **import org**.openqa.selenium.chrome.ChromeDriver;  **public class** FollowingSibling\_QAcademy {  **public static void** main(String[] args) {  System.*setProperty*("webdriver.chrome.driver", "E:\\SOFTWARE\\chromedriver.exe");  WebDriver driver = **new ChromeDriver**();  driver.get("http://www.qaclickacademy.com"); //URL OF THE Webpage  driver.manage().window().maximize(); //Maximizes the window  driver.findElement(By.*xpath*("//\*[@id=\"homepage\"]/header/div[2]/div/nav/ul/li[4]/a")).click();  driver.findElement(By.*xpath*("//\*[@id='tablist1-tab1']")).click();  driver.findElement(By.*xpath*("//\*[@id='tablist1-tab1']/following-sibling::li[1]")).click();  driver.findElement(By.*xpath*("//\*[@id='tablist1-tab1']/following-sibling::li[2]")).click();  driver.findElement(By.*xpath*("//\*[@id='tablist1-tab1']/following-sibling::li[3]")).click();  }  }  **2. Using Child element can you traverse back to parent?**  **//\*[@id=’tablist1-tab1’]/parent::ul** - **This happens only in xapth not in CSS , this is a major difference between xpath and css**  In CSS, you can’t comeback from a child to parent, but in xpath you can do that. |
| **3. How to identify a web element using only text?** |

**//\*[text()=' Selenium ']** –Selecting selenium web-element from <http://www.qaclickacademy.com/interview.php>

**package** basics;

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

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

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

**public class** LocatorPractice {

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

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

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

driver.get("http://www.qaclickacademy.com/interview.php");

driver.findElement(By.*xpath*("//li[text()=' Selenium ']")).click();

driver.findElement(By.*xpath*("//ul[@class='responsive-tabs\_\_list']/li[1]/following-sibling::li[2]")).click();

System.***out***.println(driver.findElement(By.*xpath*(".//\*[@id='tablist1-tab2']/parent::ul")).getAttribute("role"));

}

}

How to trigger selenium script from Jenkin

What is a listener class?

**Let’s Revise the most Important thing, XPATH & CSS:-**

**CSS Selector: -**

it **is 10 times faster** than xpath.

Xpath syntax is - //tagname[@attribute=’value’]

Csssystanx is - tagname[attribute=’valye’]

e.g. //div[@class=’identity first’]

div[class=’’identity first’]

* In xpath, if we are not interested to give the tagname then we can use \*, but in css, no need to mention anything. Leave it blank , like below

e.g. //\*[@class='input r4 wide mb16 mt8 username'] – In XPATH

[class='input r4 wide mb16 mt8 username'] – In CSS

* When there is class name with only one word, without any space then you can create the customized css for that also -> .class\_name(.mb16) like this.

#id, .class\_name\_without\_class\_name also works.

* If there is space in the class name and you want to use with dot, then you remove all the spaces and mention dot there.

Like - .input r4 wide mb16 mt8 username -> will throwerror but ->.input.r4.wide.mb16.mt8.username will not.

**package** basics;

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

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

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

**public class** Css {

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

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

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

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

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

driver.findElement(By.*cssSelector*("[class='input r4 wide mb16 mt8 username']")).sendKeys("Hello");

//without tagname

driver.findElement(By.*cssSelector*("input[id='password']")).sendKeys("Hello"); // with tagname

//driver.findElement(By.cssSelector("#Login")).click(); // using # before ID

driver.findElement(By.*cssSelector*(".button.r4.wide.primary")).click();

//Using dot in class where multiple words are there

}

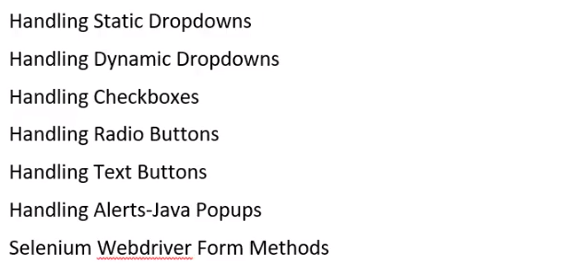
}

**So the important are:**

1. Verify in firebug , before you run the script
2. CSS customized syntax
3. Short cuts :- #id & .class\_name
4. Try to find out the unique attribute which will find the web elements

**SELENIUM – PART – 2**

**Selenium Web driver - Techniques to handle various web elements: -**

****

[**http://www.spicejet.com/**](http://www.spicejet.com/) **- This link has all the web elements that we will learn**

**Static dropdown: -The** dropdown list which is pre-loaded.

**package** basics;

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

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

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

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

**import org**.openqa.selenium.support.ui.Select;

**public class**  Dropdown {

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

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

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

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

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

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

driver.findElement(By.*name*("userName")).sendKeys("ch245510");

driver.findElement(By.*name*("password")).sendKeys("ch245510");

driver.findElement(By.*name*("login")).click();

//Now to select a dropdown item, there is a class called select

Select s = **new** Select(driver.findElement(By.*name*("passCount")));

s.selectByValue("2");//open all the child tags of ID , select whatever you want

s.selectByIndex(3);//index starts from 0.

s.selectByVisibleText("1");//the space that follows the visible text will not be considered

//We can use above functions only when there is select tag

}

}

**Dynamic dropdown& Check Boxes: -**

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

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

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

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

**import org**.openqa.selenium.support.ui.Select;

**public class** Dynamic\_DropDown {

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

System.*setProperty*("webdriver.chrome.driver", "D:\\Everything\_on\_Selenium\\chromedriver\_win32\\chromedriver.exe");

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

driver.get("http://spicejet.com/");

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

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

Select s = **new** Select(driver.findElement(By.*id*("ctl00\_mainContent\_ddl\_Adult")));

s.selectByValue("2");//open all the child tags of ID , select whatever you want

s.selectByIndex(3);//index starts from 0.

s.selectByVisibleText("1");

//the space that follows the visible text will not be considered

//We can use above functions only when there is select tag

//Now we need to select from location

driver.findElement(By.*cssSelector*("#ctl00\_mainContent\_ddl\_originStation1\_CTXT")).click();

//a[@value='GOI']

driver.findElement(By.*xpath*("//a[@value='GOI']")).click(); //FROM

driver.findElement(By.*xpath*("(//a[@value='DEL'])[2]")).click(); //TO

//Both To & from are having same code here in the html

//(//a[@value='DEL'])[2] - This will go the 2nd instance of the same XPATH

//(Most Important)

System.***out***.println(driver.findElement(By.*id*("ctl00\_mainContent\_chk\_IndArm")).isSelected());

//isSelected(); will be used to check if it already selected or not

//Clicking checkbox below

driver.findElement(By.*id*("ctl00\_mainContent\_chk\_IndArm")).click();

//Now checking whether checkbox is selected or not

System.***out***.println(driver.findElement(By.*id*("ctl00\_mainContent\_chk\_IndArm")).isSelected());

}

}

o/p: -

Starting ChromeDriver 2.34.522940 (1a76f96f66e3ca7b8e57d503b4dd3bccfba87af1) on port 1433

Only local connections are allowed.

Jan 03, 2018 10:41:31 AM org.openqa.selenium.remote.ProtocolHandshakecreateSession

INFO: Detected dialect: OSS

false

true

* One type of error is, element is not clickable or not

**Radio Buttons: -**

<http://echoecho.com/htmlforms.htm> - This webpage will give, web related elements for practice

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

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

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

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

**public class** RadioButtons {

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

System.*setProperty*("webdriver.chrome.driver", "D:\\Everything\_on\_Selenium\\chromedriver\_win32\\chromedriver.exe");

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

driver.get("http://echoecho.com/htmlforms10.htm");

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

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

driver.findElement(By.*xpath*("//input[@value='Cheese']")).click();

//Below code will count the number of radio buttons

**int count** = driver.findElements(By.*xpath*("//input[@name='group1']")).size();

//Now without value attribute still, I want to click on that radio button

**for**(**int i=**0;i<count;i++)

{

//driver.findElements(By.xpath("//input[@name='group1']")).get(i).click();

//get will give us which element has to be clicked.

//Now it will click on each item one by one.

//Now clicking on the item when we find "cheese".

String s=driver.findElements(By.*xpath*("//input[@name='group1']")).get(i).getAttribute("value");

**if**(s=="cheese")

{

driver.findElements(By.*xpath*("//input[@name='group1']")).get(i).click();

}

}

}

}

**Java Alerts: -**

<http://www.tizag.com/javascriptT/javascriptalert.php> **-** Click on “Confirmation Alert”

There are some pop-ups for which there will not be any html tags, those JAVA pop up. (like above link) – Java related POP-UP/Alerts

There are some pup-ups, for which we can spy, those html code. - click “One Way” in <https://www.goindigo.in/> . – WEB/HTML related POP-UP

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

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

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

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

**public class**  Alerts {

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

System.*setProperty*("webdriver.chrome.driver", "D:\\Everything\_on\_Selenium\\chromedriver\_win32\\chromedriver.exe");

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

driver.get("http://www.tizag.com/javascriptT/javascriptalert.php");

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

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

//Click on confirmation alert

driver.findElement(By.*xpath*("//input[@value='Confirmation Alert']")).click();

//Now Java POP-UP will open , to which we will not be able spy.

//Now we have to change our driver web element to Alert

System.***out***.println(driver.switchTo().alert().getText());//logic is written later

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

//If you have to click on OK/YES/done or any positive kind of scenario then

//you need to use accept();

//If we want to click on any negative options, then we need to use below

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

//if we want to verify the text that getting displayed is correct or not

//driver.switchTo().alert().getText();//This we need to use it before accept();

//Suppose we want to give input inside popup, then we need to use below

//driver.switchTo().alert().sendKeys(arg0);

}

}

Starting ChromeDriver 2.34.522940 (1a76f96f66e3ca7b8e57d503b4dd3bccfba87af1) on port 13547

Only local connections are allowed.

Jan 05, 2018 9:52:02 AM org.openqa.selenium.remote.ProtocolHandshakecreateSession

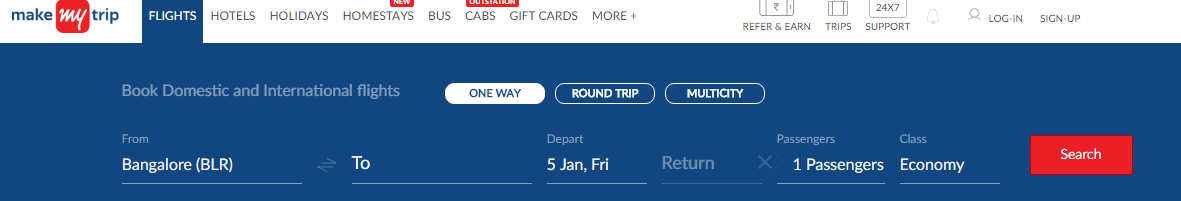
INFO: Detected dialect: OSS

Are you sure you want to give us the deed to your house?

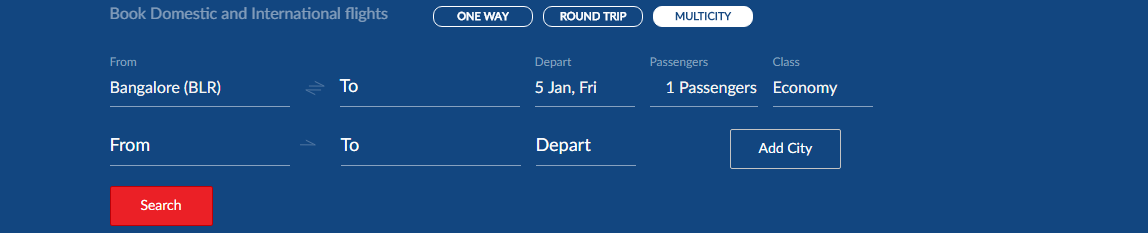
**Web Driver Methods: - Web Element Validation: -**

<https://www.makemytrip.com/>

In one way, return option is there but disabled



But in Multicity, return option is not there at all.



That means something is in hidden mode. So how do we check that the UI element is visible or hidden?

