09-Feb-2020

Pre-Req: ( Pre-Recorded Session – Intellipaat ) (50/60)

- JAVA

1. OOPS – concepts
   1. Abstraction
   2. Encapsulation
   3. Polymorphism
   4. Inheritence
2. Variables
   1. Local variables
   2. Global variables
   3. Static variables
   4. Constants
3. Methods (or functions)
   1. With return type
   2. Without return type
4. Access specifiers and Access modifiers (non-functional Access specifiers )
   1. Private
   2. Default
   3. Protected
   4. Public
   5. Static
   6. Final
   7. Abstract
   8. Synchronized ( less used )
5. Constructors
   1. Default constructor
   2. Parameterized constructor
6. Class
7. Interface and abstract class
8. Looping Statements => one program
   1. For
   2. While
   3. Do while
   4. For each
9. Conditional Statements
   1. If
   2. If else
   3. If else if else
   4. Switch
10. String <Class> ->java.lang =>\*\*\*atleaset 2-3 Question and atleast one program \*\*\*
    1. Functions – length(), split(),indexOf(),lastIndexOf(),substring(-,-) ...
    2. StringBuffer
    3. StringBuilder
    4. Programs 🡪 reversing string, reversing word, palindrome..
11. File Handling 🡺Java.io
    1. InputStream
    2. OutputStream
    3. How to print the file names in a folder
    4. How to read the text file
    5. How to write into text file
12. Exception Handling
    1. Try
    2. Catch
    3. Finally
    4. Throw
    5. Throws
13. Threads
    1. Thread
    2. Runnable Interface
14. Collections \*\*\*\*\*\*\*
    1. List
    2. Set
    3. Queues
    4. Map
15. Generics

Content:

Selenium :

* Selenium IDE🡪 POC, Record and PlayBack
* Selenium RC
* Selenium WebDriver
  + Selenium 2.0
  + Selenium 3.0 ->3.141.59
    - Xpath writing techniques \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 70%
    - Type -> text box, text area,...
    - Click -> checkbox, radio button, calendar, image,buttons,links ...
    - Reading text from application screen
    - Drag and drop
    - Moving mouse
    - webTable
    - Browser actions – maximizing, refresh, back button, forward button...
    - Implicit wait
    - Explicit wait
    - Fluent Wait
    - How to handle multiple browser
    - How to execute script on different browsers
    - How to achieve data driven testing in Selenium
      * Text file
      * Excel file
    - Validations in Selenium
    - Handling popup
      * Alerts
      * Confirmation popup
      * Authentication popup
      * File Download popup
      * File Upload popup
  + Selenium 4.0 -> alpha version
* Selenium Grid
  + Distributed Execution
* Frame Work:
  + Function Driven Automation Framework
  + Data Driven Automation Framework
  + Keyword Driven Automation Framework
  + Hybrid Driven Automation Framework
  + TestNG
  + Page Object Model
* Maven - Build Automation Tool
  + How to create maven project
  + Life cycle of maven
  + Phases of maven
  + How to manage dependencies
* Introduction Appium – Mobile Automation Tool

Advanced :

* Git
* Jenkins
* BDD

What is Automation Testing ?

1. Complex (Regression Testing)
2. 4-6 Hrs (Productive work) 8-9Hrs
3. Time consuming
4. Person – not on Process
5. Mood –
6. Coverage not 100%

Automation (24 X7) :

Process in which we relay on Tools to perform testing activities.

Tools:

1. UFT
2. Test Complete
3. Sahi
4. Selenium
5. Appium
6. Egg plant
7. Cucumber
8. RFT
9. Test Studio
10. Silk Test
11. Test Partner

Selenium :

1. Open source Freely available software
2. Multiple browser (almost all popular browsers) for automation
3. Platform independent (Windows, Linux, Mac)
4. Multiple programming language – Java, ruby, python, c#, perl, php, Javascript.
5. No need of dedicated machine while executing script
6. Parallel Execution
7. Distributed Execution



Drawbacks :

* Only Web Applications ( selenium 4.0 alpha version)
* We can not test the application with latest version of browser (at least 1 version we are behind)

Selenium IDE : - Mainly used for POC

* First version of selenium
* Only version of selenium which has UI
* Record and playback
* Only on FF and Chrome as an add-on

