JAVA –

**Strings**

**Type Casting (Wrapper Classes)**

***Threads(Optionally)***

**File Handling**

**Exceptions**

**Collections**

***Looping and conditional statements***

**SELENIUM *(Browser Automation Tool)* :**

1. **Open Source free available software**
2. **Almost all the popular browsers //QTP (HP) – IE, Mozilla, Chrome**
3. **Platform independed // QTP (HP) –Platform dependent**
4. **Supports multiple Programming languages // QTP (HP) – VBS**
   1. **Java**
   2. **C#**
   3. **Ruby**
   4. **Python**
   5. **Perl**
   6. **Javascript**
   7. **PHP**
5. **No dedicated machine is required for execution // QTP (HP) – Dedicated machine is must**
6. **Parallel Execution( 2-n ) // QTP (HP) – not supported**
7. **Distributed Execution**

***Versions :***

1. ***Selenium-IDE (Record and Playback) used by Beginners***
   1. ***Same origin Policy***

AUT

Script

1. ***~~Selenium-RC~~***
   1. ***~~Virtual Server~~***

VS

AUT

Script

* 1. ***~~Execution was very slow~~***
  2. ***~~Proxy settings, then RC will fail to perform~~***
  3. ***~~SSL, Security~~***
  4. ***~~Client and Server config was very difficult~~***
  5. ***~~JavaScript~~***

1. ***Selenium Webdriver (2.0 , 3.0, (3.141.59), Selenium 4.0 alpha)***
   1. ***JAVA***
2. ***Selenium Grid – Distributed Execution***

***TestNG, Maven, Jenkins:***

***Official WebSite :***

<https://www.seleniumhq.org/>

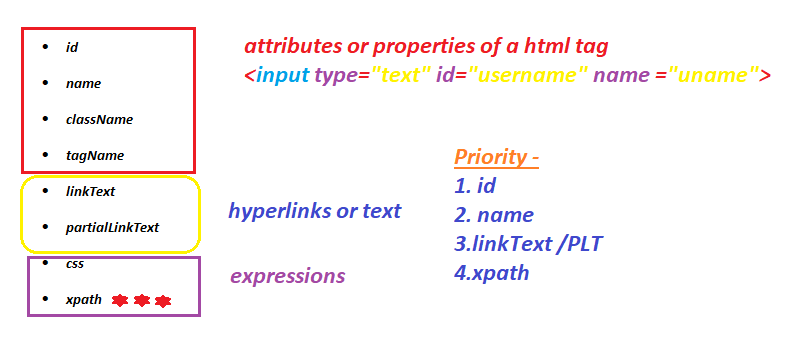
***Selenium IDE :***

***Addon with FF or Chrome***

***Script Generated by Selenium IDE is called Selenese***

***Selenese***

* ***Command - type of Action that we perform on WebElement (any component which you see on a Web Page)***
  + ***Command***
    - ***open – launch the application***
    - ***type***
    - ***click***
    - ***click at***
    - ***assert title***
    - ***assert text***
* ***Target- WebElement on we are performing action***
  + ***id***
  + ***name***
  + ***className***
  + ***tagName***
  + ***linkText***
  + ***partialLinkText***
  + ***css***
  + ***xpath***

******

* ***Value***
  + ***Optional Section , text to be typed***

***Validations in Selenium IDE:***

* ***Verify - will make the step fail and continue the execution***
* ***Assert - will make the step fail and stop the execution***

***POC – Proof of Concept***

***Asusual***

***nd 6.30 AM IST 🡪(9.30 PM CST)***

***Expressions :***

***We find the exact location of a web Element in the web page or DOM.***

***CSS - we can traverse only from parent tag to child tag in HTML DOM***

1. ***css=htmlTag[propertyname = ‘propertyValue’]***
2. ***css=htmlTag#idValue or #idValue***
3. ***css=htmlTag. valueOfClassAttribute or .valueOfClassAttribute***

