Automation

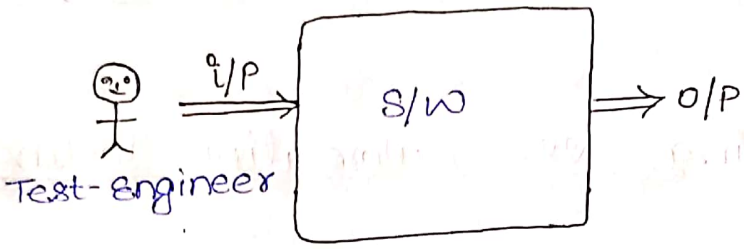
10th Mar '19

Any software can be tested using two ways:-

1. Manual Testing
2. Automation Testing

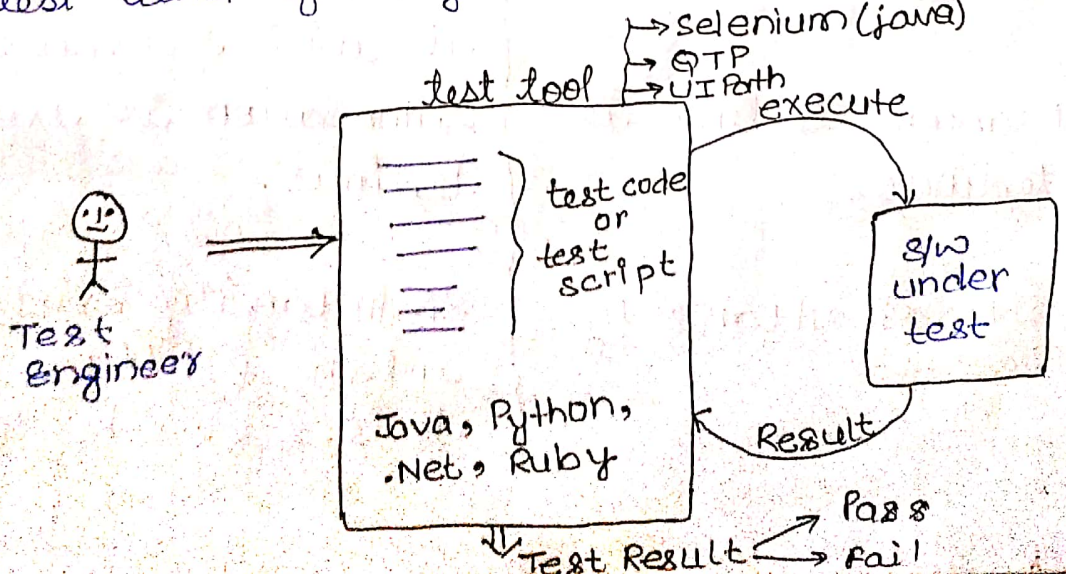
Manual Testing

Manual testing refers to the process of checking the s/w manually by providing input and analysing the output.



Automation Testing

Automation testing refers to the process of writing test code or test script in the test tool by the test-engineer and the test tool will execute the test code (or) test script on the software under test and finally obtain the test result.



Selenium tool

Test code or Test Script

```
int x = call add(3,4);
if (x == 7)
  {
    S.O.P("Pass");
  }
else
  {
    S.O.P("Fail");
  }
```

(3,4)

developer (a+b) 2/10

```
int add(int a, int b)
  {
    int c;
    c = a + b;
    return c;
  }
```

Manual Testing vs. Automation Testing

Manual Testing	Automation Testing
① It is not accurate.	It is accurate.
② More no. of human resources are required.	Less no. of human resources are required.
③ Investment is more.	Investment is less & it will be on tool.
④ Manual testing is tedious.	Automation is not tedious.
⑤ No simultaneity.	Simultaneity exists, where multiple devices can be test.

Q. When to use manual testing and when to use automation testing ?

Ans:- Manual Testing

a. Adhoc Testing
b. Software is a not stable.
c. Test the software for 1 or 2 times.
d. Usability Testing
e. Exploratory Testing

Automation Testing

a. Regression Testing
b. Functional Testing
c. Software is stable
d. Test the S/w for multiple times
e. Load Testing

Selenium :-

Selenium is a open source freely downloadable Automation testing tool using which we can test the web-application.

History

Jason Huggins (ThoughtWorkj) 2004

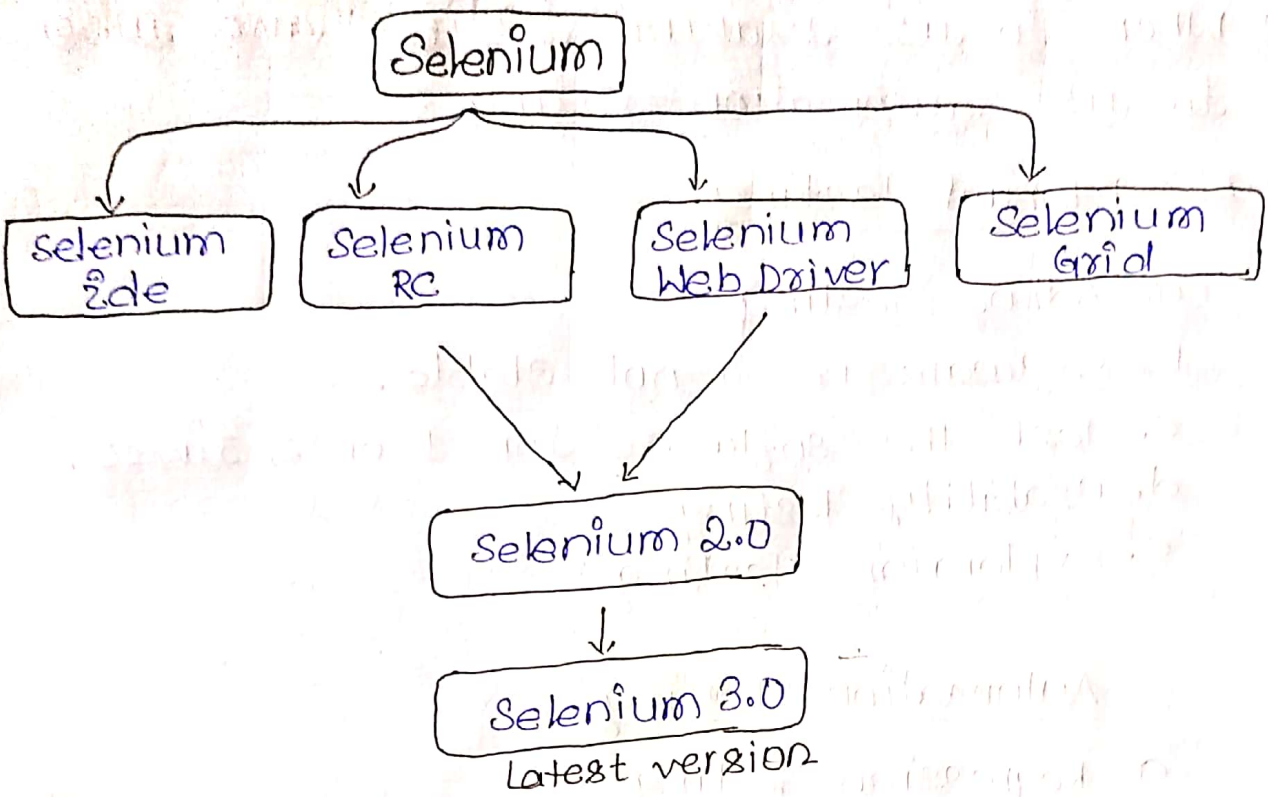
Selenium

Java Script

Selenium
- Ide
- Core
- RC

Simon (Google) 2009

Web Driver



Note :-

Selenium ide and Selenium RC is Deprecated which means no longer used in company or IT industry.

