**Selenium with Java**

**Duration: 3 Days**

**Pre-Requites:**

* Core Java
* Java 8 new features

**Course Objectives:**

* Upon training completion, participants will be ready to automate applications.

**Certification Link:**

* None

**Pre- Training Udemy Course Link:**

<https://www.udemy.com/course/automate-tests-using-selenium-webdriver-with-java-cucumber/>

**Post- Training Udemy Course Link:**

<https://www.udemy.com/course/selenium-real-time-examplesinterview-questions/>

**Lab Setup:**

* Windows Machine
* RAM: 8 GB
* JDK 1.8
* Eclipse
* Selenium libraries

**Course Outline:**

**Day 1:**

1. **Selenium WebDriver Components**

Introduction to Selenium WebDriver

Difference in RC and WebDriver

Features of Selenium WebDriver

Difference between WebDriver2.0 and 3.0

1. **Environment for Selenium WebDriver**

Introduction to Eclipse

Introduction to FireBug

**Practicals**: chropath

Introduction to XPath, CSSpath

**Practicals**: XPath Exercises

**Practicals**: CSSpath Exercises

**Practicals**: Security and Performance Testing using FireBug

1. **Process to Create Project and Create Selenium WebDriver Scripts**

**Practicals**: Create scripts to Automate Gmail Test Cases

**Practicals**: Create scripts to use Retrieve value from Web Site

**Practicals**: Create scripts to handle pop-up dialogs

**Practicals**: Create scripts to work with iFrames

**Day 2:**

1. **Selenium WebDriver Project – Build Selenium Java Scripts**

**Practicals**: Build script to automate Live Web site

Create scripts to emulate user interaction with Web site

* Typing text
* Taking action on an existing web element
* Observing if web element is accessible on Web site
* Working with web elements that have same name or id
* Working with web elements that have no name or id
* Navigating across web links
* Navigating across web pages
* Browsing Other sites and Returning to previous sites
* Browsing Other sites and Returning to specific site
* Create scripts to access within a Web Table

1. **Selenium WebDriver Project – Build Selenium Java Scripts**

Synchronization

* Conditional Synchronization
* Unconditional Synchronization
* Implicit and Explicit Wait

1. **Implementation of Action, Select, Random, Alert Class**

**Practicals**: Build scripts to access web elements in a Context-Driven menu (Runtime generated menu)

**Practicals**: Build scripts drag-drop (Runtime generated list of items in drop-drop list)

**Day 3:**

1. **Capture Snapshot of Web site during script Execution**

**Practicals**: Build script using Java and Selenium to capture screen

1. **DataDriven Tests using Selenium, Java, Excel**

Overview of variety of Test-Data sources

Introduction to POM Framework

**Practicals**: Reading data from Test-Data source

Build script to parse and extract data from Excel Test-Data source

Build script to parse and extract data from Text file Test-Data source

Build POM Framework

1. **Developing reusable script**

**Practicals: End-to-End Project using POM**

1. **Introduction to TestNG**
2. **Handling of multi-browser automation**

How to execute the test on different browser

Automatically downloading driver executable from WebDriverManager

Handling child browser and parent browser in selenium

1. **Working with popup**

Handling alert

Handling poupup from javascript

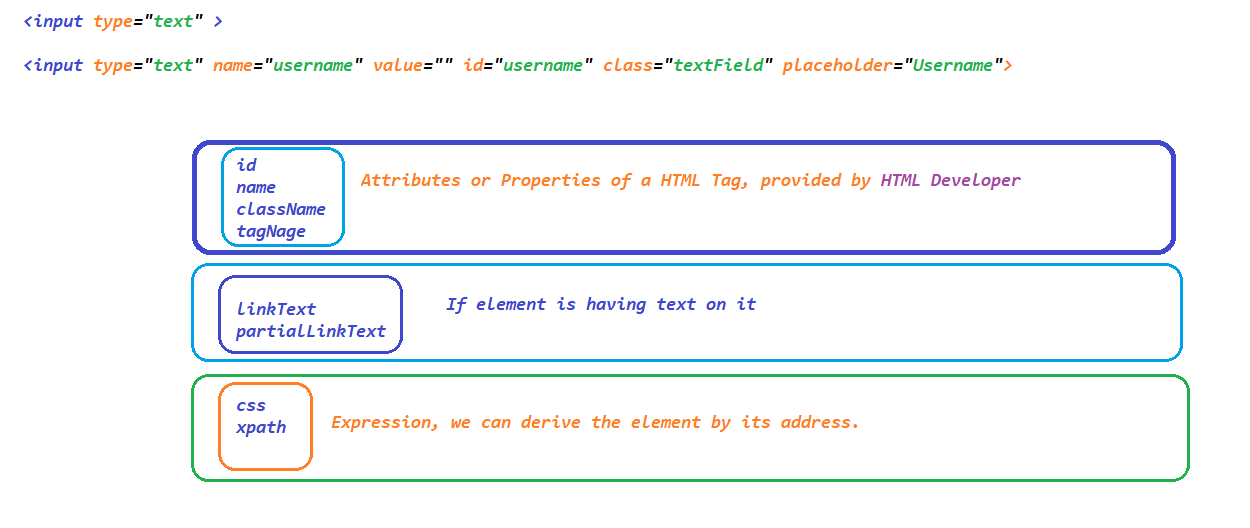
Handling authentication popup

Handling file download popup

Handling file upload popup

- Overview of AutoIT, Robot class for non-web object handling

1. Why Automation ????
2. Why Selenium????
   1. Open source and freely available software
   2. support multiple programming languages
   3. supports almost all the browsers
   4. no dedicated machine is required for Test Execution
   5. distributed execution
   6. parallel execution
   7. lot of ready made plug-in available
3. Versions of Selenium
   1. Selenium-IDE (FF, Chrome)
   2. Selenium-RC (Selenium 1.0)
   3. Selenium-WebDriver (Selenium 2.0, 3.0,4.0 beta)
   4. Selenium-Grid
4. Element identification