***Xpath - we can traverse from parent tag to child tag in HTML DOM or from child tag to parent tag***

***Xml path🡺 location of a element in DOM***

1. ***Absolute xpath***

***/html/child/superchild/child/../.../.../...***

1. ***Relative xpath***

***// 🡺 anywhere in the html page***

***Syntax :***

1. ***Basic Xpath***

***//htmlTag[@propertyname=’propertyValue’]***

1. ***Traversing from parent to child***

***//htmlTag[@propertyname=’propertyvalue’]/childTagName***

***Or***

***//htmlTag[@propertyname=’propertyvalue’]//childTagName***

***Or***

***//htmlTag[@propertyname=’propertyvalue’]//childTagName[@propertyname=’propertyvalue’]***

1. ***Using regular Expressions in Xpath***

***If one property is not sufficient to find the element then we can go with Regular Expressions***

***3.1 AND***

***- if both the properties are matched then we get the element***

***//htmltag[@property1=’value1’ and @property2=’value2’]***

***3.2 OR***

***- if any one property is matched then we get the element***

***//htmltag[@property1=’value1’ or @property2=’value2’]***

1. ***Xpath Using Functions:***
   1. ***text()***

***//htmltag[text()=’textToSearch’]***

* 1. ***contains(arg1,arg2)***
     1. ***arg1 - > text function or attribute***
     2. ***arg2 -> partial value (partial text or partial attibute value)***

***//htmltag[conatins(text(),’partialtext’)]***

***//htmltag[contains(@attriburename,’partialValueOfAttribute’)]***

* 1. ***starts-with(arg1,arg2)***
     1. ***arg1 - > text function or attribute***
     2. ***arg2 -> partial value (partial text or partial attibute value)***

1. ***Dependent and Independent Elements (Navigating from child to parent)***

When ever we are identifying or playing with dependent and independent elements always write a xpath to independent element from there navigate to dependent element

1. Xpath of child
2. //parentHtmlTAG[ Xpath of child ]
3. ***Using axes Functions :***

6.1 following-sibling🡪 Will search all the sibling tags which is next to the current tag

Xpath of independentElement/**following-sibling::**SiblingTag

6.2 preceding-sibling🡪 Will search all the sibling tags which are previous to the current tag

Xpath of independentElement/**preceding-sibling::**SiblingTag

6.3 following🡪 Will search all the tags which is next to the current tag till the end of the page

Xpath of independentElement/**following::**SiblingTag

6.4 preceding🡪 Will search all the tags which is next to the current tag till the start of the page

Xpath of independentElement/**preceding::**SiblingTag

6.5 parent🡪 will navigate to parent tag from the current tag

//th[text()='Directed by']/parent::tr

6.6 child 🡪 will navigate to child tag from the current tag

//th[text()='Directed by']/parent::tr/child::td

6.7 ancestor🡪 will navigate to parent -> super parent-> super most parent....