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

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

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

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

**public class** Webdriver\_Methods {

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

System.*setProperty*("webdriver.chrome.driver", "D:\\Everything\_on\_Selenium\\chromedriver\_win32\\chromedriver.exe");

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

driver.get("https://www.makemytrip.com/");

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

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

System.***out***.println(" Before clicking on multi-city radio button ");

System.***out***.println(driver.findElement(By.*xpath*("//input[@id='hp-widget\_\_return']")).isDisplayed());

//isDisplayed() will check , if the UI element is in visible mode or not.

driver.findElement(By.*xpath*(".//\*[@id='multicity']/label")).click();

//taking the xpath from firepath in the above case

System.***out***.println(" After clicking on multi-city radio button ");

System.***out***.println(driver.findElement(By.*xpath*("//input[@id='hp-widget\_\_return']")).isDisplayed());

}

}

Before clicking on multi-city radio button

true

After clicking on multi-city radio button

false

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

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

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

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

**public class** Webdriver\_Methods {

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

System.*setProperty*("webdriver.chrome.driver", "D:\\Everything\_on\_Selenium\\chromedriver\_win32\\chromedriver.exe");

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

driver.get("https://www.makemytrip.com/");

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

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

System.***out***.println(" Before clicking on multi-city radio button ");

System.***out***.println(driver.findElement(By.*xpath*("//input[@id='hp-widget\_\_return']")).isDisplayed());

//isDisplayed() will check , if the UI element is in visible mode or not.

driver.findElement(By.*xpath*(".//\*[@id='multicity']/label")).click();

//taking the xpath from firepath

System.***out***.println(" After clicking on multi-city radio button ");

System.***out***.println(driver.findElement(By.*xpath*("//input[@id='hp-widget\_\_return']")).isDisplayed());

**int count** = driver.findElements(By.*xpath*(".//\*[@id='multicity']/label")).size();

//Only driver.findElements will have size() method , driver.findElement will not have

**if**(count==1)

{

System.***out***.println("Web element present though it is not visible!!!");

//this is the difference between .size() and .isdisplayed()

//size will check if the element is present or not or how many element present

//But isdisplayed() will give if the elements are present but not visible

}

Thread.*sleep*(3000L);

//Now we learn calender

driver.findElement(By.*xpath*(".//\*[@id='hp-widget\_\_depart']")).click();

//Below is the way to click on a calender date when the date is given in A HREF value

//in this case xpath&css may not work , use linkText

driver.findElement(By.*linkText*("20")).click();

//Below it will choose 6 passengers

driver.findElement(By.*xpath*(".//\*[@id='hp-widget\_\_paxCounter']")).click();

driver.findElement(By.*xpath*(".//\*[@id='js-adult\_counter']/li[6]")).click();

driver.findElement(By.*xpath*(".//\*[@id='jsfilterOptins']/div/div[9]/div[1]/a")).click();

//driver.findElement(By.xpath(".//\*[@id='dp1515127469077']/div/div[1]/table/tbody/tr[4]/td[7]")).click();

//if you want to validate object which is prespent in web page or code base

//then we have to use .size()

//Now we will discuss on the topic isenabled()

//Now we will discuss on getText()

System.***out***.println(driver.findElement(By.*xpath*(".//\*[@id='fd-wrap']/div[2]/h2")).getText());

//it will o/p as Discover Hot Deals

}

}

Before clicking on multi-city radio button

true

After clicking on multi-city radio button

false

Web element present though it is not visible!!!

Discover Hot Deals

**Assertion: -**Asserts helps us to verify the conditions of the test and decide whether test has failed or passed. A test is considered successful ONLY if it is completed without throwing any exception.

Here we are verifying if the page title is equal to 'Google' or not. If the page title is not matching with the text / title that we provided, it will fail the test case.

**publicvoidtestCaseVerifyHomePage**() {

driver= **new FirefoxDriver**();

driver.navigate().to("[http://google.com"](http://google.com/));

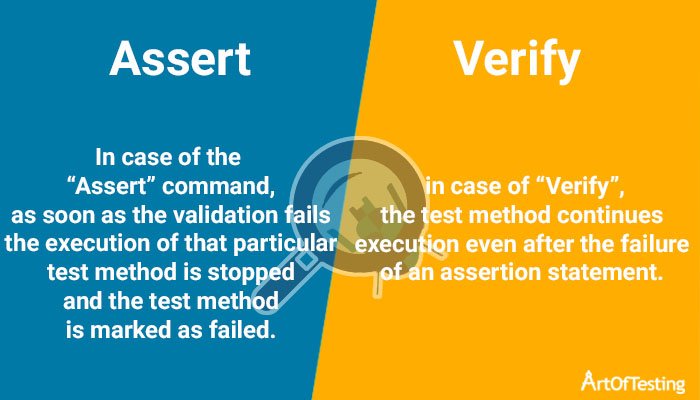
Assert.assertEquals("Google", driver.getTitle());

}

**Assert Vs Verify Commands in Selenium**

The validation checks made in Selenium usually come in two flavors: one using Assertions and the other inculcating “Verify” statements. Though both serve the same purpose there persists one major difference in their working functionality.

Verify checks get test results for multiple conditions even if one of them fails wherein Hard Assertions put a stringent restriction on the test script when it fails thereby terminating the program execution further. Although Soft Assertions work in a way different from Hard Assertion where normal flow of execution resumes although there is failure in the asserting script. Usage of either Assertions or verify statement purely falls on the user’s cup of tea.



# Commonly used Assertions

## Assert Equals

Assert Equals works by comparing the Expected condition with that of the Actual condition based on which the test results are displayed. The following example illustrates the usage of assert equals as a validation check to verify the Title of the homepage displayed for the mentioned site.

[](https://i1.wp.com/blogs.perficient.com/delivery/files/2017/01/Log-In-page-1.jpg?ssl=1)

***public void testCaseVerifyTitlePage() {***

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

***driver.navigate().to(“https://wiki.perficient.com/ /confluence/login.action?os\_destination=%2Fhomepage.action “);***

***Assert.assertEquals(“Log In – Perficient Wiki “, driver.getTitle());***

***}***

## Assert Not Equals

“Assert Not Equals” serves the purpose of negation testing for Testers. If the Actual and Expected do not match, then the test Script passes else it fails. The following example demonstrates how a user inculcates “Assert Not Equals” for verifying the display of text that dynamically changes as per date.

**public void testCaseVerifyDate () {**

**Actualtext = driver.findElement(By.xpath(“//h3/span”)). getText();**

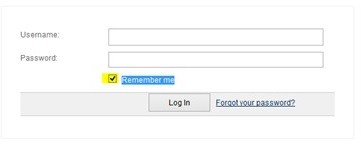
**Assert.assertNotEquals(Actualtext, “Here comes the surprise gift for every order on Sunday, 01 January 2017”, “Expected and Actual do not match as the test is performed on December 31st,2016″);**

**}**

This Test Case will pass as the above Scenario is tested on December 31st, 2016.

## Assert True

This Assertion passes the Test Step only when the boolean value returned is “True”. The following example illustrates that the user validates the default checkbox is checked in the specified site.

[](https://i1.wp.com/blogs.perficient.com/delivery/files/2017/01/Assert_True.jpg?ssl=1)

***public void testDefaultCheckbox() {***

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

***driver.get(“https://wiki.perficient.com/ /confluence/login.action?os\_destination=%2Fhomepage.action “);***

***defaultchk1 = driver.findElement(By.name(“os\_cookie”));***

***System.out.print(“\n”+ defaultchk1.isSelected());***

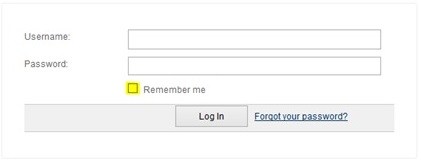
***Assert.assertTrue(defaultchk1.isSelected());***

***System.out.print(“Assertion Passed successfully “);***

***}***

## Assert False

“Assert False” passes the Test Step only when the boolean value returned is “False”. The following example illustrates that the user validates the optional checkbox is unchecked in the specified site.

[](https://i0.wp.com/blogs.perficient.com/delivery/files/2017/01/Assert-False.jpg?ssl=1)

**public void testOptionalCheckbox() {**

**driver= new FirefoxDriver();**

**driver.get(“https://wiki.perficient.com/ /confluence/login.action?os\_destination=%2Fhomepage.action “);**

**optionalchk = driver.findElement(By.name(“os\_cookie”));**

**System.out.print(“\n”+optionalchk.isSelected());**

**Assert.assertFalse(optionalchk.isSelected());**

**System.out.print(“Assertion Passed successfully “);**

**}**

## Assert Null

This Assertion verifies if the object under test is null, and the passes the same if the result is so.Let’s consider an element that is supposed to be enabled in the application and the user verifies the same using Assert Null statement.

***public void testEnabledText() {***

***driver.get(“https://wiki.perficient.com/ /confluence/login.action?os\_destination=%2Fhomepage.action “);***

***enatxt = driver.findElement(By.xpath(“//input[@id=’text2′]”));***

***Assert.assertNull(enatxt.getAttribute(“disabled”));***

***System.out.print(“Assertion Passed successfully “);***

***}***

In above example, enatxt.getAttribute(“disabled”) object will return “null” because the “enatxt” element is not a disabled attribute. So that assertion will pass.

**Assert Not Null: -**This Assertion functions opposite to that of “Assert Null”. Thus, this Assertion verifies if the object under test is not null, and the passes the same if the result is so. Let’s consider an element that is supposed to be disabled in the application and the user verifies the same using Assert Not Null statement.

**public void testDisabledText() {**

**driver.get(“https://wiki.perficient.com/ /confluence/login.action?os\_destination=%2Fhomepage.action “);**

**distxt = driver.findElement(By.xpath(“//input[@id=’text21′]”));**

**Assert.assertNull(distxt.getAttribute(“disabled”));**

**System.out.print(“Assertion Passed successfully “);**

**}**

## Assert Same

This Assertion checks that two objects refer to the same object, if it does, then the Assertion passes else it fails the same. The below example illustrates the usage of “Assert.assert same” for testing two objects that refer the same object:

***Public class AssertSameTest\_Demo {***

***public String getTextValue(final String key){***

***Map<string, string=””>textProps = new HashMap<string, string=””>();***

***textProps.put(“text1”, “value 1”);***

***textProps.put(“text2”, “value 2”);***

***textProps.put(“text3”, “value 3”);***

***return textProps.get(key);***

***}***

***@Test***

***public void isSameTest(){***

***AssertSameTest\_Demoabc = new AssertSameTest\_Demo ();***

***assertSame(abc. getTextValue (“text1”), abc. getTextValue (“text1”));***

***}***

## Assert Not Same

This Assertion checks that two objects do not refer to the same object, if it does than the Assertion passes else it fails the same. The below example illustrates the usage of “Assert.assert Not same” for testing two objects that refer two different objects.

**Public class AssertNotSameTest\_Demo {**

**public String getTextValue(final String key){**

**Map<string, string=””>textProps = new HashMap<string, string=””>();**

**textProps.put(“text1”, “value 1”);**

**textProps.put(“text2”, “value 2”);**

**textProps.put(“text3”, “value 3”);**

**return textProps.get(key);**

**}**

**@Test**

**public void isNotSameTest() {**

**AssertNotSameTest\_Demoxyz = new AssertNotSameTest\_Demo ();**

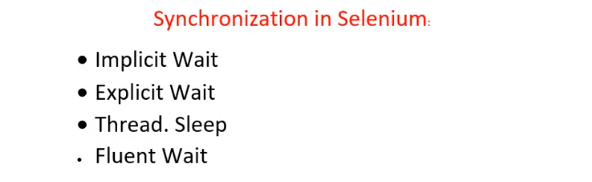
**assertNotSame(xyz. getTextValue (“text1”), xyz. getTextValue (“text3”));**

**}**

**Interview Question: -**

* How do you handle dropdowns? – By using select class
* How do you handle java alerts? – By using driver.switchTo().alert(). – for OK use accept(), for cancel use dismiss().
* How do you handle checked boxes? – Usual way but isSelected()- To make sure check box is checked or not
* How to select a radio button, when there is 3 radio buttons and all have same attribute – Use findElements(); to get all the list of the attributes , and then I can traverse to get the particular radio button. This is already being discussed earlier.

isDisabled() Method is removed from Selenium 3.0 version.



Example: You provided your criteria for searching the flight and clicked on search button. Based on your search results it will keep loading for few seconds and then it will show results. Then you clicked n first flight results and then it will again load for few seconds.

Then the first search will be loading and the 2nd search would have started again loading. How can we tell our script to wait for loading of the ‘1stsearch’ completely and then to click on the 2nd search?

**Implicit wait:** Wait for n seconds before you throw exceptions.

If explicitly we declare globally that for every loading we need to wait for n seconds, then for entire script for each search result loading it will wait for ‘n’ seconds.

‘n’ is the maximum timeout limit, but it will not wait every time, if the loading of one page is completed then it can go to the next item in the script. Maximum wait time is ‘n’ seconds, if it’s still not loading then it will fail.

Disadvantages of implicit wait - If there are more results then it will take more time to load. Then in that case if you are increasing the implicit wait value then it works fine. But after some time of other loading items will also take more time, then we will not be able to identify it. So, we will miss performance issues. Test executions will drastically slowdown.

**Explicit wait:** So to avoid above problem, we will use explicit wait. For specific cases, we can use explicit wait.

**Thread. Sleep:**

Example of implicit wait: -

Q. Find out the cheapest hotel in GOA from 20th Jan to 21st Jan for 3 adults.

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

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

**import org**.openqa.selenium.Keys;

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

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

**Public class**  Synchronize {

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

System.*setProperty*("webdriver.chrome.driver", "D:\\Everything\_on\_Selenium\\chromedriver\_win32\\chromedriver.exe");

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

driver.get("https://www.makemytrip.com/");

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

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

driver.findElement(By.*id*("header\_tab\_hotels")).click();

driver.findElement(By.*id*("hp-widget\_\_sDest")).click();

driver.findElement(By.*id*("hp-widget\_\_sDest")).sendKeys("GOA");

driver.findElement(By.*id*("hp-widget\_\_chkIn")).click();

driver.findElement(By.*linkText*("20")).click();

driver.findElement(By.*linkText*("21")).click();

driver.findElement(By.*id*("hp-widget\_\_paxCounter")).click();

driver.findElement(By.*xpath*(".//\*[@id='js-adult\_counter']/li[3]")).click();

driver.findElement(By.*linkText*("Done")).click();

//driver.findElement(By.className("close\_pax pull-right")).sendKeys(Keys.ENTER);

//above it is mentioned, how to hit enter , but it's not useful here.