Defⁿ :-

Selenium is not a single tool, rather it is a suite of tool or collection of tools.

Note :-

Using Selenium we can test only web-applications not desktop application.

11th Mar '19

Selenium Ide (Integrated Development Environment)

It is a record and playback tool which has GUI and is an add-on for chrome & Firefox Browsers.

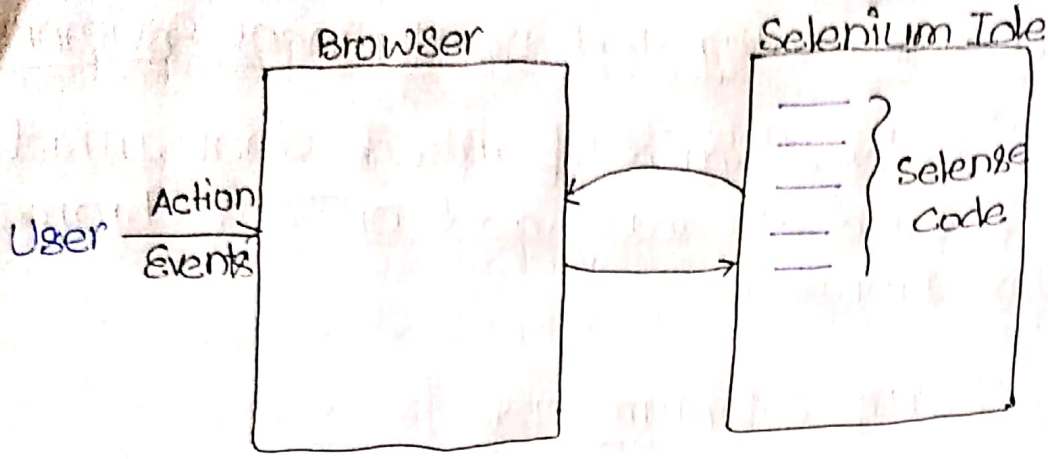
Steps to use Selenium Ide :-

1. Open chrome Browser
2. Navigate to www.seleniumhq.org
3. click on download
4. In the Selenium Ide section, click on for chrome link.
5. click on Add to chrome.

Scenario to be Automated

1. Go to magento.com
2. click on my account link present at the bottom of the page
3. Enter the email "vineetanand61@gmail.com" and Enter the password "Welcome123".
4. click on login
5. click on logout

command	Target	Value
open	/	
set window size	1382x744	
run script	window.scrollTo	



Selense code :-

The code written by selenium ide by recording the user's actions or events performed in the browser.

Selenium Web Driver

Web Driver is an interface which is used to test the software under test in different browser. or web automation framework.

Software Required to Automate using web driver :-

1. Java JDK (1.8)
2. Eclipse Ide (oxygen)
3. Selenium jar
4. Driver Software
   (gecko driver, chrome driver, IE driver)

1. Open Eclipse Ide
2. Create a workspace
3. Once the eclipse is opened click on File → New → JavaProject
4. Give the project name as SeleniumDemo
5. Click on Finish

Steps to download Selenium jar

1. Open Chrome Browser
2. Go to www.Seleniumhq.org
3. Click on download
4. In the Selenium Standalone Server, Click on Download version Link
5. Once the zip file was downloaded, create a folder in any of the drives by the Selenium Components.
6. Place the downloaded jar in Selenium component folder.

Steps to Download Driver Softwares

1. Go to seleniumhq.org website
2. Click on download
3. Scroll down to Third party Browser Drivers Section

4. a. Download Geecko Driver
   i. Click on latest link of mozilla geecko driver.
   ii. Click on geeckodriver win 64 bit.
   iii. Place the downloaded zip file in Selenium component folder.

b. Download Chrome Driver

i. Click on the latest link of Google Chrome Driver
ii. click on chrome Driver 2.46 or 7.2.0
iii. click on ChromeDriver_win32.zip
iv. Place the downloadable zip in Selenium Components folder.

c. Download IE Driver

i. In the Internet Explorer Driver Software. click on windows 64 bit IE.
ii. Place the downloaded zip in the Selenium Component folder.

13th Mar'19

Adding Driver Software

Driver S/w can be added to the project using three ways :-

1) using System.setproperty(key, value) method
2) using Environment variables
3) By pasting the drivers in the project home folder (root folder).

Q Write a script to launch Firefox Browser by adding the driver s/w to project using System.setProperty

```
public class Launch
{
    public static void main (String[] args)
    {
        System.setProperty ("webdriver.gecko.drives",
                            "c:\\Selenium ... \geckodriver.exe");
        FirefoxDriver driver = new FirefoxDriver();
    }
}
```

WAS to launch chromeBrowser by adding Driver s/w to project using System.setProperty

```
public class Launch
{
    p s v m (String[] args)
    {
        System.setProperty ("webdriver.chrome.driver",
                            "c://selenium../chromedriver.exe");
        ChromeDriver driver = new ChromeDriver();
    }
}
```

WAS to launch Internet explorer by adding the drivers to the project using System.setProperty.

```
class Launch
{
    p s v m (String [] args)
    {
        System.setProperty("webdriver.ie.driver",
                "c:\\Selenium... / IEDriverServer.exe");

        InternetExplorerDriver driver = new
                InternetExplorerDriver();

    }
}
```

Using Environment Variables

1. Select ThisPc and right click on it
2. Click on Properties
3. Click on Advance System Setting
4. Click on Environment Variables
5. Under System Variables search for path variable.
6. Select & click on Edit
7. Click on New
8. Paste the path of SeleniumComponents folder

Adding the Driver Software By Pasting in the project home folder

1. Go to Selenium Component folder
2. Copy all the executable driver s/w's
3. Go to Eclipse, select the project and paste it.