|  |  |  |  |
| --- | --- | --- | --- |
| ***Application*** | ***Type*** | ***Element Name*** | ***Xpath*** |
| ***ActiTime*** | ***basic*** | ***UserName Text Box*** | ***//input[@placeholder='Username']***  ***//input[@id='username']***  ***//input[@name='username']*** |
| ***ActiTime*** | ***basic*** | ***Password Text Box*** | ***//input[@placeholder='Password']*** |
| ***ActiTime*** | ***Traversing Parent to immediate child*** | ***Login Button*** | ***//a[@id='loginButton']/div***  ***Or***  ***//div[text()='Login ']*** |
| ***ActiTime*** | ***Traversing Parent to child present somewhere*** | ***Search all div under login button*** | ***//td[@id='loginButtonContainer']//div*** |
| ***ActiTime*** | ***Traversing Parent to child present somewhere*** | ***Print All Module Names*** | ***//table[@id='topnav']//div[@class='label']*** |
| ***ActiTime*** | ***Using Regular Expression*** | ***Search both username and password*** | ***//input[@id='username' or @name='pwd']*** |
| ***ActiTime*** | ***Using Regular Expression*** | ***Search none of un and pwd*** | ***//input[@id='username' and @name='pwd']*** |
| ***ActiTime*** | ***Using Function*** | ***logo*** | ***//img[contains(@src,'timer')]*** |
| ***Actitime*** | ***Using contains Function*** | ***Search all Log*** | ***//div[contains(text(),'Log')]*** |
| ***google*** | ***Using starts-with function*** | ***Search all elements with IBM*** | ***//h3[starts-with(text(),'IBM')]*** |
|  |  |  |  |
| ***Redbus.in*** | ***Using text() with Regular Expression*** | ***Selecting a Date*** | ***//div[@id='rb-calendar\_onward\_cal']//td[text()='29' and (@class='wd day' or @class='current day' or @class='we day')]*** |
|  |  |  | ***//label[@for='radio-button' and contains(text(),'Highest')]*** |
| ***wikipedia*** | ***Child to parent*** | ***Search Director Name*** | ***//tr[th[text()='Directed by']]/td*** |
| ***mmt*** | ***Child to parent*** | ***Price of a holiday*** | ***//div[div[div[h3[contains(text(),'Spanish Delight 2020')]]]]//p[contains(@class,'latoBold ')]*** |
|  | ***Using ancestor axes function*** | ***Price of a holiday*** | ***//h3[contains(text(),'Spanish Delight 2020')]/ancestor::div[@class='boxShadow bdr packageListing pointer packageDetailsBox']//p[contains(@class,'latoBold ')]*** |
| ***gsmarena*** | ***Child to parent*** | ***Display features of any given phone*** | ***//tbody[tr[th[text()='Display']]]//td[@class='nfo']*** |
| ***wikipedia*** | ***Axes function*** | ***Search Director Name*** | ***//th[text()='Directed by']/following-sibling::td*** |
|  |  | ***Print index Number*** | ***//span[text()='Awards and nominations']/preceding-sibling::span*** |
|  |  | ***Parent and child axes function*** | ***//th[text()='Directed by']/parent::tr/child::td*** |
|  |  |  |  |

***WebDriver***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Class / Interface* | *Method Name* | | *Return type* | *Arguments* | *Description* |
| WebDriver | **Get**() | | Void | String | Launch the browser |
| WebDriver | **findElement**() | | WebElement | By | To find the web element on web page. |
| WebElement | | **Sendkeys**() | Void | String | To perform type operation |
| WebElement | | **Click()** | Void | NA | To perform click operation |
| WebDriver | | **findElements** | List<WebElement> | By | To find the web element**s** on web page. |
| WebElement | | **getText()** | String | NA | Returns the text present out side the html tag |
| WebElement | | **getAttribute(String)** | String | String | Returns the Value present for a given attribute |
|  | |  |  |  |  |
|  | |  |  |  |  |
|  | |  |  |  |  |
|  | |  |  |  |  |
|  | |  |  |  |  |
|  | |  |  |  |  |

***Synchronization***

Sync Issue or Synchronization Issue ->

Selenium 🡪 findElement 🡪 250 ms / 0 sec 🡺 **NoSuchElementException**

**Implicit Wait (common for all elements)– Will not work for the elements generated due to**

* **Ajax call**
* **Java script execution**
* **Angular JS**

**Explicit Wait (for a single Element ) –**

* **Ajax Call**
* **JavaScript**
* **Angular JS**
* **Too much time a particular element is taking**

Sync Issues

FluentWait

-own method / own logic to wait

-polling time

- ignore exceptions

500ms

Static Waits – Thread.sleep(ms)

WebDriverWait

Implicit Wait

Explicit Wait

Dynamic Waits

Fluent Wait :

1. Create a Wait object
   1. *Specify what is the element*
   2. *What is the max time to wait*
   3. *What is the polling time*
   4. *Is there any exception to ignore*
2. Write your own wait logic
   1. Override apply method present in Function Interface
   2. Write you wait logic inside apply method
3. Call until function from wait object

DropDown:

1. Select tag
   1. Create an Object to Select class by passing dropdown WebElement as argument
      1. selectByIndex(int)
      2. selectByValue(Sting)
      3. selectByVisibleText(String)
      4. deSelectByIndex(int)
      5. deselectByValue(Sting)
      6. deselectByVisibleText(String)
      7. deselectAll()
      8. getAllSelectedOptions()
      9. getOptions()
      10. getFirstSelectedOption()
2. From any customized html tags -> div, table, td ...
   1. Click-> click will work