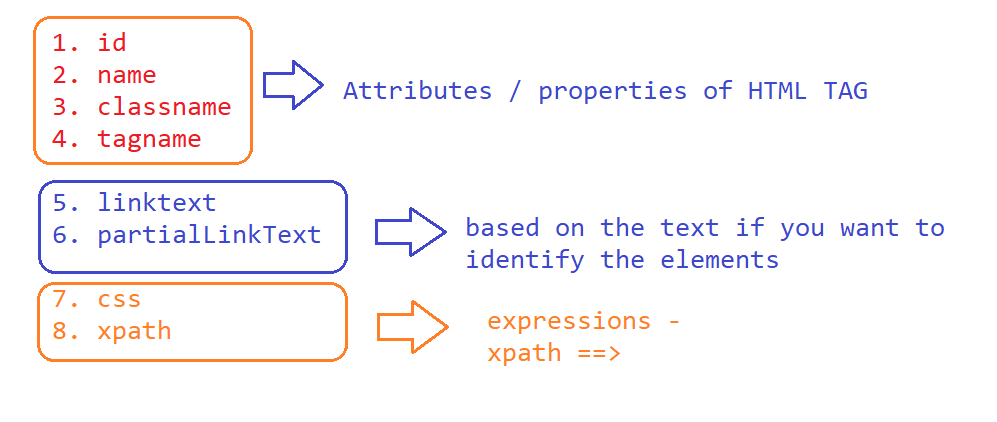
SELENESE – is the script generated by selenium IDE

Validation in IDE:

1. Assert – will not continue the execution till the last line when there is a failure
2. Verify - will continue the execution till the last line even though there is a failure

8 ways

1. Id
2. Name
3. Classname
4. Tagname
5. Linktext
6. Partiallinktext
7. Css
8. Xpath



***Expressions*** [ syntax ]:

1. Css :
   1. Html\_tag[property\_name=’property\_Value’]

<input type="text" name="username" value="" id="username" class="textField" placeholder="Username">

Input[placeholder=’ Username’]

input[id='username']

* 1. Html\_tag#id\_value OR #id\_value

Input#username OR #username

* 1. Html\_tag.class\_value OR .class\_value
  2. Parent to child
     1. Css\_for\_parent > child tag

1. Xpath – xml PATH 🡺 find the element present in DOM ( Document Object Model )
   1. Absolute xpath :
      1. /html/body/table/tbody/tr/td/table/tbody/tr[1]/td/div/table/tbody/tr/td[2]/div/table/tbody/tr[5]/td/table[1]/tbody/tr[1]/td/table/tbody/tr[1]/td/input
   2. Relative xpath:
      1. //html\_tag
      2. //html\_tag[@property\_name=’property\_value’]

//input[@name='pwd']

* + 1. LOGICAL OPERATORS
       1. AND
          1. //html\_tag[@property\_name1=’property\_value1’ and @property\_name2=’property\_value2’ ]

//input[@type='radio' and @value='radio-button-1']

* + - 1. OR
         1. tag[@property\_name1=’property\_value1’ or @property\_name2=’property\_value2’ ]

//input[@type='radio' or @type='checkbox']

Example - <https://www.redbus.in/> -

//div[@id='rb-calendar\_onward\_cal']//td[(@class='current day' or @class='wd day' or @class='we day') and text()='29']

* + 1. XPATH using Functions :
       1. Text()
          1. //html\_tag[text()=’complete value’]

//div[text()=' High School ']

* + - 1. Contains(arg1,arg2)
         1. arg1- can be a text() function or it can be attribute [property]
         2. arg2- partial value
         3. //html\_tag[contains(text(),’High’)]
         4. //img[contains(@src,'timer')]
      2. Starts-with(arg1,arg2)
         1. arg1- can be a text() function or it can be attribute [property]
         2. arg2- partial value
         3. //h3[starts-with(text(),'Intellipaat')]
         4. //button[starts-with(@id,'ext-gen')]
    1. Child to parent

Syntax:

//parentHtmlTab[expression of child element]