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

//Below will click on the first hotel of the search results

driver.findElement(By.*xpath*(".//\*[@id='listingSortDropDown']/button")).click();

driver.findElement(By.*id*("2")).click();

driver.findElement(By.*id*("js-hotelName-0")).click();

}

**}**

**Now writing the same program with Explicit wait: -**

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

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

**import org**.openqa.selenium.Keys;

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

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

**import** org.openqa.selenium.support.ui.ExpectedConditions;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**Public class**  Synchronize {

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

System.*setProperty*("webdriver.chrome.driver", "D:\\Everything\_on\_Selenium\\chromedriver\_win32\\chromedriver.exe");

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

driver.get("https://www.makemytrip.com/");

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

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

driver.findElement(By.*cssSelector*("#ch\_login\_icon>span")).click();

driver.findElement(By.*cssSelector*("#ch\_login\_email")).sendKeys("emailtochitta@gmail.com");

driver.findElement(By.*cssSelector*("#ch\_login\_password")).sendKeys("XXXXXXXXX");

driver.findElement(By.*cssSelector*("#ch\_login\_btn")).click();

driver.findElement(By.*id*("header\_tab\_hotels")).click();

driver.findElement(By.*id*("hp-widget\_\_sDest")).click();

driver.findElement(By.*id*("hp-widget\_\_sDest")).sendKeys("GOA");

driver.findElement(By.*id*("hp-widget\_\_chkIn")).click();

driver.findElement(By.*linkText*("20")).click();

driver.findElement(By.*linkText*("21")).click();

driver.findElement(By.*id*("hp-widget\_\_paxCounter")).click();

driver.findElement(By.*xpath*(".//\*[@id='js-adult\_counter']/li[3]")).click();

driver.findElement(By.*linkText*("Done")).click();

//driver.findElement(By.className("close\_pax pull-right")).sendKeys(Keys.ENTER);

//above it is mentioned, how to hit enter , but it's not useful here.

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

//Now using Explicit wait which is applicable to only next load not for all load

WebDriverWait d = **new** WebDriverWait(driver,20);

//We can keep implicit wait and explicit wait for both together, in that case

//>implicit wait time will be applicable for all but explicit wait timing will be applicable loads controlled by "d" object , see below

//Below will click on the first hotel of the search results

d.until(ExpectedConditions.*visibilityOfElementLocated*(By.*xpath*(".//\*[@id='listingSortDropDown']/button")));

//there are several functionality for Expected Conditions

driver.findElement(By.*xpath*(".//\*[@id='listingSortDropDown']/button")).click();

driver.findElement(By.*id*("2")).click();

//You need to apply the explicit wait before each loading , like below

d.until(ExpectedConditions.*visibilityOfElementLocated*(By.*id*("js-hotelName-0")));

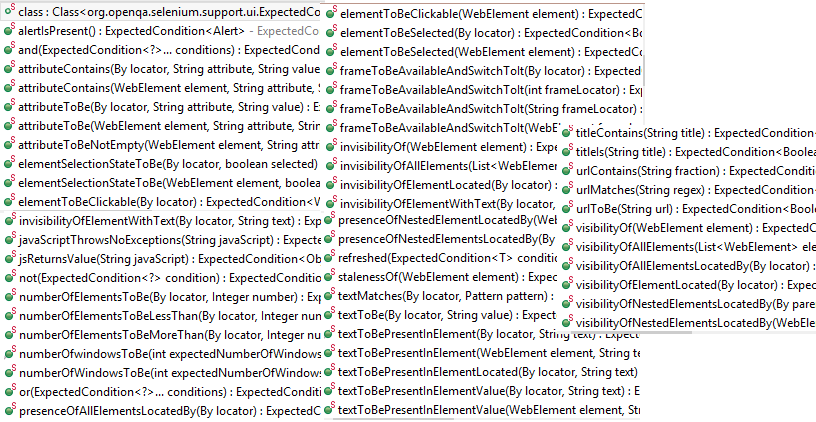
driver.findElement(By.*id*("js-hotelName-0")).click();

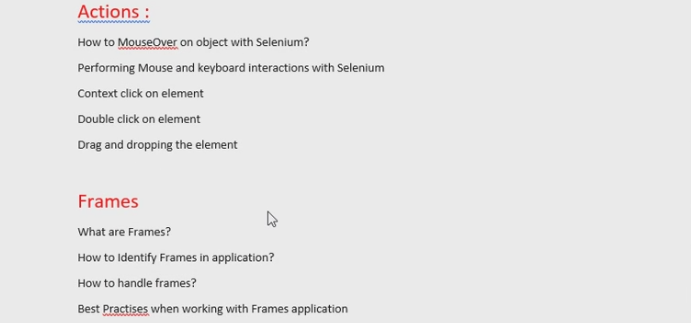
//implicit wait doesn't have condition but explicit wait have so many condition.

}

}

Explicitwait can be used during below condition, for ExpectedConditions.





**Action: What is action class in selenium webdriver?**

Action class is designed to handle the **mouse hover and keyboard** **interaction** with the user. It is also used to simulate user gestures on the application (through mobile/web)

This class provides specific methods to handle that in an application which we generally do by mouse or keyboard. Ex.

1. How do you right click with selenium?
2. How do you double click?
3. How do you write something in capital letters?
4. How do you drag & drop the element?

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

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

**import** org.openqa.selenium.Keys;

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

**import** org.openqa.selenium.WebElement;

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

**import** org.openqa.selenium.interactions.Actions;

**public class**  ActionsDemo {

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

System.*setProperty*("webdriver.chrome.driver", "D:\\Everything\_on\_Selenium\\chromedriver\_win32\\chromedriver.exe");

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

driver.get("https://www.amazon.com/");

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

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

//Below is the way to create a action claa object

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

WebElement move = driver.findElement(By.*xpath*("//a[@id='nav-link-accountList']"));

//How enter string in capital letters in any search box.

//Below we are using a text field and sending some text(in CAPs)

//a.moveToElement(driver.findElement(By.id("twotabsearchtextbox"))).click().keyDown(Keys.SHIFT).sendKeys("hello").build().perform();

//In above syntax we are pressing the shift button and writng some text

//Now how to double click on the entire text selected

a.moveToElement(driver.findElement(By.*id*("twotabsearchtextbox"))).click().keyDown(Keys.***SHIFT***).sendKeys("hello").doubleClick().build().perform();

//a.moveToElement(driver.findElement(By.xpath("//a[@id='nav-link-accountList']"))).build().perform();

//In stead of the above one, we can use it using a webelement variable also.

a.moveToElement(move).build().perform();

//When we create an action then we have to build()& perform()it, bcoz build makes it ready to execute.

//By using build we can make some composite actions also

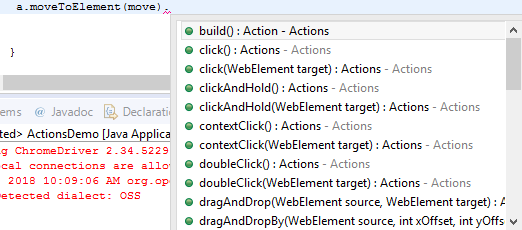
//Now let's right click on that element

a.moveToElement(move).contextClick().build().perform();

//Action class has a feature called drag & drop, that we will learn in the frames class

}

}



**How to handle multiple windows in selenium?**

How can we handle child windows in selenium?

How to switch from window to window in selenium and then get back to main/parent window?

**import** java.util.Iterator;

**import** java.util.Set;

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

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

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

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

**public class**  MultipleWindow {

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

System.*setProperty*("webdriver.chrome.driver", "D:\\Everything\_on\_Selenium\\chromedriver\_win32\\chromedriver.exe");

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

driver.get("https://accounts.google.com/SignUp");

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

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

driver.findElement(By.*xpath*(".//\*[@id='wrapper']/div[2]/div/div[1]/p/a")).click();

System.***out***.println("Before switching to child !!!");

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

//In above line it will give the title of parent window

//Now we need to get the title of child window

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

//This will give all the open window IDs that is opened at the moment

//below iterator will used for iterating on all the IDs and get whenever is ID is needed

Iterator<String>it=ids.iterator();

String parentID=it.next();

String childID=it.next();//it will take to the child window

driver.switchTo().window(childID); //it will switch to the window depending on the ID

//> that is passed to the window()

System.***out***.println("After switching to child !!!");

System.***out***.println(driver.getTitle());//it will print the child title

//All the operation you will do now , it will from child window

//Now how to switch back to parent window

driver.switchTo().window(parentID);

System.***out***.println("After switching back to parent again !!!");

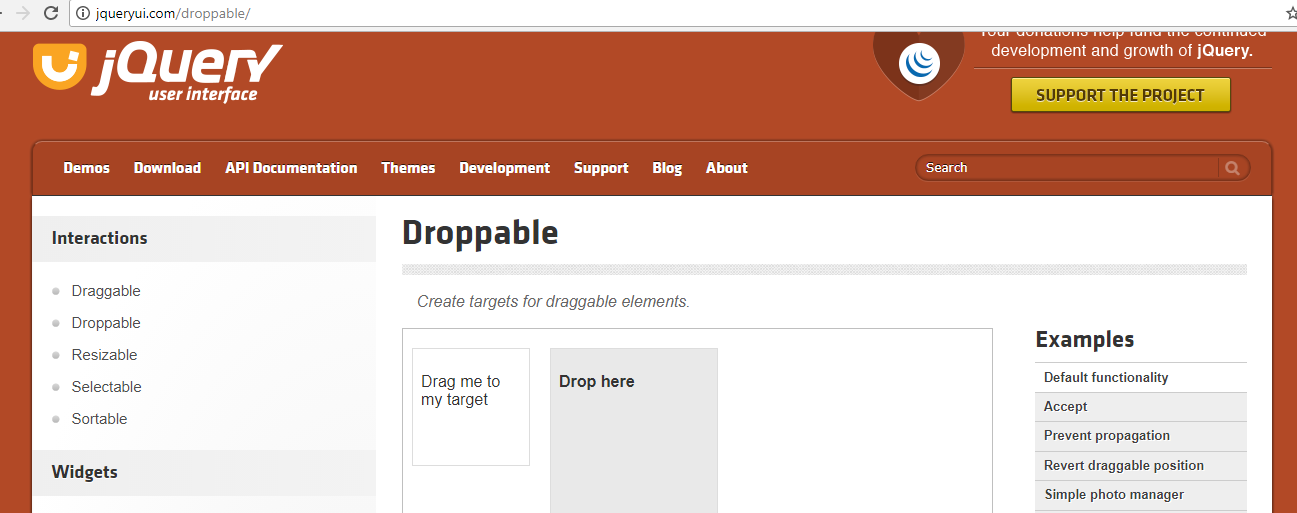
System.***out***.println(driver.getTitle());//it will print the parent title

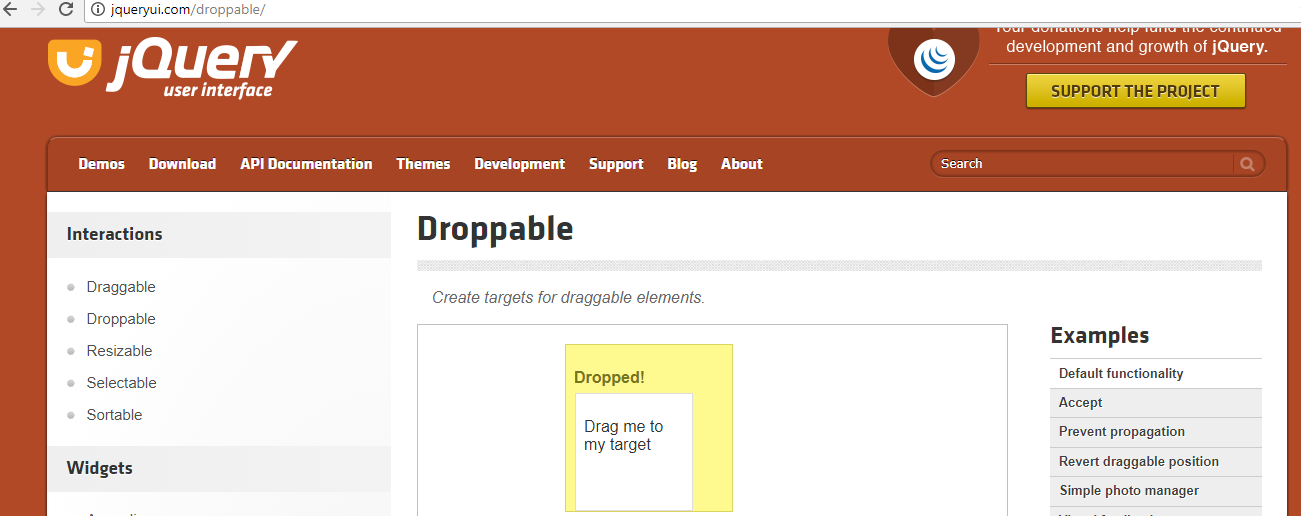
}

}

**Frames: -**

Example of drag and drop - <http://jqueryui.com/droppable/>





**What are frames in web design: -** In the context of a web browser, a frame is a part of a web page or browser window which displays content independent of its container, with the ability to load content independently. The HTML or media elements that go in a frame may or may not come from the same web site as the other elements of content on display.