Note :-

If an Internet Explorer is not working, then follow the below steps :-

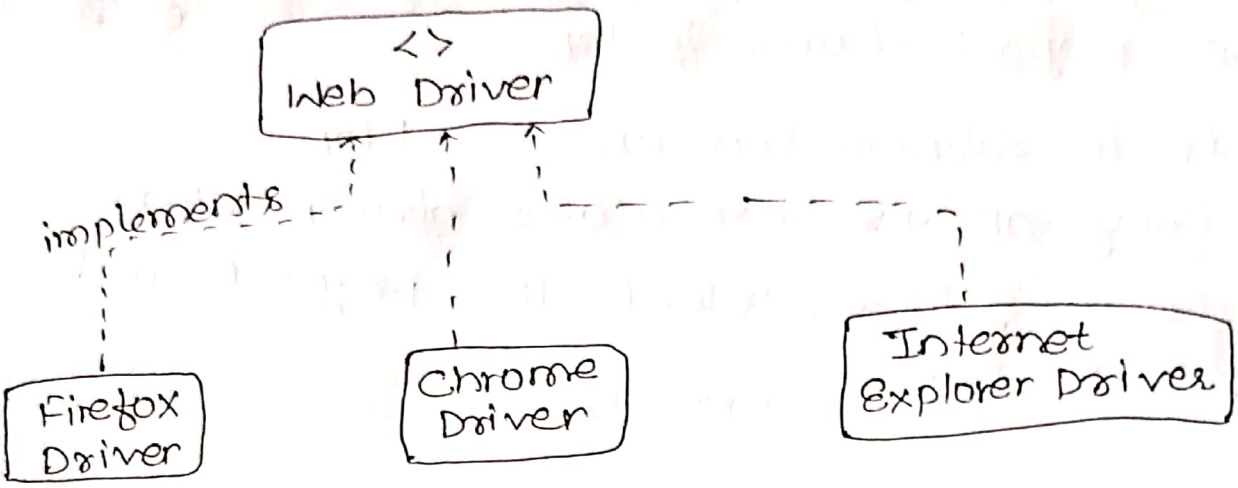
a) Open the Internet Explorer

b) Click on Setting icon, make sure the zoom is selected to 100%

c) Also make sure that in the internet options, click on the Security, make sure that all the four check box are either selected or not selected.

WAS to launch Firefox Browser, go to gmail.com, wait for 5 seconds and then close it.

```
class Launch
{
    p s v m (String[] args)
    {
        WebDriver driver = new FirefoxDriver();
        String url = "https://www.gmail.com";
        driver.get(url);
        Thread.sleep(5000);
        driver.close();
    }
}
```



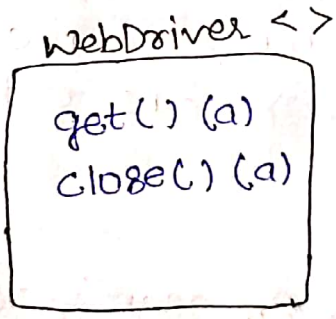

```
FirefoxDriver driver = new FirefoxDriver
```

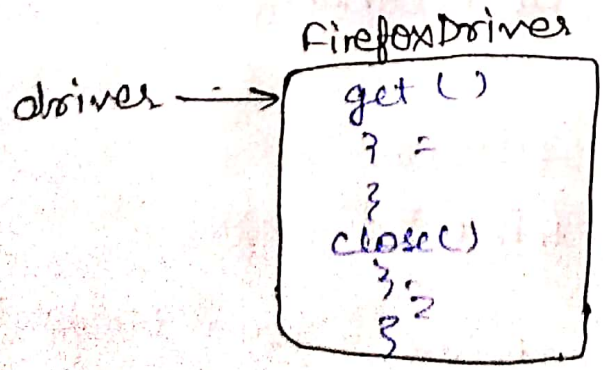
(or)

```
WebDriver driver = new FirefoxDriver
```

Parent ref creating object

} Runtime Polymorphism





Write a script to get the URL of the website

```
public class Launch
{
    p s v m (string[] args)
    {
        WebDriver driver = new chromeDriver();
        String url = "https://www.facebook.com";
        driver.get(url);
        String curl = driver.getCurrentUrl();
        S.O.P(curl);
        driver.close();
    }
}
```

Write a script to verify the title of Gmail Application.

```
class Launch
{
    p s v m (string [] args)
    {
        WebDriver driver = new chromeDriver();
        String url = "https://www.gmail.com";
        driver.get(url);
        String etitle = "gmail";
        String atitle = driver.getTitle();
        S.O.P(etitle);
        S.O.P (atitle);
        if (etitle.equalsIgnoreCase(atitle))
        {
            S.O.P("Pass");
        }
        else
        {
            S.O.P("Fail");
        }
    }
```

WAS to get the page source of website.

```
class Launch
{
    p s v m (String [] args)
    {
        WebDriver driver = new ChromeDriver();
        String url = "https://www.facebook.com";
        driver.get(url);
        String ps = driver.getPageSource();
        S.O.P(ps);
        driver.close();
    }
}
```

WAS to maximize the chrome Browser.

```
class Launch
{
    p s v m (String [] args)
    {
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
    }
}
```

Write a script to get the title and current Url of the gmail application either in Firefox or chrome Browser based on the input given by the user.

```
public class Launch
{
    p s v m (String [] args)
    {
        Scanner sc = new Scanner(System.in);
        S.O.P("Enter the browser");
        String browser = sc.next();
        if(browser.equalsIgnoreCase("Firefox"))
        {
            FirefoxDriver f = new FirefoxDriver();
            test(f);
        }
        else
        {
            ChromeDriver c = new ChromeDriver();
            test(c);
        }
    }
    p s v test(FirefoxDriver driver)
    {
        driver.get("https://www.gmail.com");
        String title = driver.getTitle();
        S.O.P(title);
        String curl = driver.getCurrentUrl();
        S.O.P(curl); driver.close();
    }
    p s v test(ChromeDriver driver)
    {
        =
        =
    }
}
```

Polymorphic Approach (Runtime Polymorphism)

```
class launch
{
    p s v m (String. args[])
    {
      Scanner sc = new Scanner (System.in);
      S.O.P("Enter the browser");
      String browser = sc. next();
      if (browser. equalsIgnoreCase("Firefox"))
        {
          FirefoxDriver f = new FirefoxDriver();
          test(f);
        }
      else
        {
          ChromeDriver c = new ChromeDriver();
          test(c);
        }
    }
public static void test (WebDriver driver)
    {
        driver.get("https://www.gmail.com");
        String title = drives .getTitle();
        S.O.P(title);
        String curl = driver. getCurrentUrl();
        S.O.P(curl);
    }
}
```

15th Mar '19

Locators

A website is a collection of web pages.
A webpage is a collection of web elements.
Ex:- TextBox, Button, Link, Image etc.
A web page is made up of html.

- Whenever we want to perform any action on web-element using selenium, selenium can't directly perform the action.
- In order to perform the action, it will make use of a concept called as Locators.
- In other words, Locators helps selenium to perform action on web elements in a web page.

There are eight types of locators :-

1) Id
2) name
3) ClassName
4) TagName
5) LinkText
6) partialLinkText
7) CSS Selectors
8) X path

Identifying the element using id

Write a script to enter the email in the email field of facebook application.

```
Class IDDemo
{
    p s v m (String [] args)
    {
        WebDriver daiver = new ChromeDriver();
        driver.manage().window().maximize();
        String url = "https://www.facebook.com";
        driver.get(url);
        WebElement email = drivers.findElement(By.id("email"));
        email.sendkeys("mzaidur rahuman@gmail.com");
    }
}
```

Note:- Inspect the webelement first & find the id name or further locators attribute value.

Ex:- < id = "email" >

Identifying the element by using name

Write a script to enter the password in the password field of facebook application.

```
class NameDemo
{
    p s v m (String [] args)
    {
        WebDriver driver = new ChromeDriver();
        String url = "https://www.facebook.com
        driver.get(url);
        WebElement pwd = driver.findElemen
                        (By.name("pass"));

        pwd.sendKeys("Welcome123");

        driver.close();
    }

}
```

Identifying the element By using className

WAS to enter the email in the email field of facebook application.