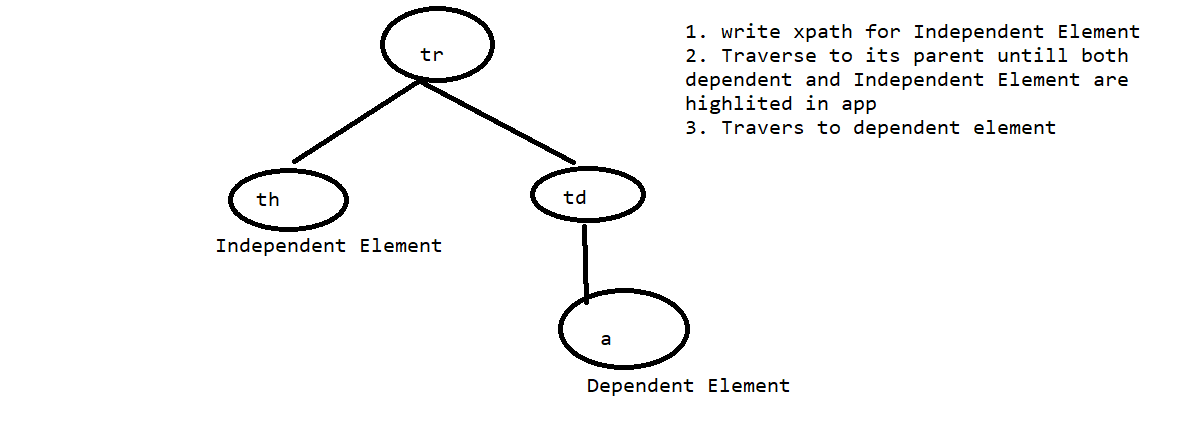
Ex - //a[div[text()='Login ']]

Ex2 - //tr[th[text()='Directed by']]//a

OR

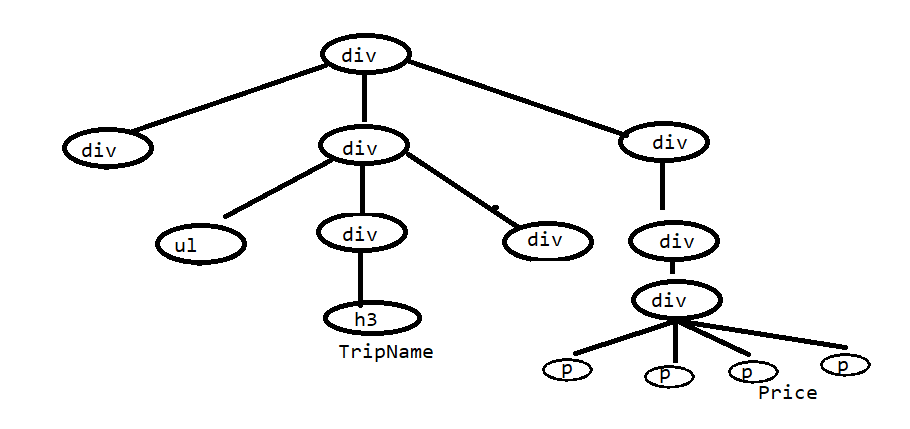
Ex3 - //th[text()='Directed by']/parent::tr//a

🡺<https://en.wikipedia.org/wiki/K.G.F:_Chapter_1>



Make my trip :

//div[div[div[h3[text()='Bali Weekend Getaway - 3 Nights']]]]//p[contains(@class,'blackText')]



//div[div[div[h3[text()=**'Soulmate Special Bali - Ubud Pool Villa Special 4 Nights'**]]]]//p[contains(@class,'blackText')]

* + 1. Axes Functions :
       1. Following-sibling
          1. //th[text()='Directed by']/following-sibling::td
       2. Preceding-sibling
          1. //a[text()='File Download']/preceding-sibling::a
       3. Following
          1. //a[span[text()='Soundtrack']]/following::a
       4. Preceding
          1. //a[span[text()='Soundtrack']]/preceding::a
       5. Parent
          1. //th[text()='Directed by']/parent::tr
       6. Child
          1. //th[text()='Directed by']/parent::tr/child::td/a
       7. Ancestor
          1. //h3[text()='Soulmate Special Bali - 7 Nights']/ancestor::div[contains(@class,'packageDetailsBox')]//p[contains(@class,'blackText')]