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

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

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

**import** org.openqa.selenium.WebElement;

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

**import** org.openqa.selenium.interactions.Actions;

**public class**  Frames {

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

System.*setProperty*("webdriver.chrome.driver", "D:\\Everything\_on\_Selenium\\chromedriver\_win32\\chromedriver.exe");

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

driver.get("http://jqueryui.com/droppable/");

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

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

//Now let's know how many frames are there in the page

//System.out.println(driver.findElements(By.tagName("ifrane")).size());

//So how to tell program that our webelemt is inside a frame

driver.switchTo().frame(driver.findElement(By.*cssSelector*("iframe.demo-frame")));

//in this CSS selector its tagname.class-name or iframe[class='demo-frame']

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

//Now we will drag & drop

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

WebElement source = driver.findElement(By.*id*("draggable"));

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

a.dragAndDrop(source, target).build().perform();

}

}

[info@qaclickacademy.com](mailto:info@qaclickacademy.com)

Working with Iframe and page content

IFrame Is another web element and you cannot locate Its element directly In selenium webdriver. To work with IFrame element In selenium webdriver, first of all you need to select that IFrame using syntax like bellow.

//switch to frame1. frame1 Is ID of frame.

driver.switchTo().frame("frame1");

Now you can work with any element which Is Inside frame1. Now suppose you want to switch back to page content then you need to use syntax like bellow.

//Switch back to page content.

driver.switchTo().defaultContent();

After above syntax execution, You can work with page elements.

Working with multiple frames on same page

If there are multiple Iframes on single page then you cannot directly navigate from Iframe1 to IFrame2. For that, you need to select page In between as bellow.

//switch to frame1

driver.switchTo().frame("frame1");

driver.findElement(By.xpath("//td[contains(text(),'Cow')]/preceding-sibling::td/input[@type='checkbox']")).click();

//Switch back to page content.

driver.switchTo().defaultContent();

//switch to frame2

driver.switchTo().frame("frame2");

driver.findElement(By.xpath("//input[@value='Boat']")).click();

**Practice: Find out all the links present in from ebay website.** Below are few more similar questions

Give me the number of links present in the entire page.

Give me report of all the links.

Give me the count of links present in footer/header section of the page.

Give me the links count of 2nd column of all the columns present in footer section.

How will you link which is always changing i.e. dynamic links?

In a webpage, if anything must be designed as a link, then designer must give as “a” i.e. anchor tag. So, in order to find out the count of links, we can get that by counting the number of anchor tags.

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

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

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

**import** org.openqa.selenium.WebElement;

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

**public class**  CountOfLinks {

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

System.*setProperty*("webdriver.chrome.driver", "D:\\Everything\_on\_Selenium\\chromedriver\_win32\\chromedriver.exe");

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

driver.get("https://www.ebay.com/");

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

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

//Count of the links in the entire page

System.***out***.print("Total number links in ebay website page : ");

System.***out***.println(driver.findElements(By.*tagName*("a")).size());

//Count of links in the footer section of the page

WebElement footer=driver.findElement(By.*xpath*("//footer[@id='glbfooter']"));

//Now in the above line I have created a footer webelemt and bu that webelement

//>I will count the number of links present in footer section.

//The above is way to create driver object for footer section

System.***out***.print("Number links present in footer section : ");

System.***out***.println(footer.findElements(By.*tagName*("a")).size());

//get the links of all links present in 2nd column of footer section

WebElement footer\_col2=driver.findElement(By.*xpath*(".//\*[@id='gf-BIG']/table/tbody/tr/td[2]/ul"));

System.***out***.print("Number links present in 2nd column of footer section : ");

System.***out***.println(footer\_col2.findElements(By.*tagName*("a")).size());

//Now we want to print all the links of 2nd column

String beforeClicking = **null**;

String afterClicking;

System.***out***.println("Print ing all the links of 2nd column below !!! ");

**for**(**int i=**0;i<footer\_col2.findElements(By.*tagName*("a")).size();i++)

{

System.***out***.println(footer\_col2.findElements(By.*tagName*("a")).get(i).getText());

//Now I need to click on the "Site map" link present in the 2nd column

//Suppose the "Site map" link is dynamic , it changes it's place everyday

**if**(footer\_col2.findElements(By.*tagName*("a")).get(i).getText().contains("Site map"))

{

//checking what is the link title before clicking

System.***out***.print("Title of main page : ");

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

beforeClicking = driver.getTitle();

footer\_col2.findElements(By.*tagName*("a")).get(i).click();

//When we get sitemap , then it will open in another page , so now when

//>we traverse again in the for loop it will throw error ,So, break it.

**break**;

}

}

System.***out***.print("Now the title of newly opened page : ");

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

afterClicking = driver.getTitle();

//Now compare the old and new page titles

**if**(beforeClicking!=afterClicking)

{

**boolean abc** = driver.findElement(By.*xpath*("html/body/div[2]/div[1]/h1")).isDisplayed();

**if**(abc==**true**)

{

System.***out***.print("Text present in SiteMap page is : ");

System.***out***.println(driver.findElement(By.*xpath*("html/body/div[2]/div[1]/h1")).getText());

System.***out***.println("Test case is PASS !!!");

}

/\*//Another way of checking it without finding the xpath/css

if(driver.getPageSource().contains("Sitemap"))

{

System.out.println("PASS !!!");

}\*/

}

**else**

{

System.***out***.println("Test case is FAIL !!!");

}

}

}

**O/P: -**

Starting ChromeDriver 2.34.522940 (1a76f96f66e3ca7b8e57d503b4dd3bccfba87af1) on port 7186

Only local connections are allowed.

Jan 16, 2018 10:21:18 AM org.openqa.selenium.remote.ProtocolHandshake createSession

INFO: Detected dialect: OSS

Total number links in ebay website page : 382

Number links present in footer section : 87

Number links present in 2nd column of footer section : 10

Print ing all the links of 2nd column below !!!

Sell

Start selling

Learn to sell

Business sellers

Affiliates

Mobile apps

Developers

Security center

eBay official time

Site map

Title of main page : Electronics, Cars, Fashion, Collectibles, Coupons and More | eBay

Now the title of newly opened page : Sitemap | eBay

Text present in SiteMap page is : Sitemap

Test case is PASS !!!

**SELENIUM PART-3**

**Handling Calendar UI in travel websites: -**

<https://www.path2usa.com/travel-companions>

We will not find any xpath or CSS here to get the address of the web element. We will try to point out on the date by giving the date and the month name. How? Let’s see.

**package** basics;

**import** java.util.List;

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

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

**import** org.openqa.selenium.WebElement;

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

**publicclass** Generic\_way\_2\_handle\_Calender {

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

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

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

driver.get("https://www.path2usa.com/travel-companions");

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

//Aug 23 - How click on a perticular date

driver.findElement(By.*xpath*(".//\*[@id='travel\_date']")).click();

//As we saw from the HTML tags from calender that all the dates falls under

//>day class. Now we will take all the dates into a list

**while**(!driver.findElement(By.*cssSelector*("[class='datepicker-days'] [class='datepicker-switch']")).getText().contains("April"))

{

driver.findElement(By.*cssSelector*("[class='datepicker-days'] th[class='next']")).click();

}

//Here while loop will give us the month, it will go inside the loop, whenever it will not find the month"April"

List<WebElement>dates = driver.findElements(By.*className*("day"));

//Grab the common attribute , put it into and iterate

**int**count = driver.findElements(By.*className*("day")).size();

**for**(**int**i=0; i<count;i++)

{

String text = driver.findElements(By.*className*("day")).get(i).getText();

**if**(text.equalsIgnoreCase("26"))

{

driver.findElements(By.*className*("day")).get(i).click();

**break**;

}

}

System.***out***.println("Test execution completed !");

}

}

Below program is the same generic calendar program on make my trip but not fully correct, do check it later

**package** basics;

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

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

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

**publicclass** Generic\_Calender\_MakeMyTrip {

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

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

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

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

driver.get("https://www.makemytrip.com/flights/");

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

Thread.*sleep*(6000L);

driver.findElement(By.*xpath*(".//\*[@id='hp-widget\_\_depart']")).click();

//Selecting your desired month

**while**(!driver.findElement(By.*xpath*("//div[@class='ui-datepicker-title']/span[@class='ui-datepicker-month']")).getText().contains("MARCH"))

{

driver.findElement(By.*cssSelector*(".ui-icon.ui-icon-circle-triangle-e")).click();

}

**int**count1=driver.findElements(By.*xpath*("//div[@class='ui-datepicker-group ui-datepicker-group-first']/table[@class='ui-datepicker-calendar']/tbody/tr/td/a[@class='ui-state-default']")).size();

System.***out***.println(count1);

//Here total count will give the size of the month, to iterate

**for** (**int**i=0;i<count1;i++)

{

String s1=driver.findElements(By.*xpath*("//div[@class='ui-datepicker-group ui-datepicker-group-first']/table[@class='ui-datepicker-calendar']/tbody/tr/td/a[@class='ui-state-default']")).get(i).getText();

System.***out***.println(s1);

**if** (s1.equalsIgnoreCase("11"))

{

driver.findElements(By.*xpath*("//div[@class='ui-datepicker-group ui-datepicker-group-first']/table[@class='ui-datepicker-calendar']/tbody/tr/td/a[@class='ui-state-default']")).get(i).click();

**break**;

}

}

//RETURN DATE DATE

driver.findElement(By.*xpath*(".//\*[@id='hp-widget\_\_return']")).click();

Thread.*sleep*(3000);

System.***out***.println(driver.findElement(By.*cssSelector*("[class='ui-datepicker-title'] [class='ui-datepicker-month']")).getText());

**while**(!driver.findElement(By.*cssSelector*("[class='ui-datepicker-title'] [class='ui-datepicker-month']")).getText().contains("JULY"))

{

driver.findElement(By.*cssSelector*(".ui-icon.ui-icon-circle-triangle-e")).click();

}

**int**count2=driver.findElements(By.*xpath*("//div[@class='ui-datepicker-group ui-datepicker-group-first']/table[@class='ui-datepicker-calendar']/tbody/tr/td/a[@class='ui-state-default']")).size();

System.***out***.println(count2);

//Here total count will give the size of the month, to iterate

**for** (**int**i=0;i<count2;i++)

{

String s2=driver.findElements(By.*xpath*("//div[@class='ui-datepicker-group ui-datepicker-group-first']/table[@class='ui-datepicker-calendar']/tbody/tr/td/a[@class='ui-state-default']")).get(i).getText();

System.***out***.println(s2);

**if** (s2.equalsIgnoreCase("15"))

{

driver.findElements(By.*xpath*("//div[@class='ui-datepicker-group ui-datepicker-group-first']/table[@class='ui-datepicker-calendar']/tbody/tr/td/a[@class='ui-state-default']")).get(i).click();

**break**;

}

}

}

}

**Practical Problems and Methods to handle them in Selenium: -**

Always check if the web element is inside the html page or the in inside the frame if it is inside the frame then we have to deal it differently.

Search in Google that a how to know whether element is present inside the page or not.

**isDisplayed() –** is used to know whether the object is visible or invisible. Invisible means code is present in application, but for user it’s in invisible mode. Pre-requisite is object should be present in the application. Either visible/invisible

Now there is a frame present in a webpage, now to know if the object present in that frame or not? –

Now suppose we don’t know which frame it is, then we need to go to each and every frame and then we need to use a for loop.



This is the way to find, if an element is present in the web page or not. We should not use isDisplayed() here.

We have to use switchTo().defaultContent() to come back to webpage mode from frames , it is mandatory

Want to get the frame id where y object is present.

**package** basics;

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

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

**import**org.openqa.selenium.WebDriverException;

**import**org.openqa.selenium.WebElement;

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

**import**org.openqa.selenium.interactions.Actions;

**import**org.openqa.selenium.support.ui.ExpectedConditions;

**import**org.openqa.selenium.support.ui.WebDriverWait;

**publicclass** Frames\_Generic\_handling\_parent {

**publicstaticint** gotoframe(WebDriver driver1,By by)

{

driver1.switchTo().defaultContent();

**int**i;

**int**num = -1;

**int**a=driver1.findElements(By.*tagName*("iframe")).size();

**for**(i=0;i<a;i++)

{

driver1.switchTo().defaultContent();

driver1.switchTo().frame(i);

**int**b=driver1.findElements(by).size();

**if**(b>0)

{

num=i;

**break**;

}

}

driver1.switchTo().defaultContent();

**return**num;

}

}

**package** basics;

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

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

**import**org.openqa.selenium.WebDriverException;

**import**org.openqa.selenium.WebElement;

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

**import**org.openqa.selenium.interactions.Actions;

**import** org.openqa.selenium.support.ui.ExpectedConditions;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**publicclass** Frames\_Generic\_handling\_child **extends** Frames\_Generic\_handling\_parent{

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

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

WebDriverWait wd=**new** WebDriverWait(driver,7);

driver.get("https://fantasycricket.dream11.com/IN/");

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

**int**m=*gotoframe*(driver,By.*xpath*(".//\*[@id='recaptcha-anchor']"));

driver.switchTo().frame(m);

driver.findElement(By.*xpath*(".//\*[@id='recaptcha-anchor']/div[5]")).click();

driver.switchTo().defaultContent();

wd.until(ExpectedConditions.*frameToBeAvailableAndSwitchToIt*(By.*id*("I1\_1441700500937")));

**int**j=*gotoframe*(driver,By.*xpath*(".//\*[@id='recaptcha-verify-button']"));

**if**(j!=-1)

{

driver.switchTo().frame(j);

//WebDriverWait wd=new WebDriverWait(driver,5);

//wd.until(ExpectedConditions.(By.xpath(".//\*[@id='recaptcha-verify-button']")));

driver.findElement(By.*xpath*(".//\*[@id='recaptcha-verify-button']")).click();

}

**Else**

{

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

}

}

}

We should avoid using thread.sleep it will make the thread performance worse.