NOTE - > if we have any id starting with ext-gen<<NUMBER>> or auto-gen<<NUMBER>> don’t use

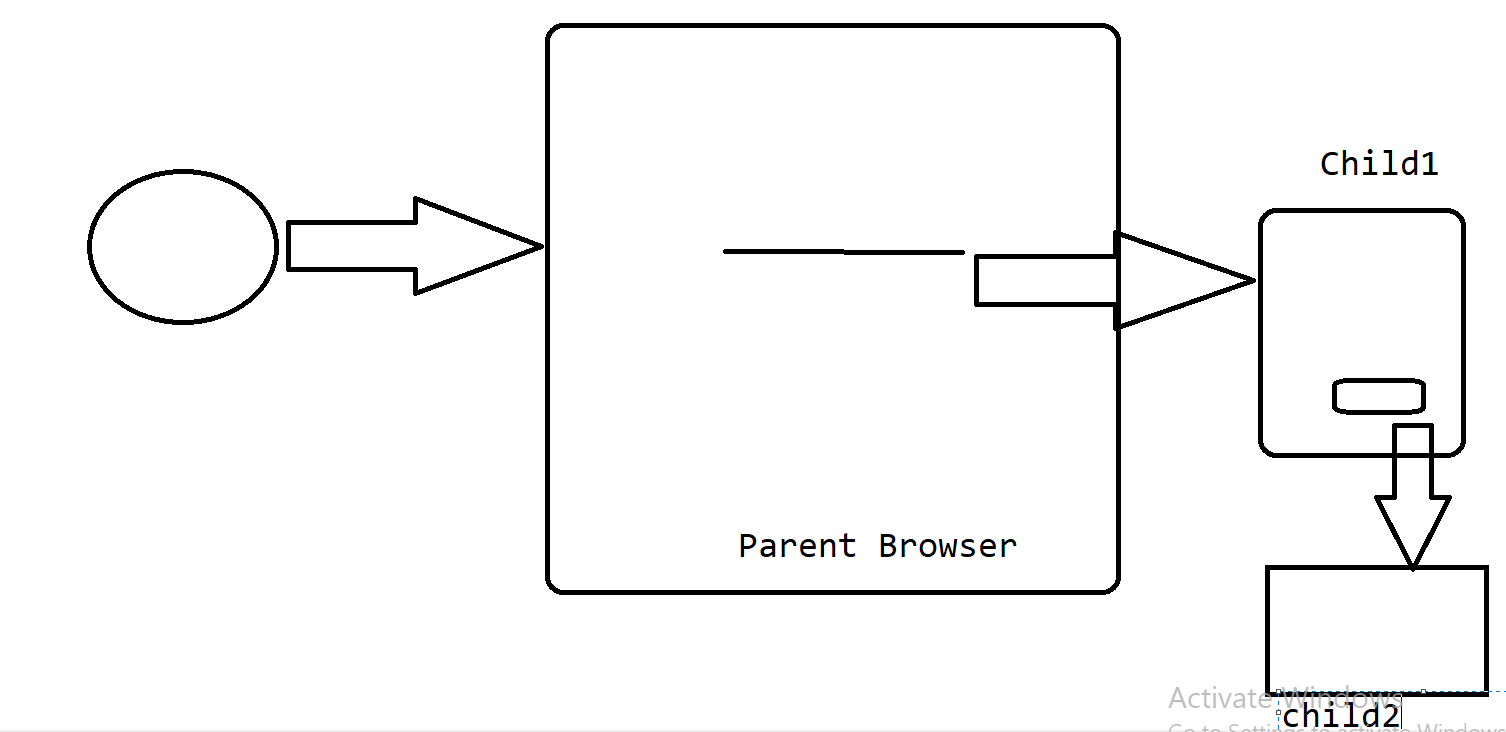
Actions in Selenium

Actions – used to perform exact keyboard and mouse operation(movment, leftclick , right click,drag and drop etc.,).