GIT SETUP

1 . Install git for windows

2. On your project folder open git bash

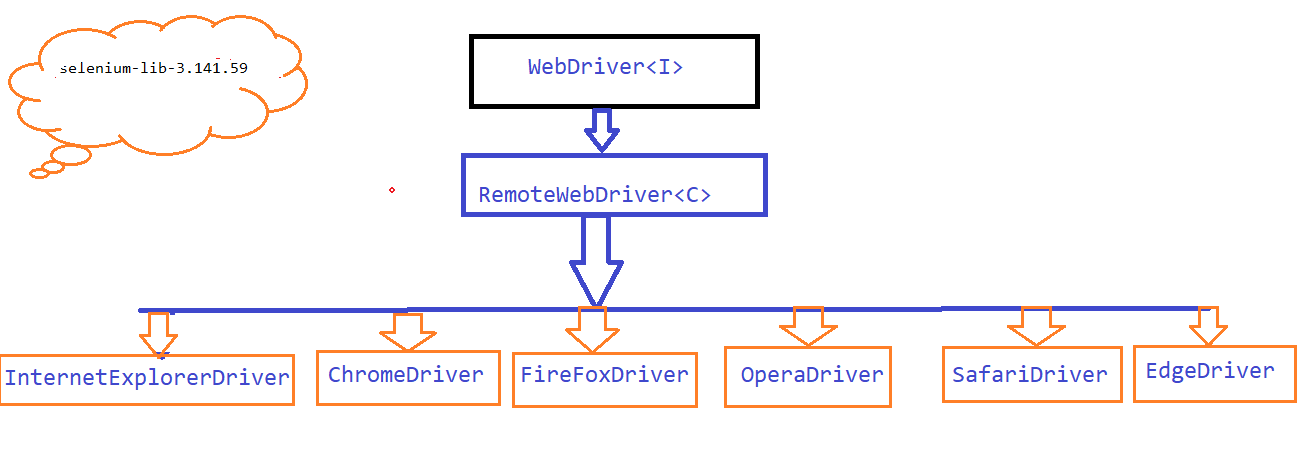
Selenium Setup

1. Download and install JAVA( 1.8 )
   1. Setting ENVIRONMENT VARIABLES

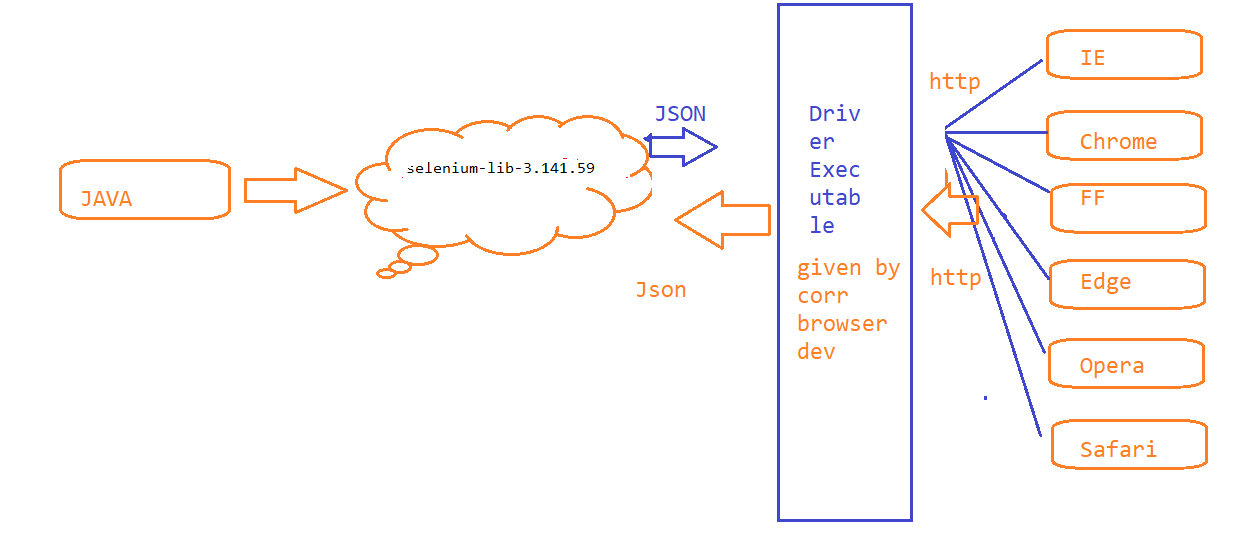
* JAVA\_HOME
* PATH

1. Eclipse ( Java Developers )

Architecture



JSON- Wired Protocol



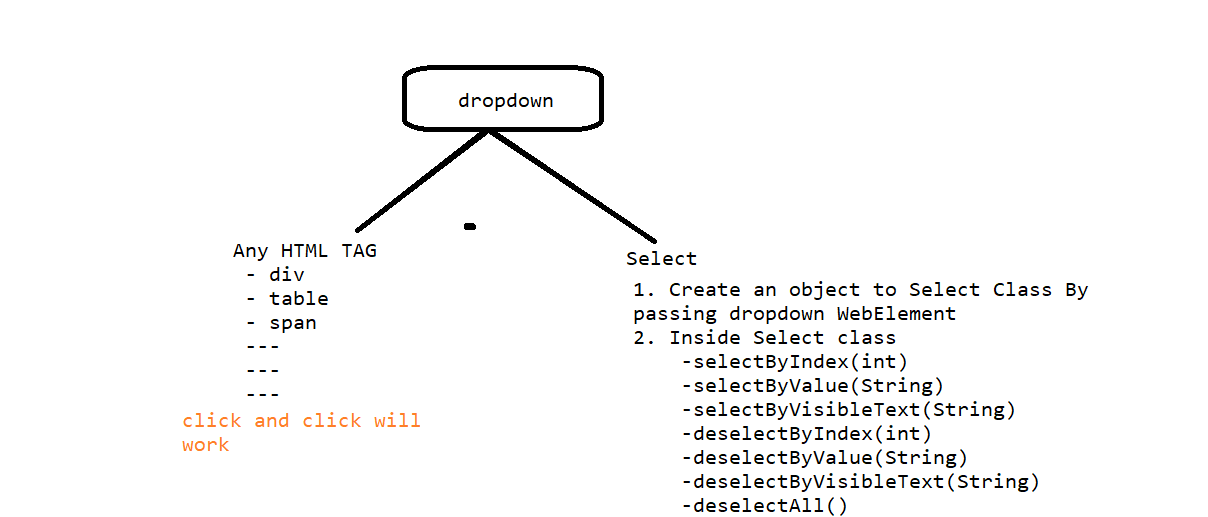