* 1. css
     1. **htmltag[attribute\_name=’attribute\_value’]**
        1. input[placeholder='Username']
        2. input[id='username']
     2. **htmltag#id\_value OR #id\_value**
        1. input#username OR #username
     3. **htmltag.class\_attribute\_value OR .class\_attribute\_value**
        1. input.textField OR .textField
     4. **htmltag[parent\_attribute\_name=’attribute\_value’] > child\_htmltag**
        1. a#loginButton > div
        2. td#loginButtonContainer > a >div
  2. xpath

*Use “xpath helper” from Chrome web store and “try xpath” addon from Firefox to validate the xpath*

* + 1. **Basic Xpath**
       1. **//htmltag[@attribure\_name=’attribute\_value’]**
          1. //input[@name='username']
          2. //input[@id='keepLoggedInCheckBox']
    2. **Xpath with Logical Operators**
       1. **AND**
          1. **//htmltag[@att1=’val1’ and @att2=’val2’]**

//input[@id='username' and @type='text']

//td[(@class='we day' or @class='wd day' or @class='current day') and text()='12']

* + - 1. **OR**
         1. **//htmltag[@att1=’val1’ or @att2=’val2’]**

//input[@id='username' or @type='text']

* + - 1. **NOT**
         1. **//htmltag[@att1=’val1’ and not @att2=’val2’]**

//input[@id='username' and not(@type='password')]

//td[text()='10' and not (@class='past day')]

* + 1. **Xpath with Functions**
       1. **text()**
          1. **//htmltag[text()=’exact text of an element’]**

//div[text()='Login ']

//label

//td[text()='Please identify yourself']

* + - 1. **contains(arg1, arg2)**

**arg1 -> attribute OR function**

**arg2 -> corresponding partial value**

* + - * 1. **//htmltag[contains(@attribute\_name,’partial\_att\_value’)]**

//img[contains(@src,'timer')]

* + - * 1. **//htmltag[contains(function,’partial\_text’)]**

//td[contains(text(),'iden')]

* + - 1. **starts-with(arg1,arg2)**

**arg1 -> attribute OR function**

**arg2 -> corresponding starting value**

* + - * 1. **//htmltag[starts-with(@attribute\_name,’starting\_value’)]**

//input[starts-with(@class,'textField')]

* + - * 1. **//htmltag[starts-with(function,’starting\_value’)]**

//h3[starts-with(text(),'Synechron')]

* + 1. **Xpath traverse from parent to child**
       1. **//EXPRESSION/IMMIDEATE CHILD**
          1. //a[@id='loginButton']/div
       2. **//EXPRESSION//IMMIDEATE CHILD**
          1. //td[@id='loginButtonContainer']//div[text()='Login ']
    2. **Xpath traverse from child to parent**

**When ever we are working with dependent and independent elements**

* + - 1. **//parent\_tag[xpath of independent element]**
         1. //tr[th[text()='Directed by']]//a
         2. //tr[td[div[div[text()='Task240']]]]/td[@class='selection']
    1. **Xpath with Axes Functions**
       1. **Traversing to next siblings**
          1. **//expression/following-sibling::siblingtag**

//th[text()='Directed by']/following-sibling::td/a

//li[a[@href='#Release']]/following-sibling::li

* + - 1. **Traversing to previous siblings**
         1. **//expression/preceding-sibling::siblingtag**
         2. //li[a[@href='#Release']]/preceding-sibling::li
      2. **Traversing to all next siblings till end of the page**
         1. **//expression/following::siblingtag**
         2. //li[a[@href='#Release']]/following::li
      3. **Traversing to all previous siblings till beginning of the page**
         1. **//expression/ preceding ::siblingtag**
         2. //li[a[@href='#Release']]/preceding::li
      4. **Traversing to child**
         1. **//expression/ child::childTAG**

//th[text()='Directed by']/following-sibling::td/child::a

* + - 1. **Traversing to parent**
         1. **//expression/ parent::parentTAG**