*If java script is disabled for any web Element we go with Actions. Or if we want to do some mouse movement operation.*

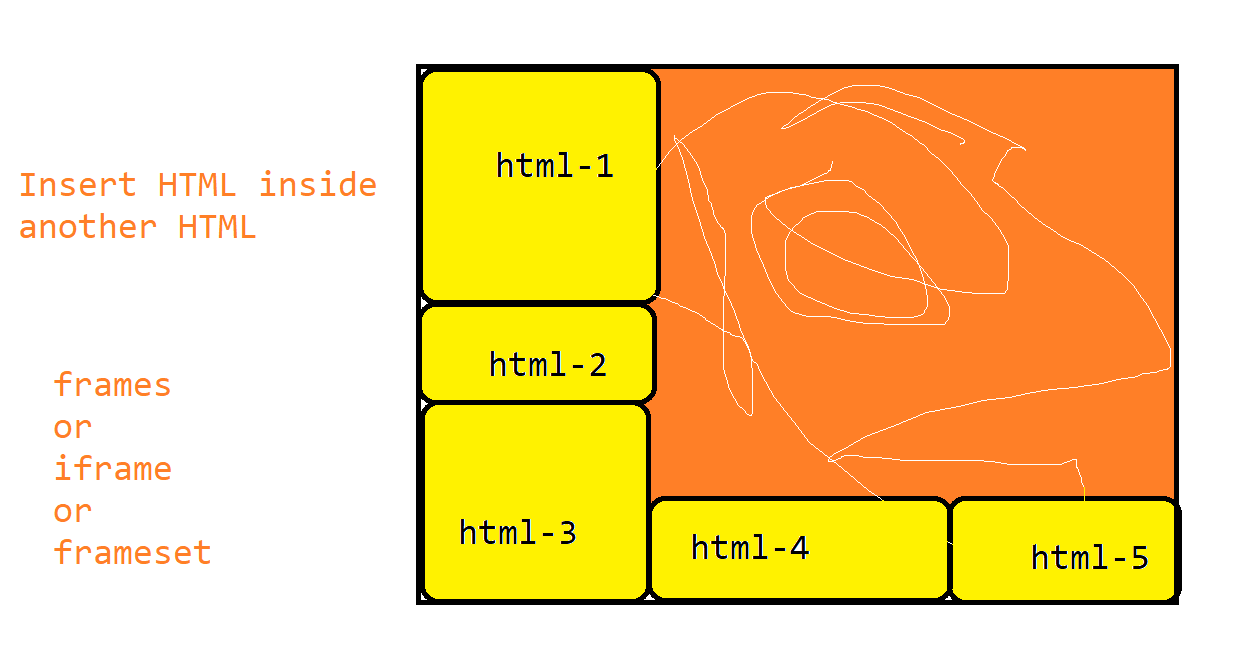
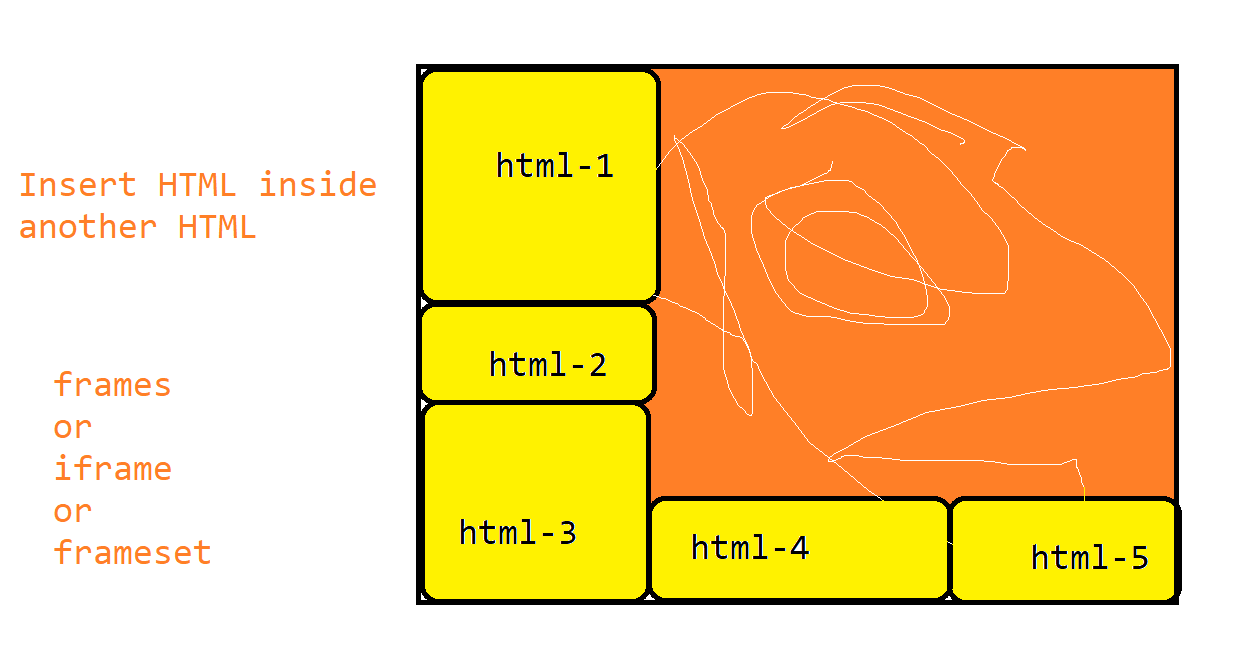
switchTo()