**Handling Auto suggestion dropdown: -**

**package** basics;

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

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

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

**import** org.openqa.selenium.support.ui.ExpectedConditions;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**publicclass** Auto\_Suggestion\_Dropdown{

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

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

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

driver.get("https://fantasycricket.dream11.com/IN/");

WebDriverWait wd=**new** WebDriverWait(driver,7);

// it will wait for the element maximum up to the given time, whenever an element is found it will continue its execution without waiting for the whole second mentioned here - This is explicit wait

//But in case thread.sleep(3000) - Though it has found in 1 sec, still it will wait for 7 sec.

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

driver.findElement(By.*xpath*(".//\*[@id='m\_rtxtEmail1']")).sendKeys("chit");

wd.until(ExpectedConditions.*visibilityOfElementLocated*(By.*xpath*(".//\*[@id='m\_frmRegister']/div[1]/ul")));

//in above it waits for auto suggestion & below it clicks on one of them

driver.findElement(By.*xpath*(".//\*[@id='m\_frmRegister']/div[1]/ul/li[2]/p")).click();

}

}

**How to handle Dynamic web element: -**If there is any object which sometimes appear and sometimes it doesn't, in that case how do we handle

Try

{

//Code to switch to alerts & click on it

}

catch(Exception e)

{

}

We can also use a size(), to check and use a if condition to go ahead on that part.

**Code on Frames:**

package basics;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebDriverException;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.interactions.Actions;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

public class Day extends Day1{

public static void main(String[] args) {

// TODO Auto-generated method stub

WebDriver driver=new FirefoxDriver();

WebDriverWait wd=new WebDriverWait(driver,7);

driver.get("https://fantasycricket.dream11.com/IN/");

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

int m=gotoframe(driver,By.xpath(".//\*[@id='recaptcha-anchor']"));

driver.switchTo().frame(m);

driver.findElement(By.xpath(".//\*[@id='recaptcha-anchor']/div[5]")).click();

driver.switchTo().defaultContent();

wd.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt(By.id("I1\_1441700500937")));

int j=gotoframe(driver,By.xpath(".//\*[@id='recaptcha-verify-button']"));

if(j!=-1)

{

driver.switchTo().frame(j);

driver.findElement(By.xpath(".//\*[@id='recaptcha-verify-button']")).click();

}

else{

System.out.println("ops");

}

}

}

// Code

package basics;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebDriverException;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.interactions.Actions;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

public class Day1 {

public static int gotoframe(WebDriver driver1,By by)

{

driver1.switchTo().defaultContent();

int i;

int num = -1;

int a=driver1.findElements(By.tagName("iframe")).size();

for(i=0;i<a;i++)

{

driver1.switchTo().defaultContent();

driver1.switchTo().frame(i);

int b=driver1.findElements(by).size();

if(b>0)

{

num=i;

break;

}

}

driver1.switchTo().defaultContent();

return num;

}}

**Famous interview question:-**

1. **How to handle table in Webpage?**

Test Case: My goal is to go to cricbuzz website, go to a match score card and get the runs scored by each batsman and add them and compare with the total

We have to use common css for a table but with parent also

**package** basics;

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

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

**import** org.openqa.selenium.WebElement;

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

**publicclass** Table\_Handing\_Cricbuzz {

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

**int**sum=0;

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

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

driver.get("http://www.cricbuzz.com/live-cricket-scorecard/17695/aus-vs-eng-3rd-odi-england-tour-of-australia-2018");

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

//Now lets search a perticular table where we want to check our test case

WebElement table = driver.findElement(By.*cssSelector*("div[class='cb-col cb-col-100 cb-ltst-wgt-hdr']"));

//Now lets use that table to get the other elements

//Now lets a go the class name of each row

**int**rowcount=table.findElements(By.*cssSelector*("div[class='cb-col cb-col-100 cb-scrd-itms']")).size();

//Now we click on the 3rd chlid of the row

**int**batsman\_count=table.findElements(By.*cssSelector*("div[class='cb-col cb-col-100 cb-scrd-itms'] div:nth-child(3)")).size();

**for**(**int**i=0;i<batsman\_count-2;i++)

{

String value=table.findElements(By.*cssSelector*("div[class='cb-col cb-col-100 cb-scrd-itms'] div:nth-child(3)")).get(i).getText();

//Now let's convert string value into int

**int**valueInt = Integer.*parseInt*(value);

sum=sum+valueInt;

}

System.***out***.println("Sum of all the runs without extra :"+sum);

//Now reading the element based on a string

String extras=driver.findElement(By.*xpath*("//div[text()='Extras']/following-sibling::div")).getText();

**int**extraInt=Integer.*parseInt*(extras);

**int**TotalInt=extraInt+sum;

System.***out***.println("Sum of all the runs & extra :"+TotalInt);

String ActualTotal=driver.findElement(By.*xpath*("//div[text()='Total']/following-sibling::div")).getText();

**int**ActualTotalInt = Integer.*parseInt*(ActualTotal);

**if**(TotalInt==ActualTotalInt)

{

System.***out***.println("Count Matches !!!");

}

**else** System.***out***.println("Count has not matched !!!");

}

}

O/P:-

**Sum of** all the runs :178

Sum of all the runs & extra :189

Count Matches !!!

1. **How to handle SSL certification?**

**package** basics;

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

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

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

**import** org.openqa.selenium.remote.CapabilityType;

**import** org.openqa.selenium.remote.DesiredCapabilities;

**publicclass** ChromeOptions\_for\_SSL {

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

// SSL certificate

// desired capability is a class to provide the knowledge to the browser

DesiredCapabilities ch = DesiredCapabilities.*chrome*();

//ch.acceptInsecureCerts();

//You can use the above or the below

ch.setCapability(CapabilityType.***ACCEPT\_INSECURE\_CERTS***, **true**);

ch.setCapability(CapabilityType.***ACCEPT\_SSL\_CERTS***, **true**);

ChromeOptions c= **new** ChromeOptions();

//chromeoptions belongs to our own local chrome browser

c.merge(ch);

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

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

driver.get("google.com");

}

}

1. **How to maximize the window & delete cookies?**

**package** basics;

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

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

**publicclass** Deleting\_Cookies {

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

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

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

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

//Below is the command to delete the cookies

driver.manage().deleteAllCookies();

driver.manage().deleteCookieNamed("sessionKey");

//Click on any link , then you will be re-direct to login page

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

}

}

1. **How to take screenshot in selenium web-browser?**

**package** basics;

**import** java.io.File;

**import** java.io.IOException;

**import** org.openqa.selenium.OutputType;

**import** org.openqa.selenium.TakesScreenshot;

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

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

**import** org.apache.commons.io.FileUtils; // if you don't have this package then download

//https://commons.apache.org/proper/commons-io/download\_io.cgi

**publicclass** Take\_ScreenShot {

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

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

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

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

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

//Below is the way to take screenshot

File src=((TakesScreenshot)driver).getScreenshotAs(OutputType.***FILE***);

//Now we will copy the screenshot into our system

FileUtils.*copyFile*(src,**new** File("E:\\AUTOMATION\_WORKSPACE\\Screenshots\\google.png"));

//You can put this C driver, you need to have admin rights and its complicated

//if you don't have any other directory except C then direct it to User folder in C

}

}

1. **How to kill the process in selenium?**

**package** basics;

**import** java.io.IOException;

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

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

**import** org.openqa.selenium.os.WindowsUtils;

**publicclass** Kill\_A\_Running\_Process {

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

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

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

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

//Killing excel opened in our system now

WindowsUtils.*killByName*("excel.exe");

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

}

}

**Xpath with 2 locators**

//tagname[@attriute=’value’]

//input[@id=’LastName’]

//input[@id=’LastName’ and @name=’LastName’] – Both should be matching

//input[@id=’LastName’ or @name=’LastName’] – anyone of them should be matching

$x(“”) – Inside this the xpath inside the console , we can use to test

Like this - $x(“//input[@id=’LastName’ or @name=’LastName’]”)

(//a)[3] – When we have multiple anchor tags , we can use them like this , here 3rdone will be selected.

**How to locate dynamic web elements: -**

firstname123

firstname234

firstname345 – Now how to locate such elements which changes dynamically like this

//input[@name=’ FirstName’]

2 functions : starts-with()

ends-with()

contains()

So the xpath is - //input[starts-with(@name,'firstname')]

For contains - //input[contains(@name,'firstname')]

$x("//input[starts-with(@name,'firstname')]") – We can try like this in console

$x("//input[contains(@name,'firstname')]") – This is for contains

**SELENIUM PART-4**

**Selenium Grid: -(Distributed Test Execution)**

* Selenium-Grid allows you to run your tests on different machines against different browser.
* Trigger the test cases from local machine and it should run in all connected remote machine and gives the results in the local machine.
* Selenium-Grid supports distributed test execution.
* You can also run the Tests in parallel in multiple machines on selenium grid.
* See Below: Hub is the local machine here.
* **What is HUB? –**The HUB is the central point that will receive all the test requests and distribute them to the right nodes. There should only be one hub in a grid. The machine containing the HUB is where the tests will be triggered, but you will see the browser being automated on the node.
* **What is Node? –** These are selenium instances that will execute the tests that you loaded on the hub. Nodes can be launched on multiple machines with platforms and browsers.

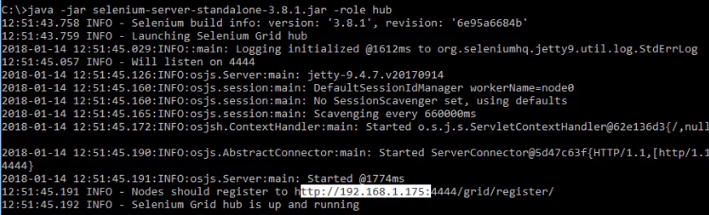


* Advantages: -

1. Hub & Nodes can be on different OS & browser. The main job in grid is to configure the hub & node.
2. Mostly for above infrastructure need, people need Grid.
3. One more advantage is parallel execution.

* To work with Selenium Grid, we need to download selenium server JAR. The Selenium Server is needed in order to run Remote Selenium WebDriver.
* **Steps to configure Grid: -**

1. Download selenium server jar. Download “Selenium Standalone Server” from <http://www.seleniumhq.org/download/>
2. Now we need to register HUB. Open cmd prompt, Go the location where your above JAR is located. Give the same JAR name in the below command.
3. Then hit a command: - **java –jar selenium-server-standalone-3.8.1.jar –role hub**

****

1. In above console as it is shown, 4444 is the port here, which is listening.
2. <http://192.168.1.175:4444/grid/console> - Copy & paste this browser, this is where our server is listening to the nodes.
3. Now to connect a node to our Hub, we need to connect to one more machine and trigger some commands.
4. Login to another machine& register it as Node for the Hub.
5. Download standalone server JAR file in the node machine as well.
6. Open the command prompt of node machine > Go to the location where the JAR is stored.
7. Hit the command:-

**java –jar selenium-server-standalone-3.8.1.jar –role webdriver**

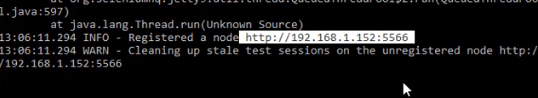
**–hub** [**http://192.168.1.175:4444/grid/register**](http://192.168.1.175:4444/grid/register) **-port 5566**

1. In the above command we need to provide hub IP address. So for that Open Hub machine’s command prompt, it is clearly mentioned when we ran the same command earlier there, that node should register to <http://192.168.1.175:4444/grid/register>. Give port as 5566 to which the hub will listen to node.
2. We can also get the Hub IP address using ipconfig command.
3. Once you hit the above command, you will see a successful log message in node console. And at the same time you will see the below in the Hub console also.

Node Machine console: -



Hub Machine console: -



1. But one thing you need to remember here is to run the node in the machine, you have to download JAVA & make sure you set in system variable just like we did for our common selenium set-up.
2. Previously we were giving the local path of Gecko driver/ Chrome driver in our selenium program, But that will not work when we run the same programs on node.
3. We can give the geckodriver/chomedriver knowledge to node. Download them in node machine &Place it in the same path where exactly the selenium standalone Jar is present in node machine.
4. **Java –Dwebdriver.chrome.driver=”C:\Chromedriver.exe” –jar selenium-server-standalone-3.8.1.jar –role webdriver–hub** [**http://192.168.1.175:4444/grid/register -port 5566**](http://192.168.1.175:4444/grid/register%20-port%205566)
5. You need to give above command in node not the previous one where drivers were not defined.