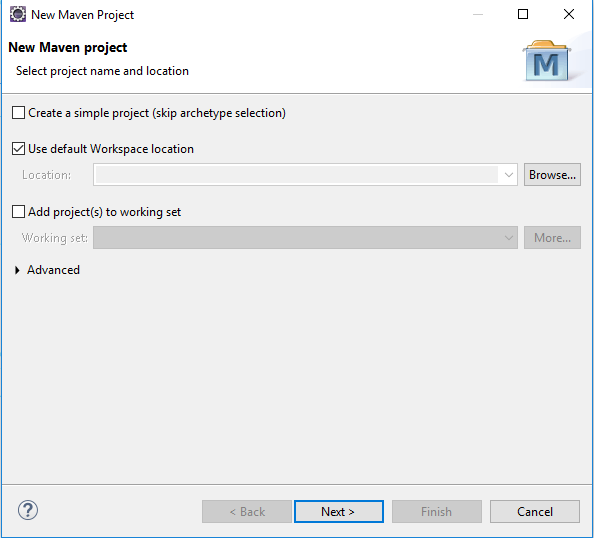
//th[text()='Directed by']/parent::tr

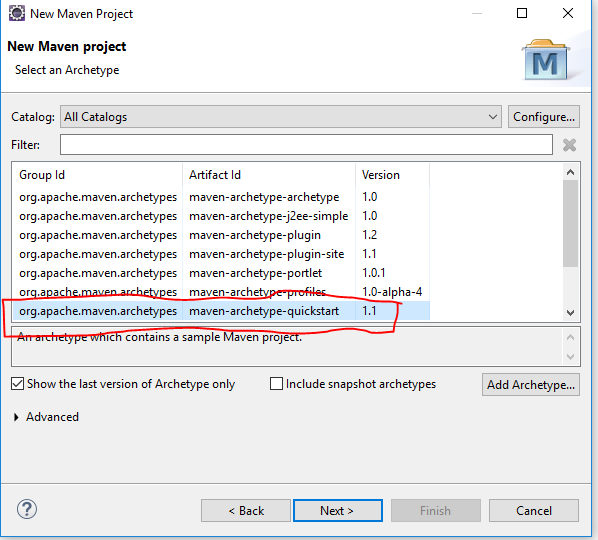
* + - 1. **Traversing to grandparent**
         1. **//expression/ancestor::ancestorTAG**

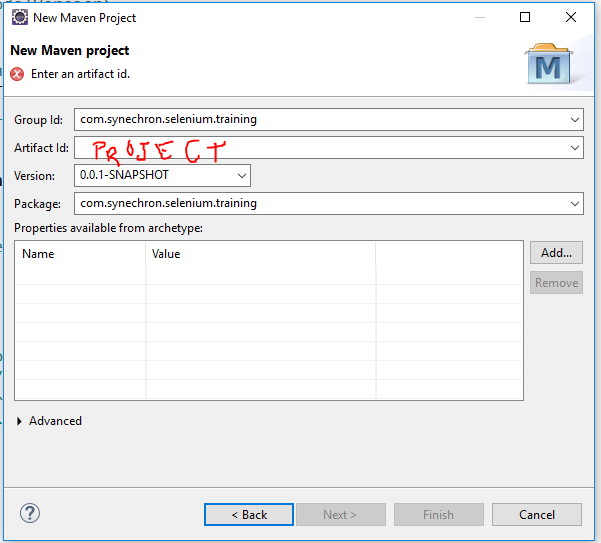
//div[text()='Task240']/ancestor::tr/td[@class='selection']

* + 1. Assignments
       1. W.X.P To find the price of a selected flight in MMT
       2. W.X.P To find the price of a selected holiday package <https://holidayz.makemytrip.com/holidays/india/search?dest=Dubai>
       3. Print display property of a selected phone
          1. DISPLAY
          2. PLATFORM
          3. BATTERY
       4. <http://automationpractice.com/index.php?id_category=9&controller=category>

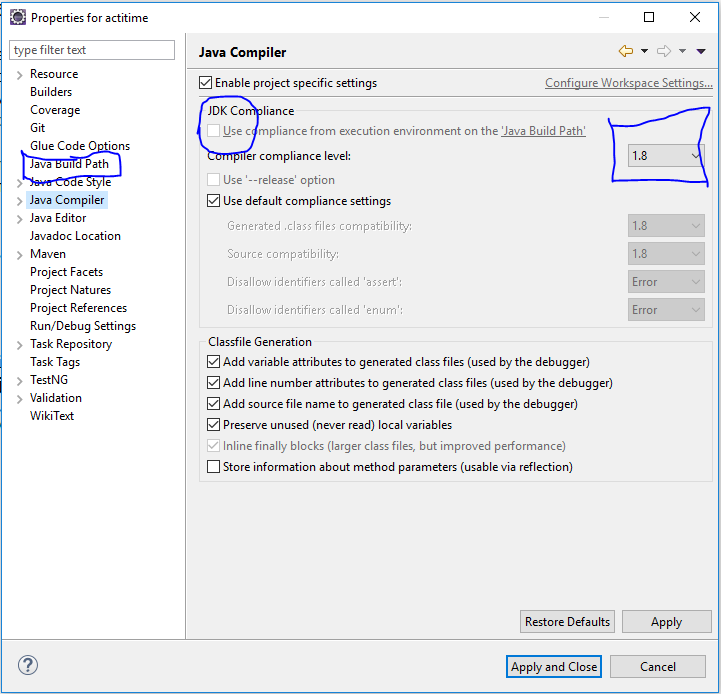
1. MAVEN PROJECT:

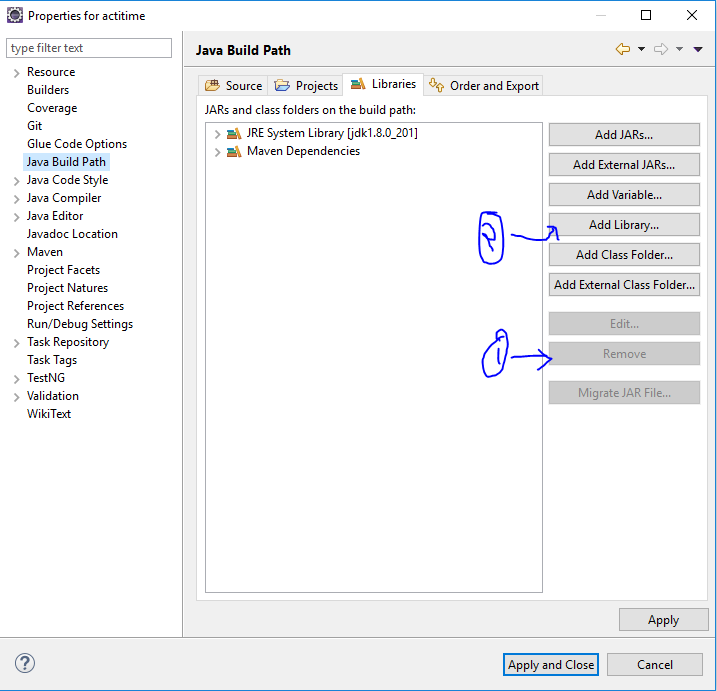


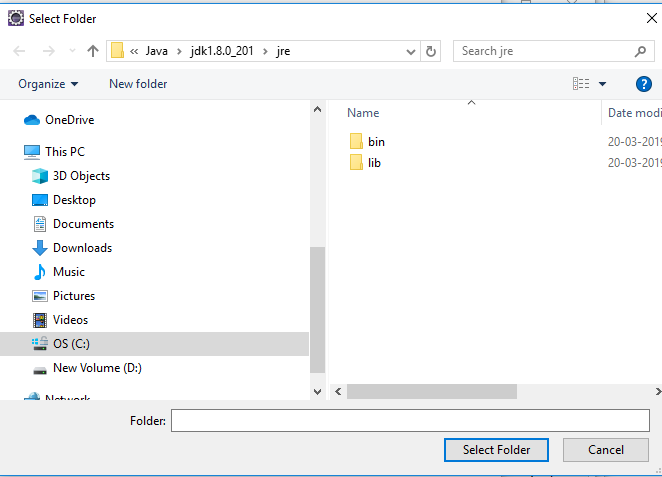




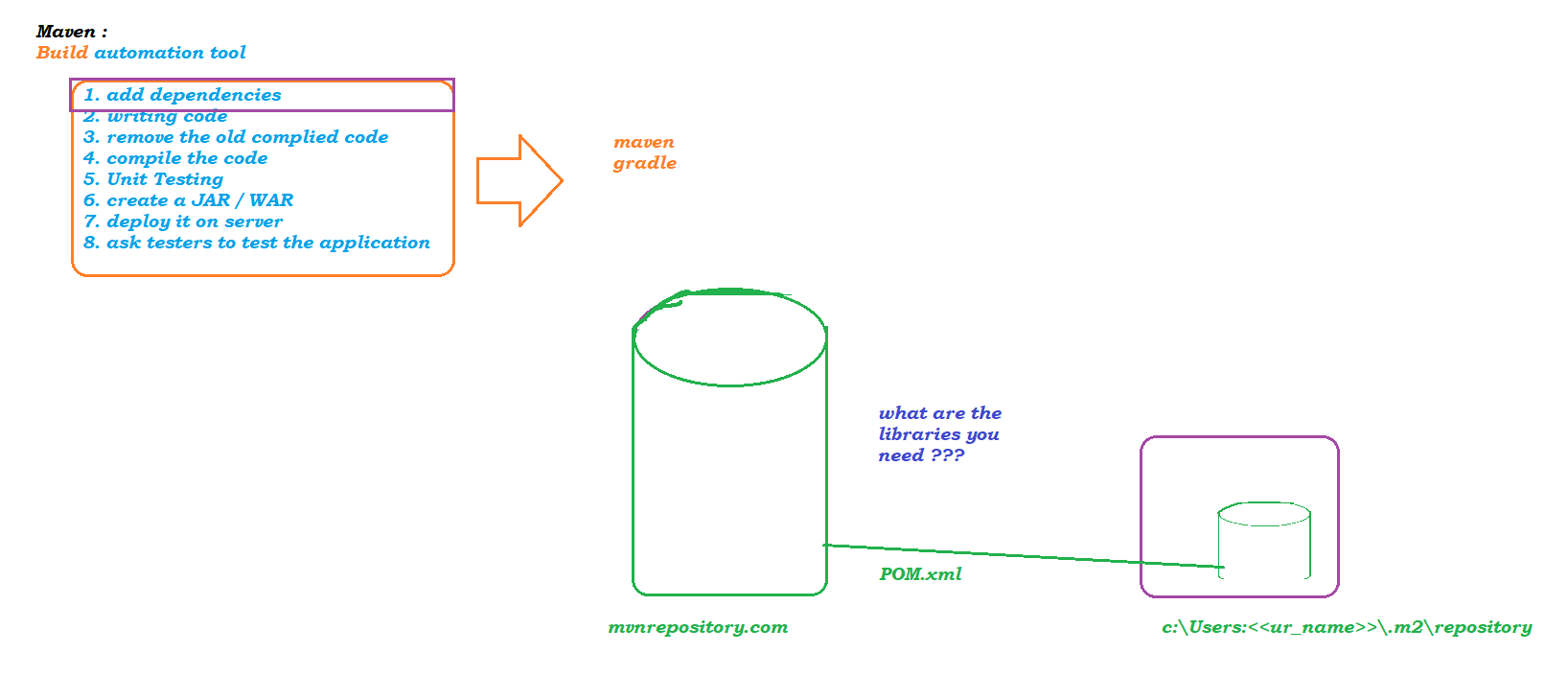
Once the project is created update Java Compiler (should be updated to the the version which is installed in the machine )and JRE (should be from JDK Location) in the project