Important Functions :

* get(**String**) 🡺 Launch the application
* findElement(By) 🡺 is used to find the Web element in the webpage.
* By 🡺 Class 🡪 8 static methods each method is for one identifier
  + Id
  + name
  + classname
  + tagname
  + linktext
  + paritalLinktext
  + css
  + xpath

Priority :

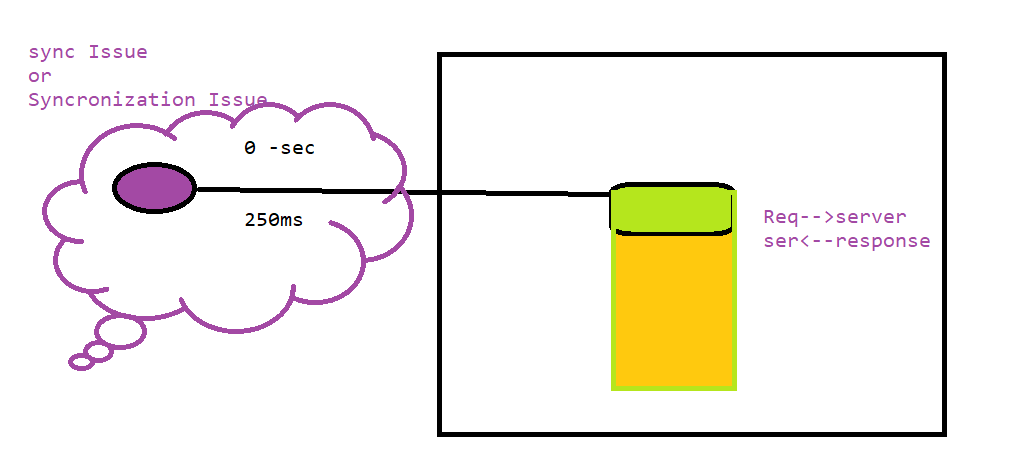
* + 1. ID
    2. NAME
    3. LinkText
    4. XPATH