1. Alerts or confirmation popups – Java Script
2. Child browser (Parent and child Browsers) –
   1. getWindowHandle() - > String which is a unique ID for a Browser
   2. getWindowHandles() - > Set<String> which is a unique ID for a Browser



1. Frames
   1. Index
   2. String - name or id attribute of a iframe tag
   3. Webelement
   4. defaultContent() - > to switch the control from frame to parnet brower

NOTE - we can not switch from one frame to another directly.



Data driven Testing :

1. Difficult to maintain the Automated Tests if we hardcode the data
2. Test can execute with Max one test data

Test Data - text file, properties file, excel file, csv file, database etc...

Test Data

Local Data – excel, textfile, db, csv ...etc...

Global Data

properties file

Url, Username, Password, Data specific to testcase

browser, maxTimeout

JRE – word, pdf, excel –

POI – by apache

***Workbook<I>*** – excel file

***Sheet< I >*** – one sheet at a time

***Row***< I >– identified by its index

***Cell***< I > – represent one column

WorkbookFactory<C> - class to create object to excel file

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 ,0 | 1 | 2 | 3 | 4 | 5 |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |
| 6 |  |  |  |  |  |
| 7 |  |  |  |  |  |

**FrameWorks :**

Set of guidelines or procedure which contains many informations,

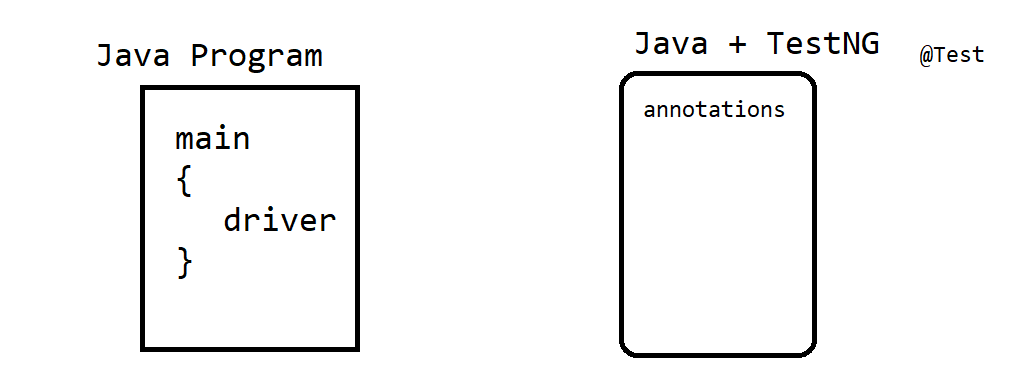
1. How the Test Cases are written
2. Where the Test cases should be stored
3. What are the coding standards to be followed
4. How the reports should be generated
5. How the tests should be executed one after the other
6. Once the test is failed- should we raise a defect automatically
7. Whom all to send a mail on completion of Test Execution

Types :