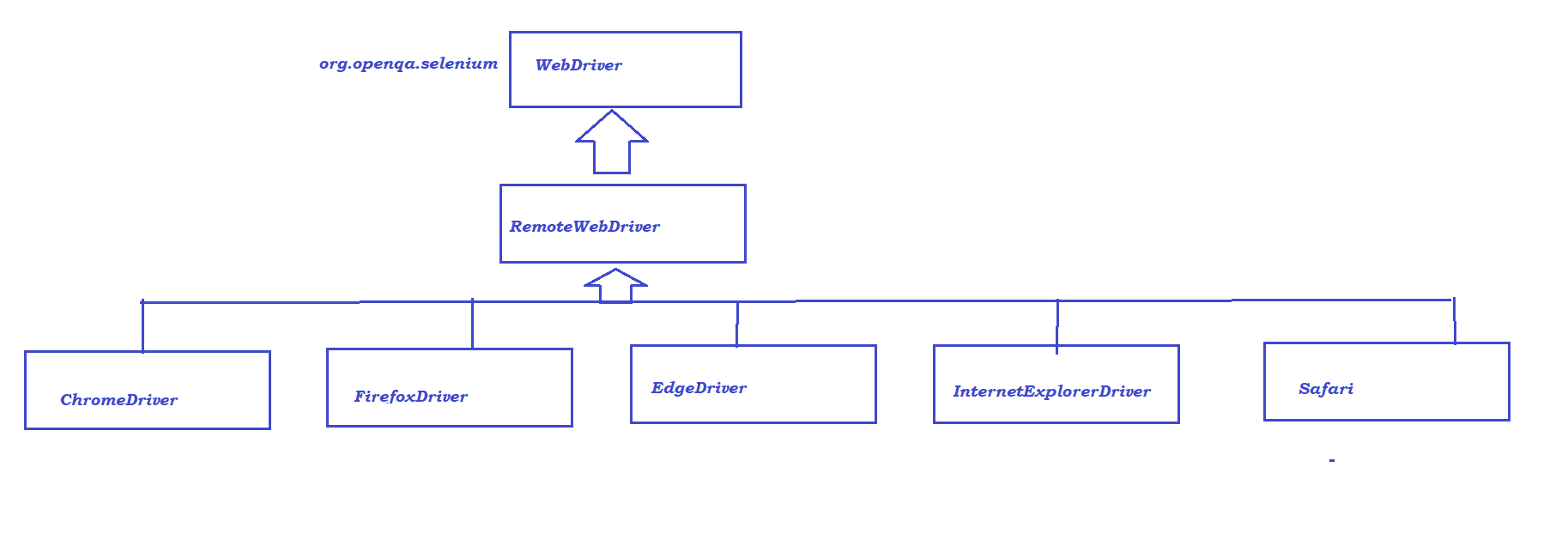


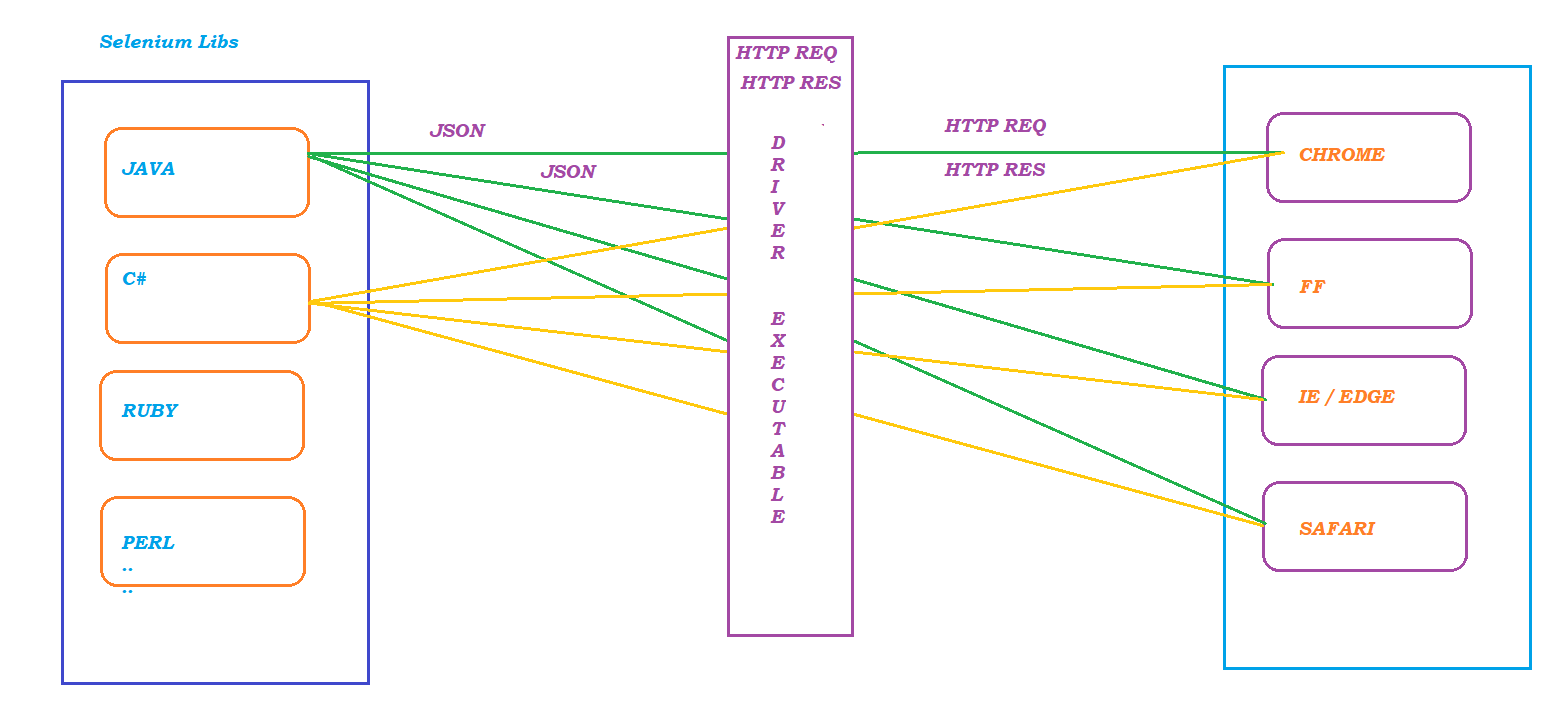


NOTE – Everything comes to maven as a plugin.



Architecture :