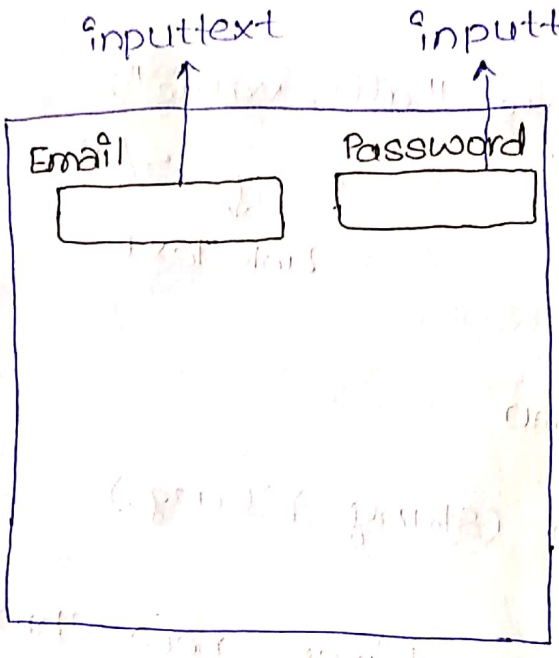
```
class CNDemo
{
    p s v m (string [] args)
    {
        WebDriver driver = new
                    ChromeDriver();
        driver.manage().window().
                    maximize();
        string url = "https://www.
                    facebook.com";
        driver.get(url);
        WebElement email = driver.
            findElement(By.className
                    ("inputtext"));
        email.sendKeys("mail@gmail.com");
        driver.close();
    }
}
```

email

```
<input type = "email" class = "inputtext"
data-testid = "royal_email">
```

password

```
<input type = "password" class="inputtext"
name = "pass" id = "pass" data-testid = "royal-
                                        pass">
```



*** Here, email & password have same className. When the java program is to come to find Element by using className then it give keys to the first one (here email is first one), when the className is same for both the web elements.

Whenever we write the test code in Selenium, if it identifies multiple elements, it will not generate an exception or display an error rather action would be performed in first web element. As discussed in the above example, the class value of email field &

password field is same. As email is the first element, then the action would be perform in the email field.

Identifying the element by using Link Text

Write a script to click on forgotten account link of facebook application.

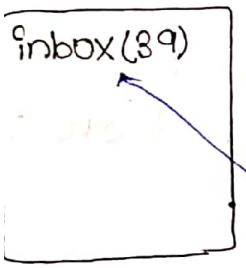
```
<tag attrname = "attr_value">
<a  href = "          " > ——— </a>
                          ↓
                       Link Text
```

```
class LinkDemo
{
    p s v m (String [] args)
    {
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        String url = "https://www.facebook.com";
        driver.get(url);
        WebElement link = driver.findElement
                    (By.linkText("Forgotten account?"));

        link.click();

    }
}
```

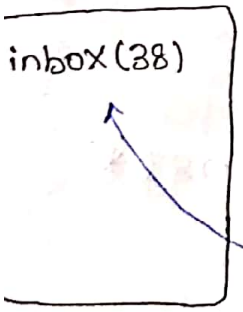
Identifying the element using partialLinkText

```
WebElement link = driver.findElement
                  (By.partialLinkText("Forgot"));

link.click();
```

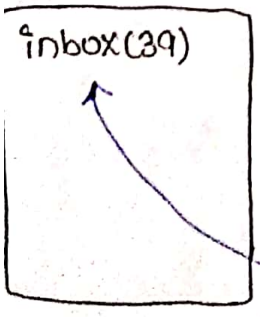

```
WebElement link = driver.findElement
                  (By.linkText("inbox(38)"));

link.click();
```




```
WebElement link = driver.findElement
                  (By.linkText("inbox(39)"));

link.click();
```




```
WebElement link = driver.findElement
                  (By.partialLinkText("inbox"));

link.click();
```




```
WebElement link = driver.findElement
                  (By.partialLinkText("inbox"));

link.click();
```

Notes :-

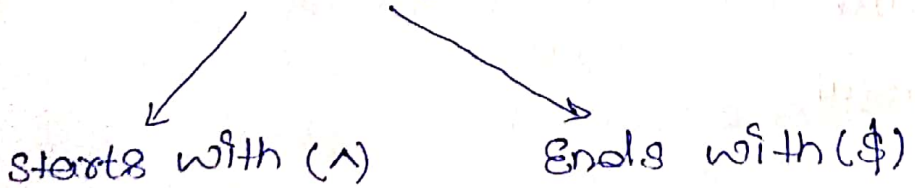
Whenever we want to handle static links (which will not change), then we will use LinkText.

Whenever we want to handle dynamic links (which will change), then we will use partialLinkText.

Identifying the elements by using CSS Selectors

There are totally five types of CSS selector syntax which are listed here :-

1. Tag & id
2. Tag & class
3. Tag & attribute
4. Tag.class & attribute
5. substring ⟶ contains (*)



18th Mar'19

① Tag & id

Syntax :- tag#id

input #email

```
public class Launch {

    p s v m (string [] args) {

        ==========

        WebElement email = driver.findElement(By.
                           cssSelector("input#email"));

        email.sendKeys("mzaidurrahman@gmail.com
                       );

    }

}
```

② Tag & class

Syntax :- tag.className

input.inputtext

```
WebElement email = driver.findElement(By.css
                   Selector("input.inputtext"));

email.sendKeys("mzaidurrahman@gmail.com");
```

③ Tag & attribute

Syntax :- tag [attr._name = 'attr._value']

input [data-testid = 'royal-email']

```
WebElement email = driver.findElement(By.
    cssSelector("input[data-testid='royal-
                                email']"));

email.sendKeys("pshibus11@gmail.com");
```

④ tag.class & attribute

Syntax :- tag.className [attr._name = 'attr._value']

input.inputtext[type = 'password']

```
WebElement pass = driver.findElement(By.
    cssSelector("input.inputtext
    [data-testid = 'royal_pass']"));

pass.sendKeys("hello123");
```

⑤ Substring

a) starts with (^)

Syntax :- tag[attr_name ^ = 'attr_value']

input[id ^ = 'iden']

```
WebElement email = driver.findElement
        (By.cssSelector("input[id ^= 'iden']"));
email.sendKeys("pshibu311@gmail.com");
```

b) Ends with ($)

Syntax :- tag[attr_name $ = 'attr_value']

input[id $ = 'rId']

```
WebElement email = driver.findElement(By.cssSelector
        ("input[id $ = 'rId']"));
email.sendKeys("pshibu311@gmail.com");
```

c) contains (*)

Syntax :- tag[attr_name * = 'attr_value']

input[id * = 'enti']