* **Let’s run a test case which will run in Node & will be triggered from Hub: -**
* Now when we refresh<http://192.168.1.175:4444/grid/console> in our HUB browser. It will give below

****

* Where it says that it can open 5 firefox instances, 5 chrome instances & 1 IE instance. So, at a time it can open 11 instances.
* Selenium Standalone Server JAR needs to be imported to the selenium grid project running in HUB machine. Go to Libraries > Add external JARs.
* So our machine should set on which browser & OS our test should execute. So we need to use the concept “DesiredCapabilities” in selenium grid.

**package** basics;

**import** java.net.MalformedURLException;

**import** java.net.URL;

**import** org.openqa.selenium.Platform;

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

**import** org.openqa.selenium.remote.DesiredCapabilities;

**import** org.openqa.selenium.remote.RemoteWebDriver;

**publicclass** remotetest {

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

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

//Automated- chrome,firefox,ie,safari

DesiredCapabilities dc= **new** DesiredCapabilities();

dc.setBrowserName("chrome");

dc.setPlatform(Platform.***WINDOWS***);

//dc.se

//Webdriver driver=new ChromeDriver();

WebDriver driver= **new** RemoteWebDriver(**new** URL("http://localhost:4444/wd/hub"),dc);

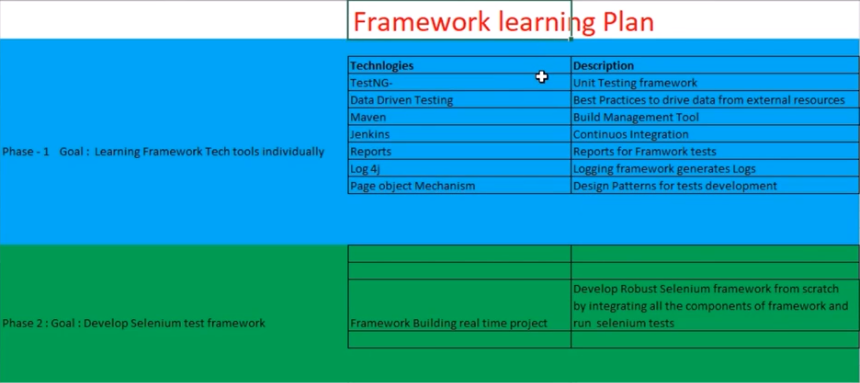
driver.get("http://rediff.com");

}

}

* To each browser, we have to hit the node command according to the browser specific driver and then we can run the test case.

**FRAMEWORK**

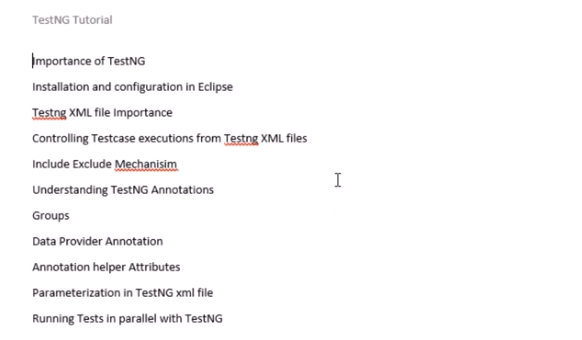
****

**Framework Part-1 TestNG:-**

**Scenario –** If we want to run test case 2 , but there is a pre-requisite that we have to run test case 1 before that, then only we can run test case 2.

Run only 40 test cases out of 500.

Run a test case with multiple set of data



<http://testng.org/doc/> - Official Website

TestNG is a testing framework inspired from JUnit and NUnit but introducing some new functionality that make it more powerful and easier to use, such as:

* Annotations.
* Run your tests in arbitrarily big thread pools with various policies available (all methods in their own thread, one thread per test class, etc...).
* Test that your code is multithread safe.
* Flexible test configuration.
* Support for data-driven testing (with @DataProvider).
* Support for parameters.
* Powerful execution model (no more TestSuite).
* Supported by a variety of tools and plug-ins (Eclipse, IDEA, Maven, etc...).
* Embeds BeanShell for further flexibility.
* Default JDK functions for runtime and logging (no dependencies).
* Dependent methods for application server testing.

TestNG is designed to cover all categories of tests:  unit, functional, end-to-end, integration, etc...

**How to download and configure it in Eclipse: -**

<http://testng.org/doc/>> Click on Eclipse tab present on the top of the webpage >(<http://testng.org/doc/eclipse.html> ) >

We can download testNG jars and can add it to our project. But if we are adding testNG plug-in, then no need to ad jars to each projects.

Go to Eclipse > Help > Install new software > carefully copy the site URL (<http://beust.com/eclipse-old/eclipse-6.7.0.20120825_1316/>) from install plug-in page of testNG website >paste it in “work with” tab of “Install new software” in eclipse >

To check TestNG > Go to Window > Click preferences > Search for TestNG

TestNG acts as a java compiler and execute all our test cases.

TestNG needs all its functions in method.

You need to have @Test annotation followed by method.Below is a testNG program.

**package** tutorial;

**import** org.testng.annotations.Test;

**publicclass** TestNG\_Day1 {

@Test

**publicvoid** demo()

{

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

}

}

[TestNG] Running:

C:\Users\LENOVO\AppData\Local\Temp\testng-eclipse--666284172\testng-customsuite.xml

Hello

PASSED: demo

===============================================

Default test

Tests run: 1, Failures: 0, Skips: 0

===============================================

===============================================

Default suite

Total tests run: 1, Failures: 0, Skips: 0

===============================================

[TestNG] Time taken by [FailedReporter passed=0 failed=0 skipped=0]: 1 ms

[TestNG] Time taken by org.testng.reporters.jq.Main@ea30797: 214 ms

[TestNG] Time taken by org.testng.reporters.EmailableReporter@dcf3e99: 35 ms

[TestNG] Time taken by org.testng.reporters.JUnitReportReporter@75a1cd57: 3 ms

[TestNG] Time taken by org.testng.reporters.XMLReporter@1963006a: 11 ms

[TestNG] Time taken by org.testng.reporters.SuiteHTMLReporter@6bf2d08e: 41 ms

How to trigger the program from a XML of testNG?

You can define multiple test cases from single class.

**package** tutorial;

**import** org.testng.annotations.Test;

**publicclass** TestNG\_Day1 {

@Test

**publicvoid** demo()

{

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

}

@Test

**publicvoid** secondTest()

{

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

}

}

[TestNG] Running:

C:\Users\LENOVO\AppData\Local\Temp\testng-eclipse--911341862\testng-customsuite.xml

Hello

Bye

PASSED: demo

PASSED: secondTest

===============================================

Default test

Tests run: 2, Failures: 0, Skips: 0

===============================================

===============================================

Default suite

Total tests run: 2, Failures: 0, Skips: 0

===============================================

[TestNG] Time taken by org.testng.reporters.XMLReporter@4f6f416f: 24 ms

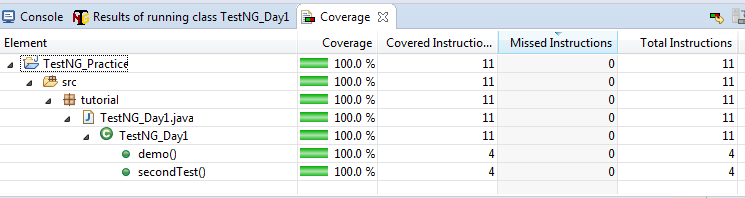
[TestNG] Time taken by org.testng.reporters.SuiteHTMLReporter@52815fa3: 190 ms

[TestNG] Time taken by org.testng.reporters.JUnitReportReporter@6dab9b6d: 5 ms

[TestNG] Time taken by org.testng.reporters.jq.Main@28b46423: 170 ms

[TestNG] Time taken by org.testng.reporters.EmailableReporter@1e461e41: 6 ms

[TestNG] Time taken by [FailedReporter passed=0 failed=0 skipped=0]: 0 ms



**How to create a testNG XML file for our project?** – Right click on the project > testNG > Convert to testNG > click next > Finish > Then it will create a testing.xml under our project

Below is the content of the xml file

<?xmlversion=*"1.0"*encoding=*"UTF-8"*?>

<!DOCTYPEsuiteSYSTEM"http://testng.org/testng-1.0.dtd">

<suitename=*"Suite"*parallel=*"none"*> - Test suite

<testname=*"Test"*> - Test module

<classes>

<classname=*"tutorial.TestNG\_Day1"*/>

</classes>

</test><!-- Test -->

</suite><!-- Suite -->

As you can see, in the above xml, suite is the starting and ending tag. So it considers test suite to be the parent of all the test cases. So, below is the hierarchy.

Test suite -> Test folder/module -> Test cases

When you create a new class under the same test module then you to add that class in the testing.xml? Now the testing .xml looks like below.

<?xmlversion=*"1.0"*encoding=*"UTF-8"*?>

<!DOCTYPEsuiteSYSTEM"http://testng.org/testng-1.0.dtd">

<suitename=*"Loan Department"*parallel=*"none"*>- Here test suite name is loan department.

<testname=*"Personal Loan"*>- Here test module name is personal loan

<classes>

<classname=*"tutorial.TestNG\_Day1"*/>

<classname=*"tutorial.TestNG\_Day2"*/>

</classes>

</test><!-- Test -->

</suite><!-- Suite -->

Now run this xml files as testNG suite: -

[TestNG] Running:

E:\AUTOMATION\_WORKSPACE\TestNG\_Practice\testng.xml

Hello

Bye

Personal loan!!

===============================================

Loan Department

Total tests run: 3, Failures: 0, Skips: 0

===============================================

Remember we need to always run the testing.xml file as testNG suite.

**package** tutorial;

**import** org.testng.annotations.Test;

**publicclass**TestNG\_Day2 {

@Test

**publicvoid** plone()

{

System.***out***.println("Personal loan !!");

}

}

**package** tutorial;

**import** org.testng.annotations.Test;

**publicclass**TestNG\_Day3 {

@Test

**publicvoid** WebloginCarloan()

{

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

}

@Test

**publicvoid** mobileloginCarloan()

{

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

}

@Test

**publicvoid** loginAPICarloan()

{

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

}

}

**package** tutorial;

**import** org.testng.annotations.Test;

**publicclass**TestNG\_Day4 {

@Test

**publicvoid** WebloginHomeloan()

{

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

}

@Test

**publicvoid** mobileloginHomeloan()

{

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

}

@Test

**publicvoid** loginAPIHomeloan()

{

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

}

}

**Prioritizing the test cases with testNG: -**

Below is the way to segregate the test cases into different test modules.

<?xmlversion=*"1.0"*encoding=*"UTF-8"*?>

<!DOCTYPEsuiteSYSTEM"http://testng.org/testng-1.0.dtd">

<suitename=*"Loan Department"*parallel=*"none"*>

<testname=*"Personal Loan"*>

<classes>

<classname=*"tutorial.TestNG\_Day1"*/>

<classname=*"tutorial.TestNG\_Day2"*/>

</classes>

</test><!-- Test -->

<testname=*"Car loan"*>

<classes>

<classname=*"tutorial.TestNG\_Day3"*></class>

</classes>

</test>

<testname=*"Home loan"*>

<classes>

<classname=*"tutorial.TestNG\_Day4"*></class>

</classes>

</test>

</suite><!-- Suite -->

[TestNG] Running:

E:\AUTOMATION\_WORKSPACE\TestNG\_Practice\testng.xml

Hello

Bye

Personal loan !!

WebloginCar

loginAPICar

mobileloginCar

WebloginHome

loginAPIHome

mobileloginHome

===============================================

Loan Department

Total tests run: 9, Failures: 0, Skips: 0

===============================================

Now to exclude or include the test modules

**How to exclude a specific test case from a test module?**

<?xmlversion=*"1.0"*encoding=*"UTF-8"*?>

<!DOCTYPEsuiteSYSTEM"http://testng.org/testng-1.0.dtd">

<suitename=*"Loan Department"*parallel=*"none"*>

<testname=*"Personal Loan"*>

<classes>

<classname=*"tutorial.TestNG\_Day1"*/>

<classname=*"tutorial.TestNG\_Day2"*/>

</classes>

</test><!-- Test -->

<testname=*"Car loan"*>

<classes>

<classname=*"tutorial.TestNG\_Day3"*></class>

</classes>

</test>

<testname=*"Home loan"*>

**<classes>**

**<classname=*"tutorial.TestNG\_Day4"*>**

**<methods>**

**<excludename=*"mobileloginHomeloan"*></exclude>**

**</methods>**

**</class>**

</classes>

</test>

</suite><!-- Suite -->

[TestNG] Running:

E:\AUTOMATION\_WORKSPACE\TestNG\_Practice\testng.xml

Hello

Bye

Personal loan!!

WebloginCar

loginAPICar

mobileloginCar

WebloginHome

loginAPIHome

===============================================

Loan Department

Total tests run: 8, Failures: 0, Skips: 0

===============================================

* See now 1 test case has reduced, from 9 to 8. So, this is how we can remove methods from our test execution.
* There is another tag called include. We can use it when only 1/few test cases are asked to include.