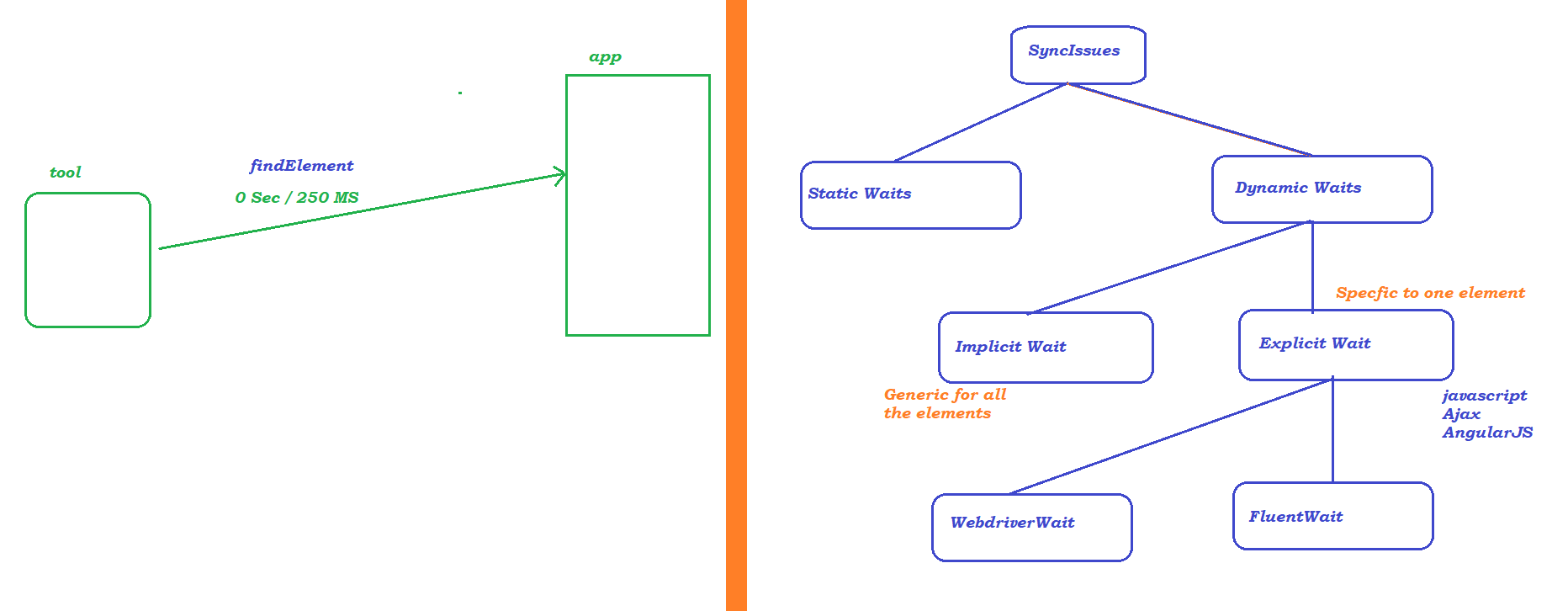




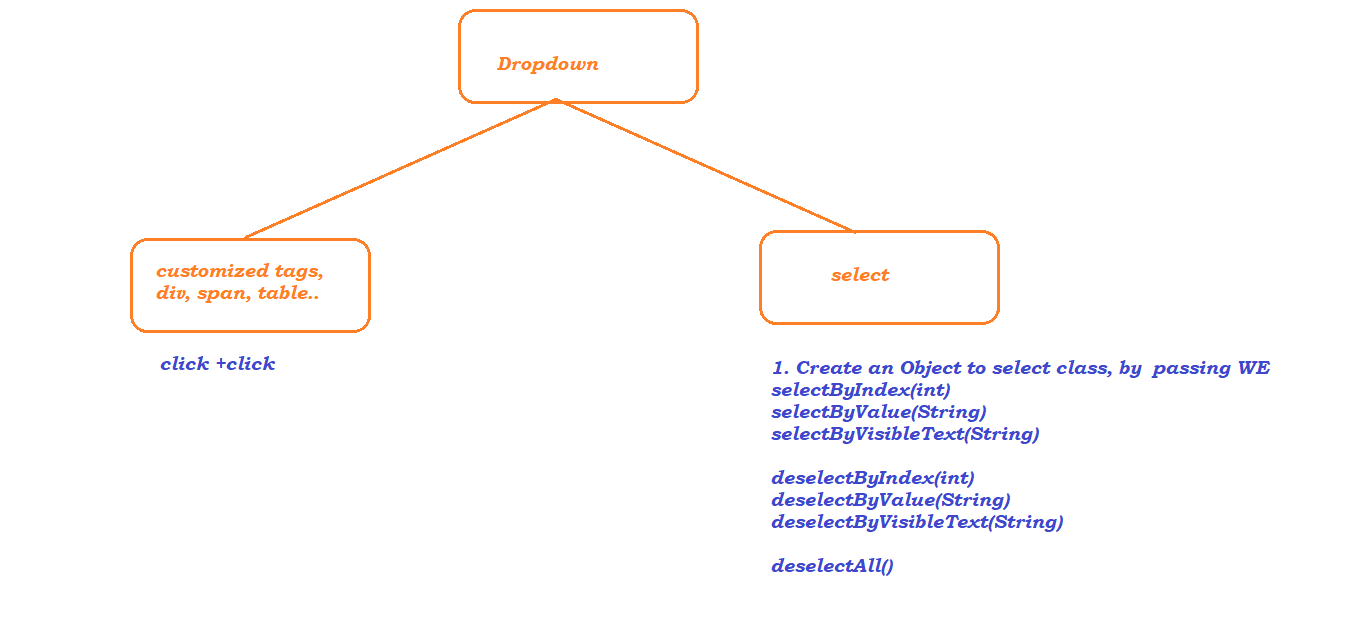
Getting Text OR Attribute in Runtime:



Sync Issues :



DropDown:

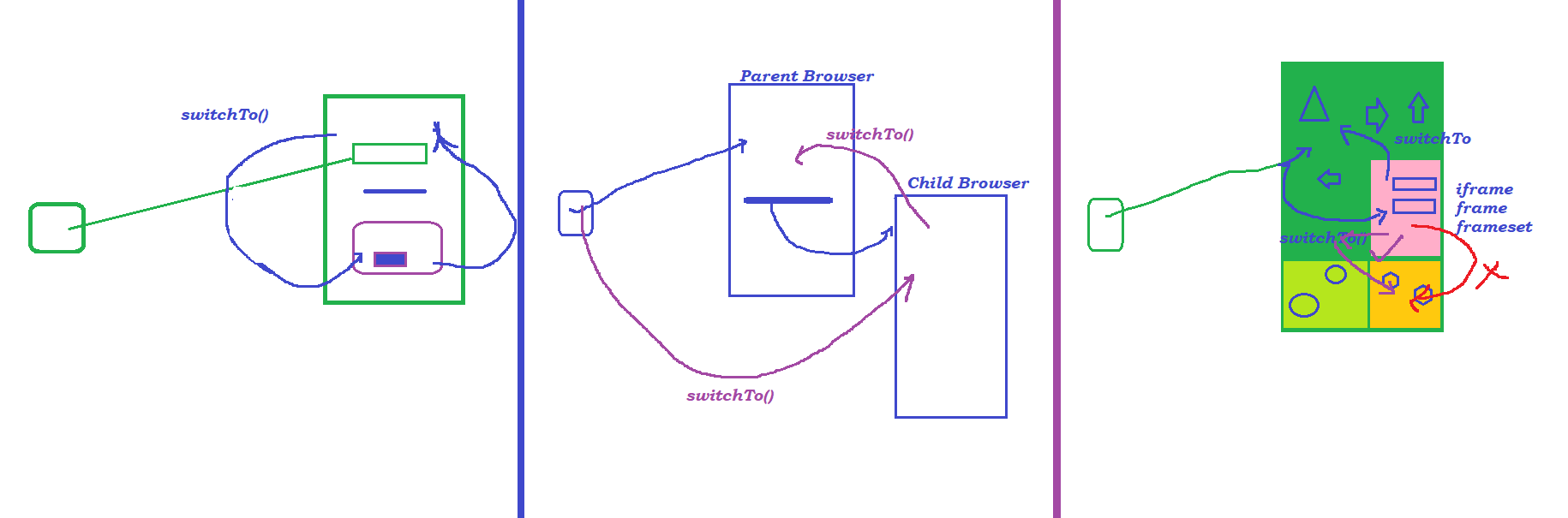


Browser Operations:



Actions in selenium:

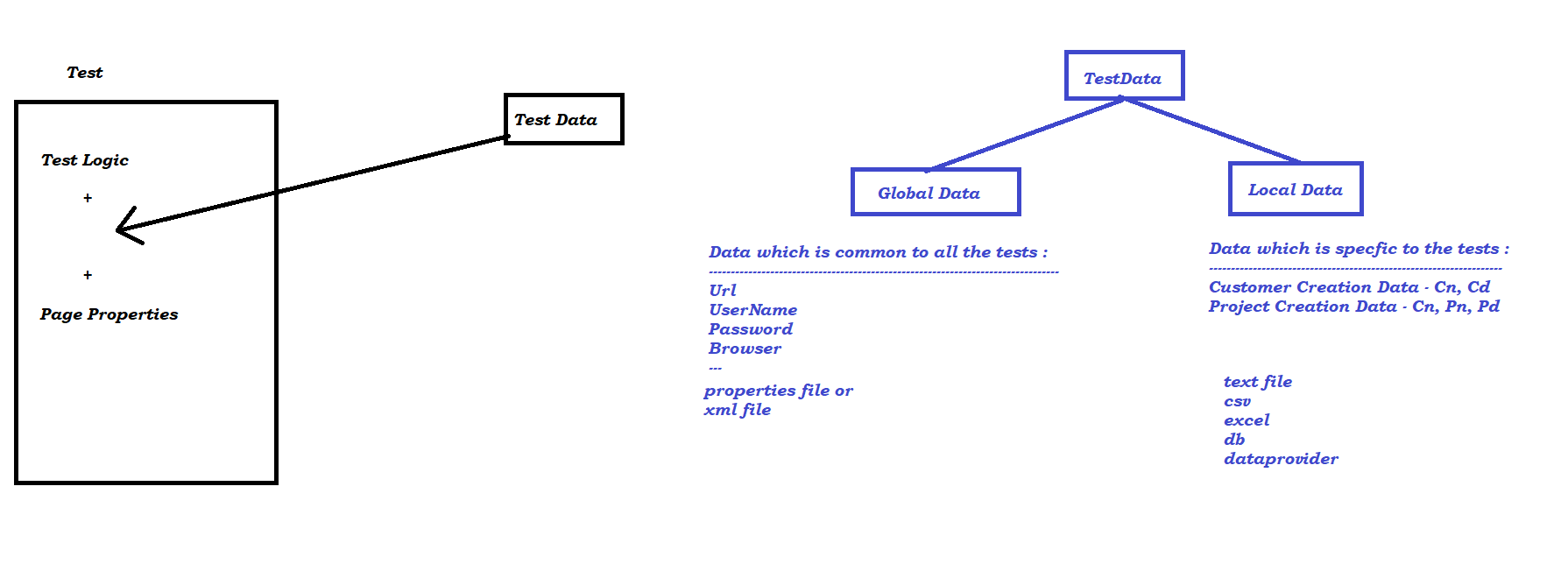
SwitchTO()

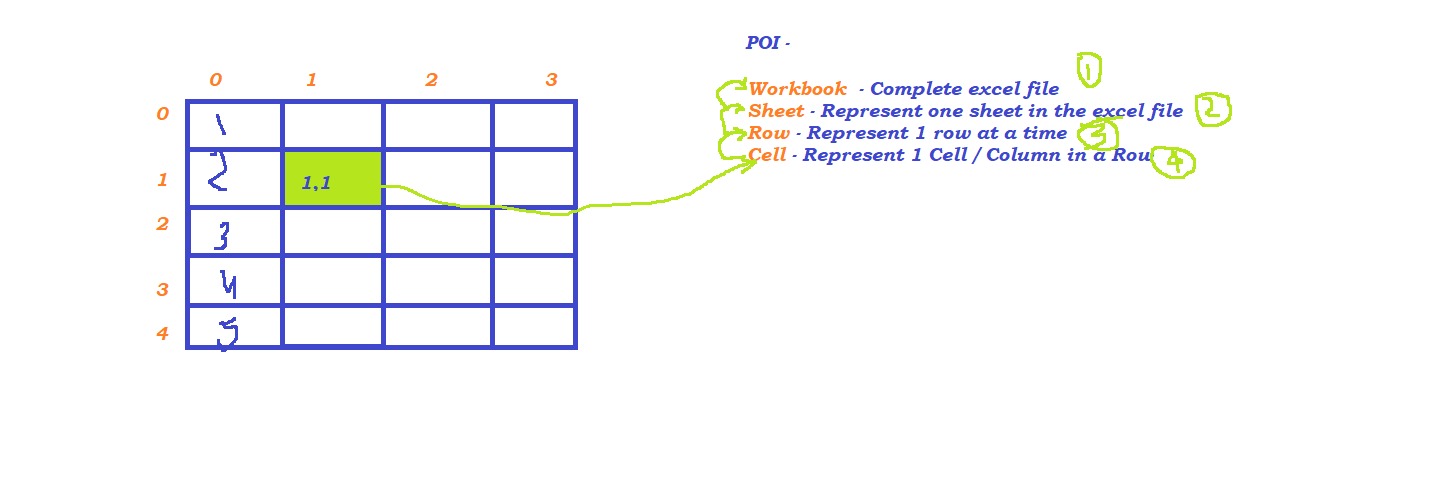


End-to-End Testing :

1. Execute the scenario / Test manually
2. Identify the required test data
3. Identify the places where we can write reusable functions
4. Identify the Validation Steps.
5. Start implementing the reusable functions by using external data, validations etc
6. Call the functions which you have implemented in step-5 to automate the test
7. Execute the newly written test at least 5-10 times

DataDriven Testing





POP UP :

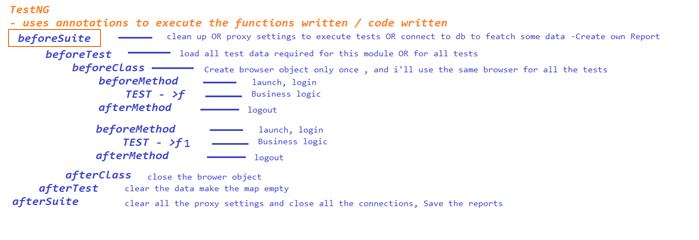
* alerts
* confirmation popup
* hidden division popup
* file download popup
* file upload popup

Windows Automation Tools :

AutoIT

Sikuli

TESTNG:



POM :