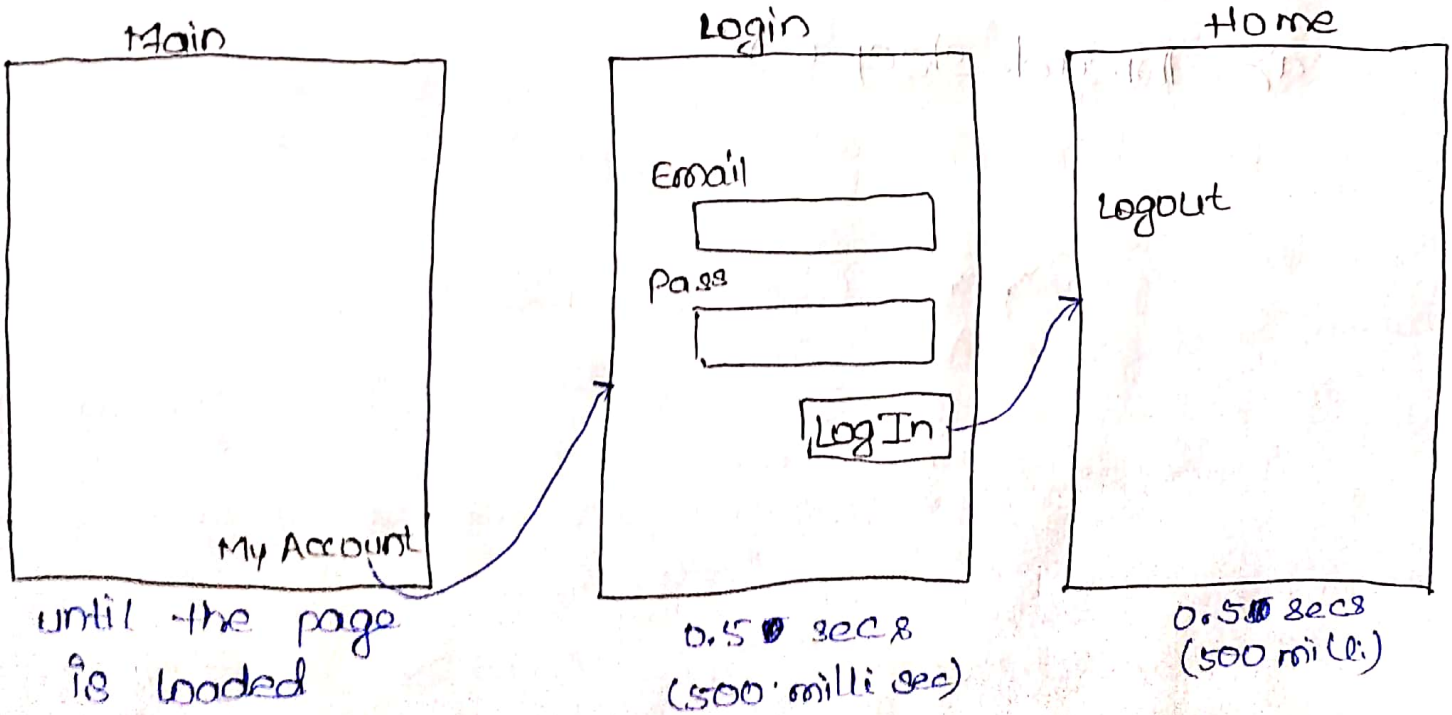
```
WebElement email = driver.findElement(By.css
        Selector("input[id * = 'enti']"));
email.sendKeys("pshibu311@gmail.com");
```

Automate the below Scenario :-

Open Chrome Browser
Navigate to Magento.com
click on MyAccount
Enter the email "vineetanand61@gmail.com"
Enter the password "Welcome123"
click on Login
click on Logout

```
class Launch
{
    p s v m (String [] args)
    {
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        String url= "https://magento.com";
        driver.get(url);

        WebElement myacct = driver.findElement
                (By.LinkText("My Account"));
        myacct.click();
        Thread.sleep(5000);
        WebElement email = driver.findElement
                (By.id("email"));
        email.sendKeys("vineetananad61@gmail
                .com");
```

```
WebElement pass = driver.findElement(By.id("pass"));
pass.sendKeys("Welcome123");
WebElement login = driver.findElement(By.id("Send2"));
login.click();
Thread.sleep(5000);
WebElement logout = driver.findElement(By.linkText
                                ("Log Out"));
logout.click();
Thread.sleep(5000);
driver.close();
}
}
```



Note :-

* Whenever selenium wants to perform action in the first Web page or the page loaded by get(.) method, then it will wait until the complete web page is loaded (all the WebElements are loaded).

But in subsequent pages or redirected pages, it will wait only 0.5 sec or (500 millisec)

* Within the 0.5 sec, if the element is not loaded then it will throw an Error i.e. NoSuchElement Exception.

* This problem can be overcome by three ways:

a) Thread.sleep()

19th Mar '19

PageLoadTimeOut( )

It is used to check whether the web page is loaded within the specified time.

If the page is loaded within the specified page then it will continue the execution of script or it will throw an exception i.e. TimeOutException.

Ex:-

Automate the scenario

① Go to Magenlo.com

② check whether the page is loaded within 5 sec

③ If the page is loaded within 5 secs. click on My Account else throw an exception.

```
class PageLoadDemo
{
    p s v m (String args[])
    {
        WebDriver driver = new ChromeDriver();
        driver.manage().timeOut().pageLoadTimeout
                                (5, TimeUnit.SECONDS);
        String url = "https://magenlo.com";
        driver.get(url);
        WebElement myacct = driver.findElement
                (By.LinkText("My Account"));
        myacct.click();
        Thread.sleep(5000);
        driver.close();
    }
}
```

Navigation commands

There are four navigation commands listed below :-

1. navigate().to(url);
2. navigate().back();
3. navigate().forward();
4. navigate().refresh();

Example :-

Scenario to be Automated

1. Go to Amazon.in
2. click on Your Amazon.in Link
3. click on Backward icon of the browser
4. click on Forward icon of the browser
5. click on Refresh icon of the browser.

```
class PageLoadDemo
{
    p s v m (String [] args)
    {
      WebDriver driver = new ChromeDriver();
    // driver.manage().timeOuts().pageLoadTime
                        Out(5, TimeUnit.SECONDS);
      String url = "https://amazon.in";
      driver.get(url);
      WebElement wy
      driver.navigate().to(url);
      WebElement myamazon = driver.findElement
                (By.LinkText("Your Amazon.in"));
      myamazon.click();
```

```
driver.navigate().back();
Thread.sleep(5000);
driver.navigate().forward();
Thread.sleep(5000);
driver.navigate().refresh();
Thread.sleep(5000);
driver.close();
}
}
```

Handling DropDown or Combo Box

In order to handle a dropdown, follow the below steps :-

1. Identify the dropdown using any one of the locators.
2. Create a object of Select class.
3. Pass the address of dropdown to the Select class constructor.
4. We can use any of the below listed methods to select ~~elements~~ options from the dropdown.

   i) SelectByIndex()
   ii) SelectByValue()
   iii) SelectByVisibleText()

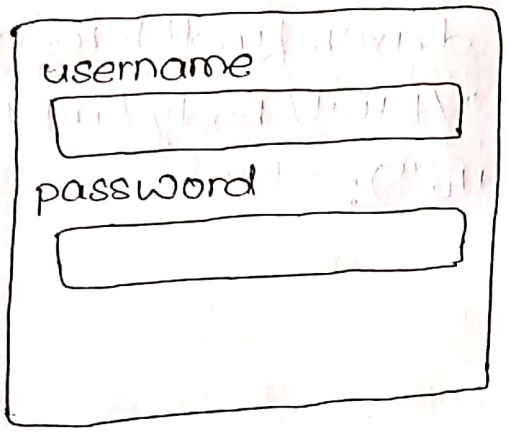
```
class Launch
{
    p s v m (String [] args)
    {
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();

        String url = "https://account.magento.com/
                customer/account/create/";
        driver.navigate().to(url);

        WebElement dropdown = driver.findElem
                (By.id("customer_company_type
        Select s = new Select (dropdown);

        s.selectByIndex(2);
        Thread.sleep (5000);
        s.selectByValue("analyst_media");
        Thread.sleep(5000);
        s.selectByVisibleText("Tech Partner");
        Thread.sleep(5000);

    }

}
```

Xpath

Whenever in Selenium, we are not able to identify and webelement using any one of the discuss locators, then we finally go for Xpath.