Example of include:

<?xmlversion=*"1.0"*encoding=*"UTF-8"*?>

<!DOCTYPEsuiteSYSTEM"http://testng.org/testng-1.0.dtd">

<suitename=*"Loan Department"*parallel=*"none"*>

<testname=*"Personal Loan"*>

<classes>

<classname=*"tutorial.TestNG\_Day1"*>

<methods>

<includename=*"secondTest"*></include>

</methods>

</class >

<classname=*"tutorial.TestNG\_Day2"*/>

</classes>

</test><!-- Test -->

<testname=*"Car loan"*>

<classes>

<classname=*"tutorial.TestNG\_Day3"*></class>

</classes>

</test>

<testname=*"Home loan"*>

<classes>

<classname=*"tutorial.TestNG\_Day4"*>

<methods>

<excludename=*"mobileloginHomeloan"*></exclude>

</methods>

</class>

</classes>

</test>

</suite><!-- Suite -->

[TestNG] Running:

E:\AUTOMATION\_WORKSPACE\TestNG\_Practice\testng.xml

Bye

Personal loan !!

WebloginCar

loginAPICar

mobileloginCar

WebloginHome

loginAPIHome

===============================================

Loan Department

Total tests run: 7, Failures: 0, Skips: 0

===============================================

**Executing the Test cases at package level with regex: -**

Q) Suppose there is a situation that we need to remove all the test cases/methods from all the class/test modules which name starts “mobile” while executing them.

Ans: - We can achieve this by using mobile.\* in exclude statement in testing.xml

<?xmlversion=*"1.0"*encoding=*"UTF-8"*?>

<!DOCTYPEsuiteSYSTEM"http://testng.org/testng-1.0.dtd">

<suitename=*"Loan Department"*parallel=*"none"*>

<testname=*"Personal Loan"*>

<classes>

<classname=*"tutorial.TestNG\_Day1"*>

<methods>

<includename=*"secondTest"*></include>

</methods>

</class >

<classname=*"tutorial.TestNG\_Day2"*/>

</classes>

</test><!-- Test -->

<testname=*"Car loan"*>

<classes>

<classname=*"tutorial.TestNG\_Day3"*>

<methods>

<excludename=*"mobile.\*"*></exclude>

</methods>

</class>

</classes>

</test>

<testname=*"Home loan"*>

<classes>

<classname=*"tutorial.TestNG\_Day4"*>

<methods>

<excludename=*"mobile.\*"*></exclude>

</methods>

</class>

</classes>

</test>

</suite><!-- Suite -->

**How to run package wise:** Below is the way to run all test cases of package

<?xmlversion=*"1.0"*encoding=*"UTF-8"*?>

<!DOCTYPEsuiteSYSTEM"http://testng.org/testng-1.0.dtd">

<suitename=*"Loan Department"*parallel=*"none"*>

<testname=*"Personal Loan"*>

<packages>

<packagename=*"tutorial"*></package>

</packages>

</test>

</suite><!-- Suite -->

**TestNG Annotation: -**

Suppose we have to run something before any test case start. i.e. pre-requisite of any test case.

**@BeforeTest**

**publicvoid** prerequisite()

{

System.***out***.println("Before executing any test cases !!!");

}

We can put this method any part of any class but it will get executed first.

**@AfterTest**

**publicvoid** atTheend()

{

System.***out***.println("This is will after Test!!");

}

We can put this method any part of any class but it will get executed at the last.

* Both the above function will work at the package level. But if we give it in test module level. Then they will run in as per the module level only. **Start of the module and end of the module.**

**@BeforeSuite**

**publicvoid** beforeSuite()

{

System.***out***.println("beforeSuite : Before anything !!!");

}

BeforeSuite method any part of any class but it will get executed at the starting of the all test cases be it any folder/module but it will have package scope or entire xml file.

**@AfterSuite**

**publicvoid** beforeSuite()

{

System.***out***.println("AftereSuite : After everything !!!");

}

Similarity AfterSuite method any part of any class but it will get executed at the end of the all test cases be it any folder/module but it will have package scope or entire xml file.

**@BeforeMethod**

**publicvoid** beforeMethodCar()

{

System.***out***.println("WeblogoutCarloan : Before Method implemeted !!!");

}

**@AfterMethod**

**publicvoid** afterMethodCar()

{

System.***out***.println("WeblogoutCarloan : After Method implemeted !!!");

}

@BeforeMethod&@AfterMethod will run before & after every method in the same class/module.

@BeforeClass

**publicvoid** beforeclass()

{

System.***out***.println("Before implementing TestNG\_Day3 or Car loan module ");

}

@AfterClass

**publicvoid** afterclass()

{

System.***out***.println("After implementing TestNG\_Day3 or Car loan module ");

}

This above 2 annotations used for Before & after implementing class or test module

* **Important: -Order of test order execution depends on method name alphabetic order.**

**Grouping Test cases: -**How to prepare a group of test cases? Suppose for smoke testing I need to run some 100 test cases out of all. So, how to group them and run them together?

We have to give group tag like below in all the methods in which ever class they may be but it will be considered under same group.

@**Test(groups={"Smoke"})**

**publicvoid** mobilelogoutCarloan()

{

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

}

@Test(groups={"Smoke"})

**publicvoid** mobileloginHomeloan()

{

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

} @Test(groups={"Smoke"})

**publicvoid** plone()

{

System.***out***.println("Personal loan !!");

}

Below is the xml file for the same.

<?xmlversion=*"1.0"*encoding=*"UTF-8"*?>

<!DOCTYPEsuiteSYSTEM"http://testng.org/testng-1.0.dtd">

<suitename=*"Loan Department"*parallel=*"none"*>

<testname=*"Regression"*>

<groups>

<run>

<includename=*"Smoke"*></include>

</run>

</groups>

<classes>

<classname=*"tutorial.TestNG\_Day1"*/>

<classname=*"tutorial.TestNG\_Day2"*/>

<classname=*"tutorial.TestNG\_Day3"*/>

<classname=*"tutorial.TestNG\_Day4"*/>

</classes>

</test>

</suite><!-- Suite -->

As we have used include in the above program, we can use exclude a group also.

**Annotation Helper attribute with examples: -**Depends attribute

Below program without depends attribute

**package** tutorial;

**import** org.testng.annotations.Test;

**publicclass** Day2 {

@Test

**publicvoid** webloginCarloan()

{

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

}

@Test

**publicvoid** mobileloginCarloan()

{

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

}

@Test

**publicvoid** APICarloan()

{

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

}

}

[TestNG] Running:

C:\Users\chitta\AppData\Local\Temp\testng-eclipse--21603853\testng-customsuite.xml

loginAPICarloan

mobileloginCarloan

WebloginCarloan

PASSED: APICarloan

PASSED: mobileloginCarloan

PASSED: webloginCarloan

- As you can see here, methods have got executed in alphabetic order, though they are placed in the code in different manner.

===============================================

Default test

Tests run: 3, Failures: 0, Skips: 0

===============================================

===============================================

Default suite

Total tests run: 3, Failures: 0, Skips: 0

===============================================

**Now let’s run with depends attribute**

**package** tutorial;

**import** org.testng.annotations.Test;

**publicclass** Day2 {

@Test

**publicvoid** webloginCarloan()

{

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

}

@Test

**publicvoid** mobileloginCarloan()

{

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

}

@Test(dependsOnMethods={"webloginCarloan"})

**publicvoid** APICarloan()

{

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

}

}

[TestNG] Running:

C:\Users\chitta\AppData\Local\Temp\testng-eclipse-149977187\testng-customsuite.xml

mobileloginCarloan

WebloginCarloan

loginAPICarloan

PASSED: mobileloginCarloan

PASSED: webloginCarloan

PASSED: APICarloan

- As you can see it here APICarloan() has been run after webloginCarloan() only. So this is the functionality of depends.

===============================================

Default test

Tests run: 3, Failures: 0, Skips: 0

===============================================

===============================================

Default suite

Total tests run: 3, Failures: 0, Skips: 0

===============================================

**Suppose we want to wait for 2 methods to complete their execution now. Like below**

@Test(dependsOnMethods={"webloginCarloan","mobileloginCarloan"})

**publicvoid** APICarloan()

{

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

}

**Helper attribute: Enabled: -**

Suppose in our entire test suite we know that; certain flow is not working. And we want to ignore that. Then “enabled” will be used to skip that method. Like below.

**package** tutorial;

**import** org.testng.annotations.Test;

**publicclass** Day2 {

@Test(enabled=**false**)

**publicvoid** webloginCarloan()

{

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

}

@Test

**publicvoid** mobileloginCarloan()

{

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

}

@Test(dependsOnMethods={"mobileloginCarloan"})

**publicvoid** APICarloan()

{

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

}

}

[TestNG] Running:

mobileloginCarloan

loginAPICarloan

PASSED: mobileloginCarloan

PASSED: APICarloan

- As you can see webloginCarloan()method has been ignored now. You can enable it by using enabled=true.

===============================================

Default test

Tests run: 2, Failures: 0, Skips: 0

===============================================

===============================================

Default suite

Total tests run: 2, Failures: 0, Skips: 0

One of your testcases is taking more than usual.

**package** tutorial;

**import** org.testng.annotations.Test;

**publicclass** Day2 {

@Test(enabled=**true**)

**publicvoid** webloginCarloan()

{

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

}

@Test(timeOut=4000)//This will wait for 4 sec before failing

**publicvoid** mobileloginCarloan()

{

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

}

@Test(dependsOnMethods={"mobileloginCarloan"})

**publicvoid** APICarloan()

{

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

}

}

**Parameterizing in TestNG: -**

<?xmlversion=*"1.0"*encoding=*"UTF-8"*?>

<!DOCTYPEsuiteSYSTEM"http://testng.org/testng-1.0.dtd">

<suitename=*"Loan Department"*parallel=*"none"*>

<parametername=*"URL"*value=*"https://www.google.co.in/"*></parameter>

<testname=*"Personal Loan"*>

<classes>

<classname=*"tutorial.Day1"*/>

</classes>

</test><!-- Test -->

<testname=*"Car Loan"*>

<classes>

<classname=*"tutorial.Day2"*/>

</classes>

</test><!-- Test -->

<testname=*"Home Loan"*>

<classes>

<classname=*"tutorial.Day3"*/>

</classes>

</test><!-- Test -->

</suite><!-- Suite -->

**package** tutorial;

**import** org.testng.annotations.Parameters;

**import** org.testng.annotations.Test;

**publicclass** Day1 {

@Parameters({"URL"}) //This is applicable only for demo() method

@Test

**public void** demo(String urlName)

{

System.***out***.println("URL is : "+urlName);

}

@Test

**publicvoid** secondTest()

{

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

}

}

[TestNG] Running:

D:\JAVA\_SELENIUM\_WORKSPACE\TestNGPractice\testng.xml

URL is : <https://www.google.co.in/>//As you can see it is printed here

Bye

mobileloginCarloan

WebloginCarloan

loginAPICarloan

loginAPIHomeloan

mobileloginHomeloan

WebloginHomeloan

===============================================

Loan Department

Total tests run: 8, Failures: 0, Skips: 0

===============================================

We can put <parametername=*"URL"*value=*"https://www.google.co.in/"*></parameter>in suite level also and in test level also. That is called suite level & test level parameterization respectively.

**Data Provider Annotation – Parameterizing testcases: -**

How to pass multiple values from testing.xml file? Now let me write all my classes (Office)

**package** tutorial;

**import** org.testng.annotations.Parameters;

**import** org.testng.annotations.Test;

**publicclass**Day1 {

@Parameters({"URL","APIKey/username"}) //This is applicable only for demo() method

@Test

**publicvoid** demo(String urlName,String key )

{

System.***out***.println("URL is : "+urlName);

System.***out***.println("Key is : "+key);

}

@Test

**publicvoid** secondTest()

{

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

}

}

**package** tutorial;

**import** org.testng.annotations.Test;

**publicclass**Day2 {

@Test(enabled=**true**)

**publicvoid** webloginCarloan()

{

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

}

@Test(timeOut=4000)

**publicvoid** mobileloginCarloan()

{

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

}

@Test(dependsOnMethods={"mobileloginCarloan"})

**publicvoid** APICarloan()

{

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

}

}

**package** tutorial;

**import** org.testng.annotations.Test;

**publicclass** Day3 {

@Test

**publicvoid** webloginHomeloan()

{

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

}

@Test

**publicvoid** mobileloginHomeloan()

{

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

}

@Test

**publicvoid** loginAPIHomeloan()

{

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

}

}

<?xmlversion=*"1.0"*encoding=*"UTF-8"*?>

<!DOCTYPEsuiteSYSTEM"http://testng.org/testng-1.0.dtd">

<suitename=*"Loan Department"*parallel=*"none"*>

<parametername=*"URL"*value=*"https://www.google.co.in/"*></parameter>

<parametername=*"APIKey/username"*value=*"123456"*></parameter>

<testname=*"Personal Loan"*>

<classes>

<classname=*"tutorial.Day1"*/>

</classes>

</test><!-- Test -->

<testname=*"Car Loan"*>

<classes>

<classname=*"tutorial.Day2"*/>

</classes>

</test><!-- Test -->

<testname=*"Home Loan"*>

<classes>

<classname=*"tutorial.Day3"*/>

</classes>

</test><!-- Test -->

</suite><!-- Suite -->

[TestNG] Running:

D:\JAVA\_SELENIUM\_WORKSPACE\TestNGPractice\testng.xml

URL is : https://www.google.co.in/

Key is : 123456

Bye

mobileloginCarloan

WebloginCarloan

loginAPICarloan

loginAPIHomeloan

mobileloginHomeloan

WebloginHomeloan

===============================================

Loan Department

Total tests run: 8, Failures: 0, Skips: 0

Till now we have seen folder level, but there will be some parameters which we need in test/method level.