1. Function Driven Automation Framework
2. Data Driven Automation Framework
3. Key Word Driven Automation Framework
4. Hybrid Driven Automation Framework
5. TestNg
6. Page object Model

TestNG :

1. Download Test NG JAR FILES
2. Install Testng Plugin – (Optional)



* @beforeSuite->Create Reporting(HTML, PDF…)
  + @beforeClass-> Opening a browser, configuring proxy settings for module
    - @beforeTest -> only once before executing test tag ( load testdata for module )
      * @beforeMethod -> Executed before Executing every Test
        + @Test 🡪 Each Test Annotation represents Automated Test
      * @afterMethod-> Executed after Executing every test
    - @afterTest-> only once after executing test tag( clear test data for the module )
  + @afterClass-> close a browser, reset the configuring proxy settings for module
* @afterSuite->Saving the Reporting

We can keep as many Test Annotations as we want. The order of execution is based on the alphabetical of method name.

Controlling Order of Execution :

1. dependsOnMethods
2. priority

<suite guice-stage="DEVELOPMENT" name="Default suite">

<test verbose="2" name="Default test">

<classes>

<class name="testng.BeforeMethodDemo"/> -🡪JAVA PROGRAM (@TEST)

<class name="testng.BeforeMethodDemo1"/>

</classes>

</test> <!— **Module1** -->

<test verbose="2" name="Default test">

<classes>

<class name="testng.BeforeMethodDemo"/>

</classes>

</test> <!— **Module2** -->

</suite> <!-- Default suite -->

MAVEN : ( software that manage the project )

* is a ***build automation*** tool.

Build Process :

1. Download the required Dependencies
2. *Delete the previous class files*
3. *Compile the code*
4. *Unit test – Execute Unit Test Cases*
5. *Execute the functional / Regression test cases*
6. *Create jar / war file*
7. *Deploy it in the server*

Testers :

1. Managing Dependencies
2. Test Code

Installation:

1. Download maven and keep the zip file in any location
2. Unzip the file
3. Set Environment Variables
   1. MAVEN\_HOME - D:\maven\apache-maven-3.6.1
   2. M2=D:\maven\apache-maven-3.6.1\bin or % MAVEN\_HOME%\bin
   3. MAVEN\_OPTS=-Xms 256m –Xms
   4. Update PATH VARIABLE - %M2%

Create Project in eclipse :

1. File-> new-> Maven project
2. Update the settings
   1. Check compiler - and update it to the version u r using
   2. Check JRE – take the jre from jdk location

POM.xml

* Is a core or heart of maven project
* Project object model
* All information about the project

Maven Repository

* Local Repository : Repository present in your machine
* Central Repository ( Global Repository ) : <https://mvnrepository.com>

Build Life Cycle :

* Validate
* clean
* Initialize
* Compile
* Test-compile
* Test
* Package
* Install (store the jar file in m2-repo)
* Deploy

To Execute the TESTS in maven we have to use surefire plugin

### Copy the code sample from below link under Using Suite XML Files

* 1. <https://maven.apache.org/surefire/maven-surefire-plugin/examples/testng.html>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0-M4</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>testng.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

</plugins>

</build>

1. Paste it in pom.xml by creating build tag
2. Create one new testing.xml file and specify all the tests that are to be executed by maven
3. Add compiler plugin (optional)

**Page Properties**

**Page Properties**

+

**Business logic**

**Business logic**

Drawbacks for FrameWork (TestNG)

1. Test Contains both Page Properties and Business Logic
2. If we store Page Properties inside a test then there are lot of duplicate Page Properties present in multiple places ( it can be functions or it can be tests )
3. Maintenance is difficult

POM – Page object Model

1. Mainly we will have 3 packages
   1. Test – automated
      1. TestNG Annotaions
   2. Page – each page in application will have a class
      1. Variable -> webElements
      2. Methods->Functions (reusable functions / application reusable functions only for a page)
      3. Constructor -> To initialize all the elements of a Page
   3. Util - reusable functions if any
2. POM using Basic Approach while creating Page Class
   1. We will use findElement to find the element of a web Page
3. POM using Page Factory Methodology while creating Page Class
   1. We will use Annotations to find the element