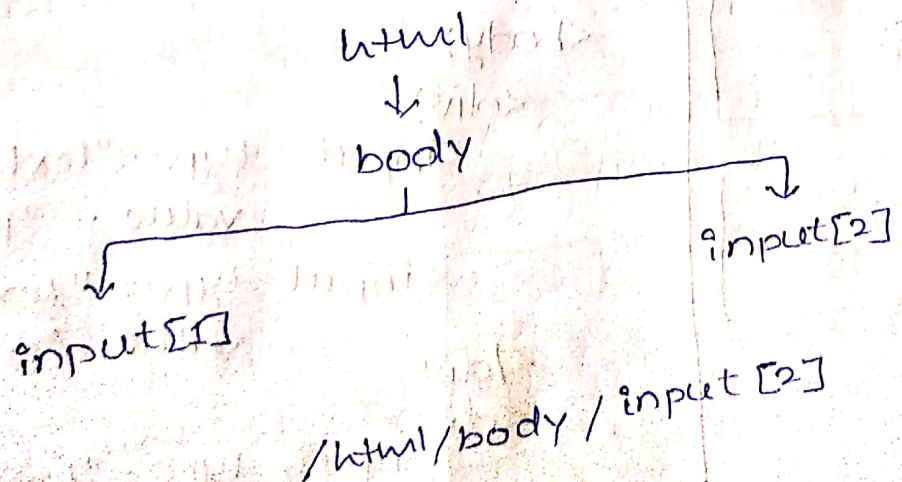
Xpath is a path which is obtained by traversing html tree structure.

Ex 1 :-




```
<html>
<body>
  username <input type="
      text" ><br>
  password <input type="
      text" >

</body>
</html>
```

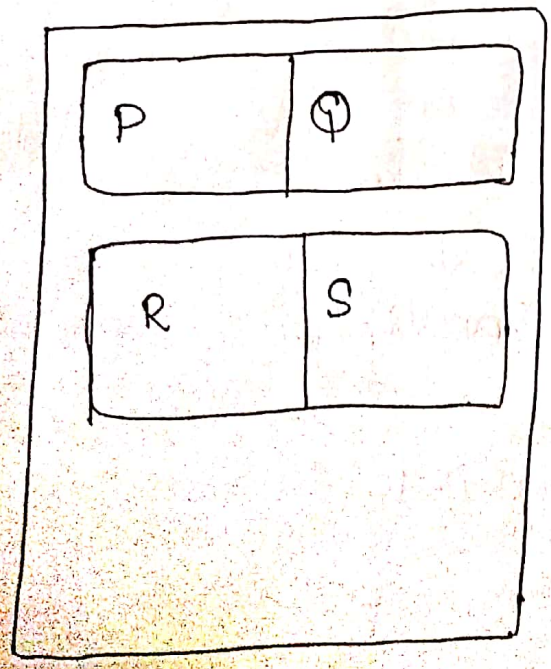
tag & attribute
input [type='text']



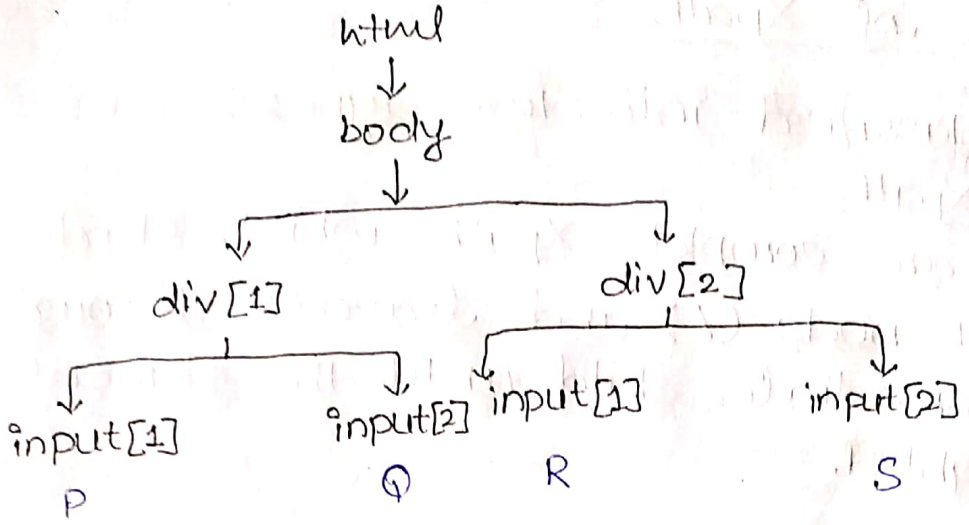
/html/body / input [2]

```
class Launch
{
    p s v m (String [] args)
    {
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        String url = "file:///C:  -  -  -  -  ";
        driver.get(url);
        WebElement pwd = driver.findElement(By.
                         Xpath("/html/body/input[2]"));
        pwd.sendKeys("hello");
        driver.close();
        }
}
```

20th Mar '19




```
<html>
  <body>
    <div>
      <input type="text"
             value = "P">
      <input type="text"
             value ="Q">
    </div>
    <div>
      <input type="text"
             value = "R">
      <input type="text"
             value="S">
    </div>
  </body>
</html>
```



Xpath	Matching Elements
/html/body/div[1]/input[1]	P
/html/body/div[1]/input[2]	Q
/html/body/div[2]/input[1]	R
/html/body/div[2]/input[2]	S
/html/body/div[1]/input	PQ
/html/body/div[2]/input	RS
/html/body/div[1]/input[1] | /html/body/div[2]/input[1]	PR
/html/body/div[1]/input[2] | /html/body/div[2]/input[2]	QS
/html/body/div[1]/input | /html/body/div[2]/input[1]	PQR
/html/body/div[1]/input[2] | /html/body/div[2]/input	QRS
/html/body/div[1]/input[1] | /html/body/div[2]/input	PRS
/html/body/div	PQRS

Classification of Xpath :-

Xpath is classified into two types :-

① Absolute Xpath

These are complete Xpath which starts from root node (/) and traverse through every immediate child until the element is highlighted.

② Relative Xpath

These are not complete Xpath. This Xpath starts from any intermediate node in a web page.

Note :-

The disadvantage of the absolute Xpath is that they are very lengthy, because Xpath is written from the root node.

Because of this disadvantage, In Industry, we will use Relative Xpath.