**Method/Test level Annotation: - or Data Provider Annotation: -**

Global environment variables can be incorporated through Selenium. Parameterizing with multiple data sets by running tests with multiple combination. Let’s implement this in below method of Day2 class.

@Test(timeOut=4000)

**publicvoid**mobileloginCarloan()

{

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

}

If you are running the class only (not through .xml), then if there is any dependency (parameters passed from .xml), then only class run will fail.

Amongst the above classes and the xml file, we have only changed the Day2 class.

**package** tutorial;

**import** org.testng.annotations.DataProvider;

**import** org.testng.annotations.Test;

**publicclass** Day2 {

@Test(enabled=**true**)

**publicvoid** webloginCarloan()

{

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

}

@Test(dataProvider="getData")

**publicvoid** mobileloginCarloan(String username,String password)

{

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

System.***out***.println(username+" "+password);

}

@Test(dependsOnMethods={"mobileloginCarloan"})

**publicvoid** APICarloan()

{

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

}

@DataProvider

**public** Object[][] getData()

{

//1st combination - With a username & password - good credit history

//2nd combination - with another username & password - No credit history

//3rd - Fraudulent credit history

//Now without writing the test case/method 3 times , we can use it 3 times

Object[][] data=**new** Object[3][2]; //defined multi-dimentional object array

//Here it will 3 rows(3 combinations) & 2 columns(username & password)

//1st set

data[0][0]="1stchitta";

data[0][1]="1stpassword";

//2nd set

data[1][0]="2ndchitta";

data[1][1]="2ndpassword";

//3rd set

data[2][0]="3rdchitta";

data[2][1]="3rdpassword";

**return**data;

}

}

**TestNG Listeners: -**

Scenario: You must take a screenshot in every test failure.

Amongst the above classes and the xml file, we have only added Listeners class and changed the testing.xml file. So am pasting the same below.

**package** tutorial;

**import** org.testng.ITestContext;

**import** org.testng.ITestListener;

**import** org.testng.ITestResult;

//There is an interface called ITestLister, which implements TestNG

**publicclass** Listeners **implements** ITestListener {

@Override

**publicvoid** onTestStart(ITestResult result) {

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

}

@Override

**publicvoid** onTestSuccess(ITestResult result) {

System.***out***.println("Listeners : Script is successful !!!");

}

@Override

**publicvoid** onTestFailure(ITestResult result) {

System.***out***.println("Script is Failed !!!"+result.getName());

//Writedown screenshot code here

//result.getName() - Will give which test case is failed

}

@Override

**publicvoid** onTestSkipped(ITestResult result) {

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

}

@Override

**publicvoid** onTestFailedButWithinSuccessPercentage(ITestResult result) {

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

}

@Override

**publicvoid** onStart(ITestContext context) {

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

}

@Override

**publicvoid** onFinish(ITestContext context) {

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

}

}

<?xmlversion=*"1.0"*encoding=*"UTF-8"*?>

<!DOCTYPEsuiteSYSTEM"http://testng.org/testng-1.0.dtd">

<suitename=*"Loan Department"*parallel=*"none"*>

<listeners>

<listenerclass-name=*"tutorial.Listeners"*></listener>

</listeners>

<parametername=*"URL"*value=*"https://www.google.co.in/"*></parameter>

<parametername=*"APIKey/username"*value=*"123456"*></parameter>

<testname=*"Personal Loan"*>

<classes>

<classname=*"tutorial.Day1"*/>

</classes>

</test><!-- Test -->

<testname=*"Car Loan"*>

<classes>

<classname=*"tutorial.Day2"*/>

</classes>

</test><!-- Test -->

<testname=*"Home Loan"*>

<classes>

<classname=*"tutorial.Day3"*/>

</classes>

</test><!-- Test -->

</suite><!-- Suite -->

[TestNG] Running:

D:\JAVA\_SELENIUM\_WORKSPACE\TestNGPractice\testng.xml

URL is : https://www.google.co.in/

Key is : 123456

Listeners : Script is successful !!!

Bye

Listeners : Script is successful !!!

mobileloginCarloan

1stchitta 1stpassword

Listeners : Script is successful !!!

mobileloginCarloan

2ndchitta 2ndpassword

Listeners : Script is successful !!!

mobileloginCarloan

3rdchitta 3rdpassword

Listeners : Script is successful !!!

WebloginCarloan

Listeners : Script is successful !!!

loginAPICarloan

Listeners : Script is successful !!!

loginAPIHomeloan

Listeners : Script is successful !!!

mobileloginHomeloan

Listeners : Script is successful !!!

WebloginHomeloan

Listeners : Script is successful !!!

===============================================

Loan Department

Total tests run: 10, Failures: 0, Skips: 0

**Running Tests in parallel and generating Reports: -**

Previously we are running the tests/method sequentially as mentioned in testing.xml. But we want to run it parallelly.

If we want to run Personal Loan& Car Loan method together, then put parallel="tests"thread-count="2" in suite.

thread-count will give how many tests will run in parallel.

<?xmlversion=*"1.0"*encoding=*"UTF-8"*?>

<!DOCTYPEsuiteSYSTEM"http://testng.org/testng-1.0.dtd">

<suitename=*"Loan Department"*parallel=*"tests"*thread-count=*"2"*>

<listeners>

<listenerclass-name=*"tutorial.Listeners"*></listener>

</listeners>

<parametername=*"URL"*value=*"https://www.google.co.in/"*></parameter>

<parametername=*"APIKey/username"*value=*"123456"*></parameter>

<testname=*"Personal Loan"*parallel=*"classes"*thread-count=*"2"*>

<classes>

<classname=*"tutorial.Day1"*/>

</classes>

</test><!-- Test -->

<testname=*"Car Loan"*>

<classes>

<classname=*"tutorial.Day2"*/>

</classes>

</test><!-- Test -->

<testname=*"Home Loan"*>

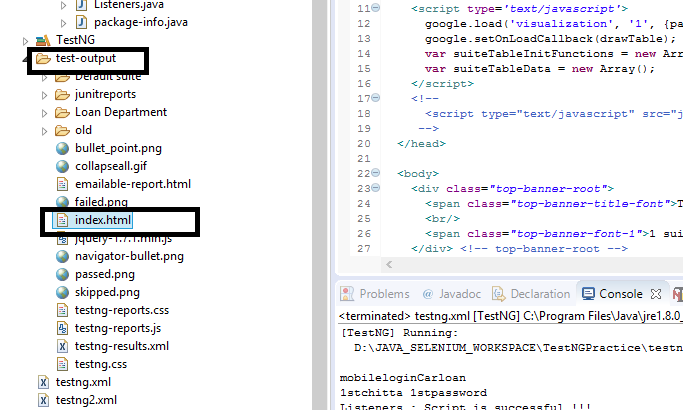
<classes>

<classname=*"tutorial.Day3"*/>

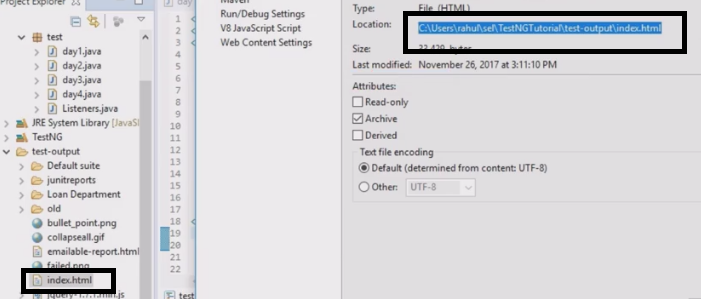
</classes>

</test><!-- Test -->

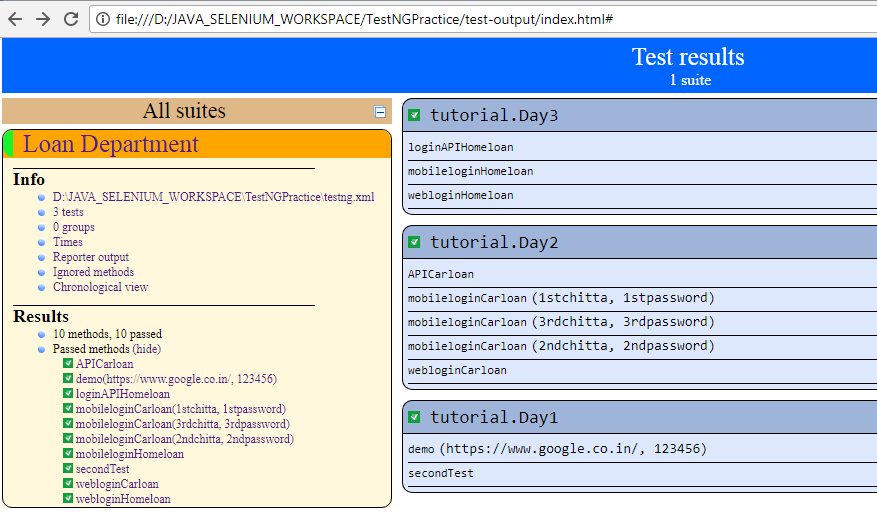
</suite><!-- Suite -->



Right click on index and go to properties. Take the link



Open that link, it will look like below



**Framework Part-2 Data driven framework & Excel API: -**

**Why we should not hardcode the data: -**

**Rough: -**

* **Though Properties class we can declare global variable – it will interpret**
* **Datadriven.properties – here we can define username & password – in key=value manner. Define it using fileinputstream, define the properties file here. – it will load**
* **prop.load(fis)**
* **prop.getproperty(keyname) – it will give value**
* **if(prop.getproperty(“keyname”)).contains(“xyz”) then do below/open this browser – contains/equals**
* **public Webdriver driver = null; - has to be defined at the top. Else it will not able to change outside of the if block when we will use it for locating.**
* **Now we can put this above code in a class called login and can use it as parent class when we wnt to use it and call the function as and when it is needed.**
* **How to connect t Excel – download and add POI JARs.**
* **Fileinputstream fis – put the path of the xls file**
* **Workbook > excel > sheet>row> columns>grab the value**
* **XSSFWorkbook wb = new XSSWorkbook(fis)**
* **XSSFSheet sheet = wb.getSheet(“script”)**
* **XSSFRow row = Sheet.getrow(2);**
* **XSSFCell cell = Row.getcell(2);**
* **String val = Cell.getstringcellvalue();**
* **Same way we can also dp setcelldata by using below**
* **Cell.setcellvalue(“hyy”);**

**Maven: -** Software project Management tool

Maven–

Apache Maven is a software project management and build management tool for Java Frameworks.

1. Why Maven?

* Central repository to get dependencies - <https://mvnrepository.com/>

Selenium java - <https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>

* Maintaining common structure across the organization

<https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html>

* Flexibility in Integrating with CI tools
* Plugins for Test framework execution.

1. Install Maven
2. Set System variables to recognize Maven
3. Understanding Maven terminologies

**Artifact**:An artifact is a file, usually a JAR, that gets deployed to a Maven repository.

**GroupId**: groupId will identify your project uniquely across all projects,

**archetype:generate**; Generates a new project from an archetype

1. Creating Maven Project

mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

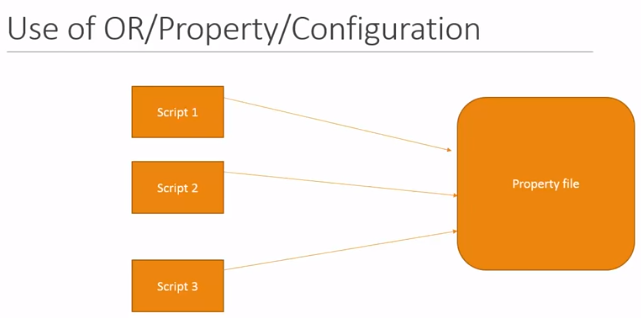
1. Integrate Maven with Eclipse
2. Maven Phases for Test Automation
3. Clean, compile, test
4. Understanding POM.xml file dependencies..
5. How to find the dependencies of the Softwares?
6. Importance of Maven Sure fire plugin
7. Running Tests with Surefireplugin
8. Integrating Testng into Maven
9. Testng xml files configuration in POM file
10. Setting up Profiles in POM.xml file
11. Running selected Tests only with Maven commands
12. Maven Sure fire reports

**Object Repository in SELENIUM:-**

1. What is Object Repository(OR)/ Properties/ Configuration file – all are same
2. What is the use of it
3. Program to read the property file
4. Create library for reuse
5. Use property file in selenium script

What is Object Repository(OR)/ Properties/ Configuration file – all are same

Ans – It’s a plain text file which contains key & value pair. Using keys , we can get the values respectively. You can use this as configuration file. In selenium you can use this as Object Repository as well.



Created on – 22/07/2018

Revision Completed on – 13/04/2020