DropDown:

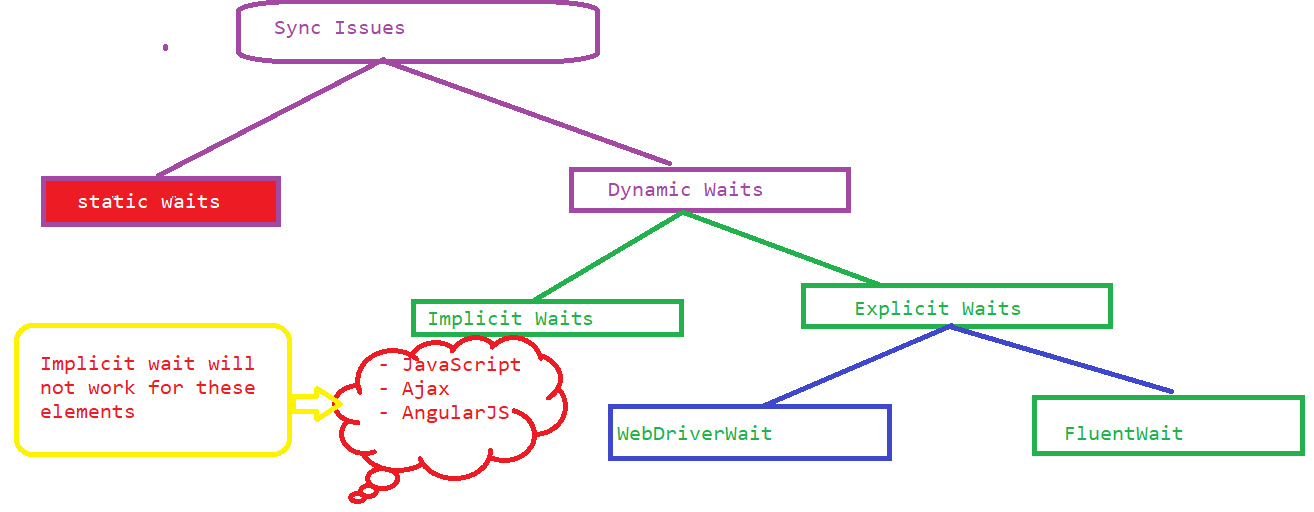


|  |  |  |
| --- | --- | --- |
| Class | Methods | Use |
| WebDriver | get() | Launch Browser |
|  | findElment() | To find the web element |
| By | id(String) |  |
|  | name(String) |  |
|  | className(String) |  |
|  | tagName(String) |  |
|  | linkText(String) |  |
|  | partialLinkText(String) |  |
|  | cssSelector(String) |  |
|  | xpath(String) |  |
| Select | selectByIndex(int) |  |
|  | selectByValue(String) |  |
|  | selectByVisibleText(Sting) |  |
|  | deselectByIndex(int) |  |
|  | deselectByValue(String) |  |
|  | deselectByVisibleText(Sting) |  |
|  | deselectAll() |  |
|  | getOptions() |  |

AutoSuggestions:



Handling Sync Issue:



Implicit wait: -> Override the default value( 0sec or 250 ms ) of timeout specified in Webdriver.

* We Will keep only once

WebDriverWait – (90% can be handled using WDW)

* You can not keep your own logic to wait for the element
* Interval of time (500ms) – polling time
* Ignoring exception is not possible
* Max time out
* WebElment for which you want to keep fluent wait

FluentWait :

1. Create an Object to Fluent Wait by passing T Object
   1. WebElement for which you want to keep fluent wait
   2. Max time out
   3. Interval of time (500ms) – polling time
   4. Ignoring exception is not possible
2. Write your own wait logic inside apply method present in function interface.
   1. Function<Interface> - > apply()

3. call function (written by you)

Tooltip

Information stored an element –

**alt** or **title** property - of a web element

**Actions**

* Class ->Actions (exact kb and mouse operations)
* Even though the element is displayed if we are not able to perform click and if we are not getting any exception then go with Actions
* Mouse movement Operations
* Rightclick
* Drag and Drop

NOTE – call perform method at the end of every line in actions

Validation Function

* Element Displayed
* Element Enabled
* Action (Type or click or getting text Ect..)