Relative Xpath	Matching Elements
//div[1]/input[1]	P
//div[1]/input[2]	Q
//div[1]/input	PQ
//div[1]/input[1]|//div[2]/input[1]	PR
//div[1]/input[2]|//div[2]/input[2]	QS
//div[1]/input|//div[2]/input[1]	PQR
//div[1]/input[2]|//div[2]/input	QRS
//div[1]/input[1]|//div[2]/input	PRS
//div	PQRS

Xpath with Attributes

Syntax :- //tag [@attr_name = 'attr_value']

//input [@id = 'u_0_l']

↓

To identify surname field in fb.

Xpath with Multiple Attributes

Syntax :- //tag[@attr_name = 'attr_value']
[@attr_name = 'attr_value']

//input [@class = 'inputtext'][@id = 'pass']

↓

To identify the password in fb.

Xpath with AND and OR operator

Syntax :-

//tag [@attr_name = 'attr_value' and
@attr_name = 'attr_value']

//input [@class = 'inputtext' and
@id = 'pass']

↓

To identify the password in fb.

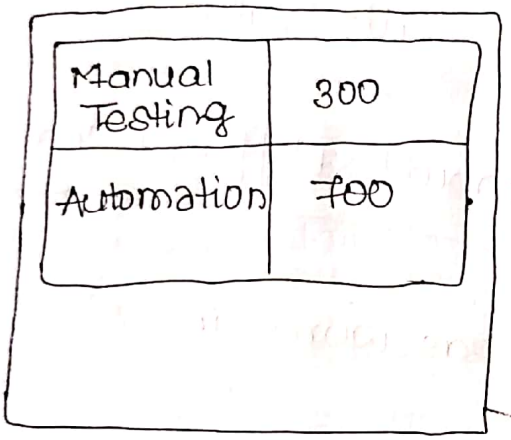
Syntax :-

//tag[@attr_name = 'attr_value' or
@attr_name = 'attr_value']

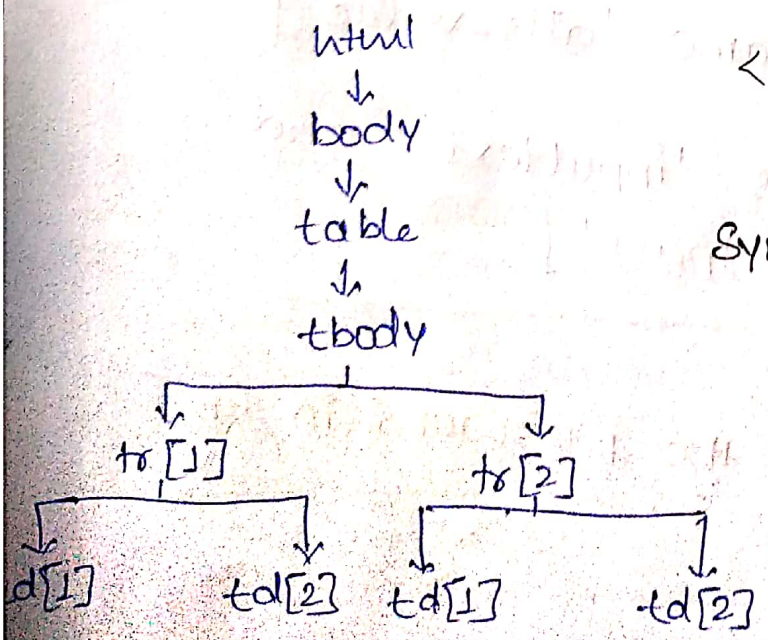
//input[@class = 'inputtext' or id = 'pass']

To identify the password in fb

Xpath using Text Method




```
<html>
  <body>
    <table border = "1">
    <tbody>
    <tr>
      <td>Manual Testing </td>
      <td> 300 </td>
    </tr>
    <tr>
      <td>Automation </td>
      <td> 700 </td>
    </tr>
    </tbody>
    </table>
  </body>
</html>
```



Syntax :- //tag[text() = 'text value']

//td[text() = 'Manual Testing']

Xpath using contains Method

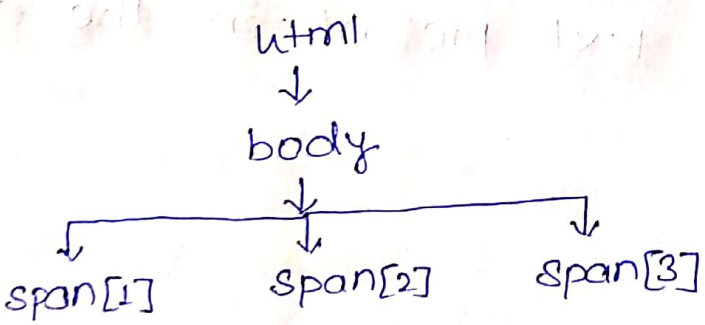
Syntax :-

```
//tag[contains(text(), 'text-value')]
//td[contains(text(), 'Manual Testing')]
```

21st March '19

Manual Testing Selenium Testing basic Selenium Testing advanced

```
<html>
 <body>
   <span> Manual Testing </span>
   <span> Selenium   Testing basic </span>
   <span> Selenium Testing advanced </span>
 </body>
</html>
```



//span[text() = 'Manual Testing'] → 1 element Matching

//span[text() = 'Selenium testing basic'] → 0

//span[text() = 'Selenium testing advanced'] → 1

//span[contains(text(), 'Manual Testing'] → 1

//span[contains(text(), 'Selenium'] → 2

//span[contains(text(), 'basic'] → 1

Note :-

In order to create a space in the html, we have two ways

① By using   (non-breakable space)

② By pressing space bar key from keyboard.

Whenever we have created a front-end ~~and~~ used in which for an element if we have used non-breakable space then text() will never identify that element.

In order to overcome this, we will use contains().

Even contains() has a disadvantage i.e. if we have an non-breakable space in between the text of the element then it will never identify that element.

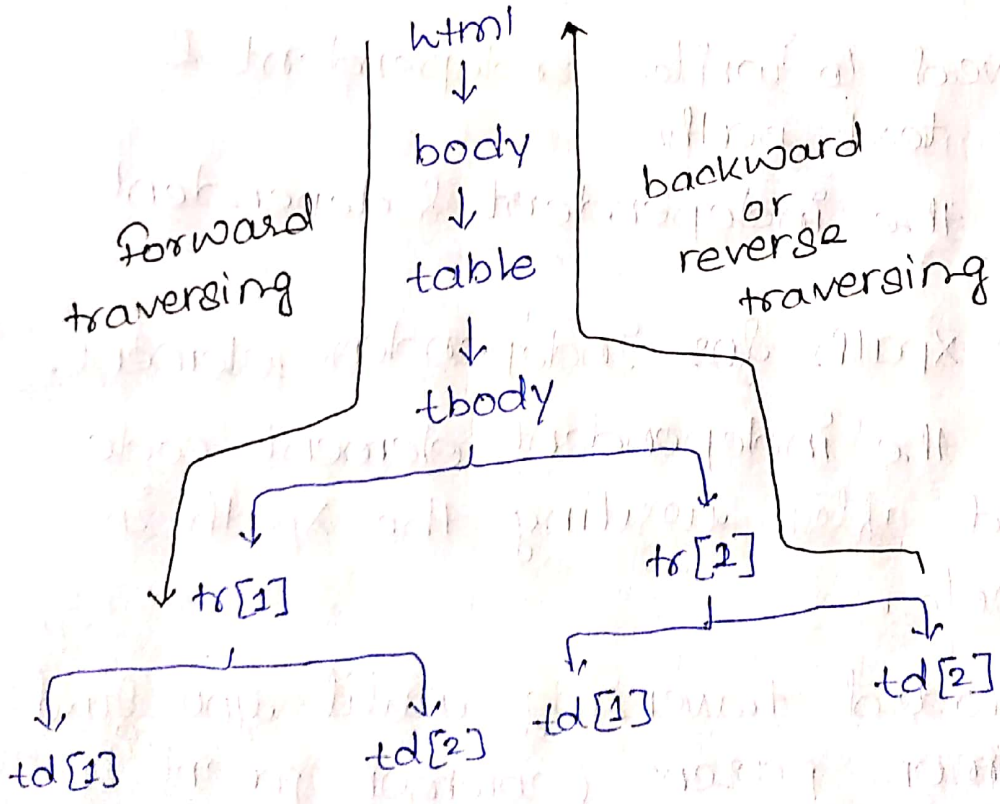
To overcome this, we can provide a substring of text present in the element.

Forward & Backward Traversing (or) Reverse Traversing

Whenever we move from one element to another element, in the html i.e. only called as traversing.

Whenever in the html, we move from parent element to the child element, it is only referred as forward traversing.

Whenever in the html, we move from child element to the parent element, then it is only referred as Backward traversing or Reverse Traversing.



Forward traversing

//tbody/tr[1]/td[2] ⇒ tbody → 300

//table/tbody/tr[2]/td[2] ⇒ table → 700

Backward traversing

//td[text() = '300']/../.. ⇒ 300 → tbody

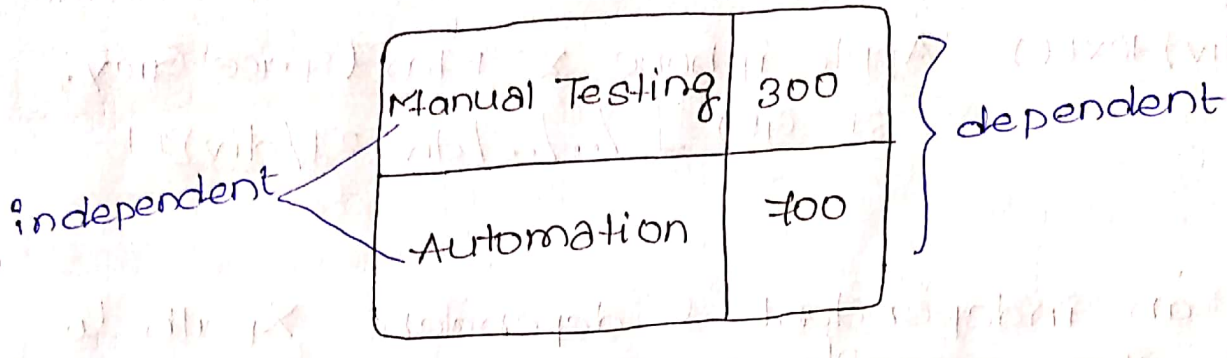
//td[text() = '700']/../../..

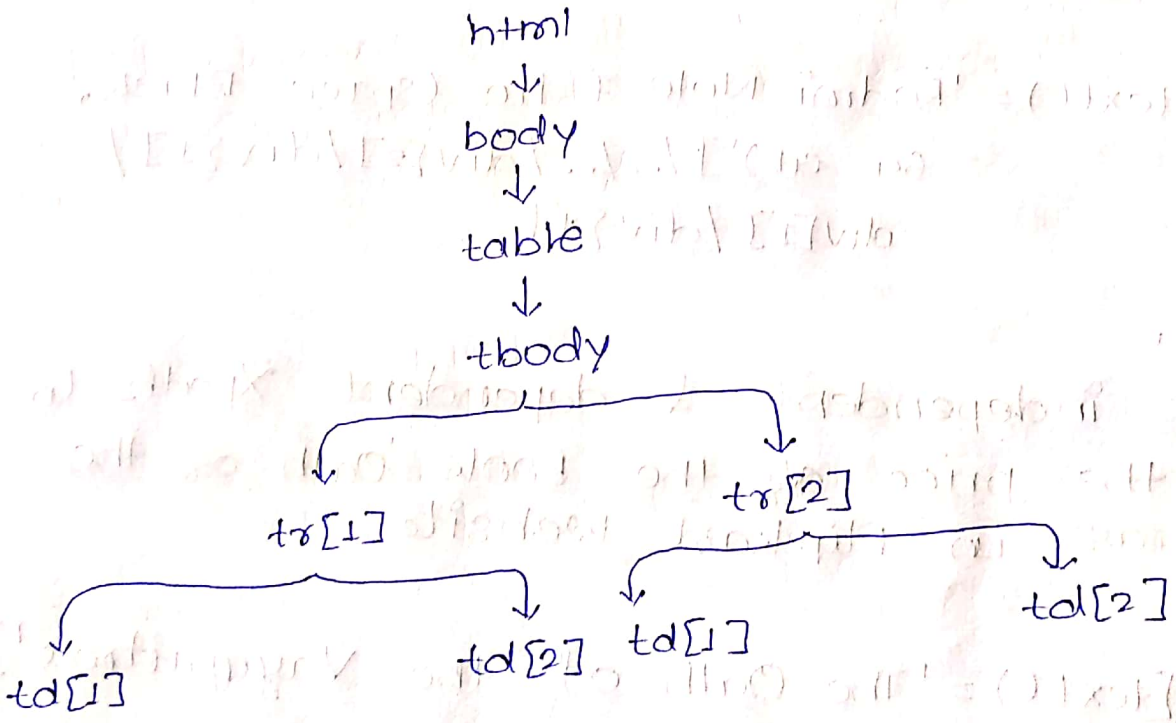
** Handling Dynamic Elements (or) Duplicate Elements (or) Independent and Dependent Text Path

Steps involved to write independent & dependent text path

1. Identify the independent & dependent element
2. Write the xpath for independent element.
3. Click on the independent element code highlighted after writing the xpath in the console.
4. Do backward traversing until you find the common parent (common parent is nothing but an element which highlights the both the independent & dependent elements), update in Xpath.

⑤ From the common parent, do forward traversing until the dependent element is highlighted.





//td[text()= 'Manual Testing']

Write an independent & dependent Xpath to match the price of Apple Iphone XS Max in Flipkart Website.

```
//div[text()='Apple iphone XS Max (Space Grey, 512 GB)']/../../div[2]/div[1]
```

Write an independent & dependent Xpath to match the price of Redmi Note 7 pro in Flipkart Website.

```
//div[text()='Redmi Note 7 Pro (Space Black, 64 GB)']/../../div[2]/div[1]/div[1]/div[1]
```

Write an independent & dependent Xpath to match the price of the Book 'Oath of the Vayuputras' in Flipkart Website.

```
(//a[text()='The Oath of the Vayuputras'])[2]/../a[3]/div[1]/div[1]
```

Note:-

Whenever we write an Xpath and it identifies multiple elements, if we want to identify only one element, then we can use the concept of group by index

Syntax :- Groupby index

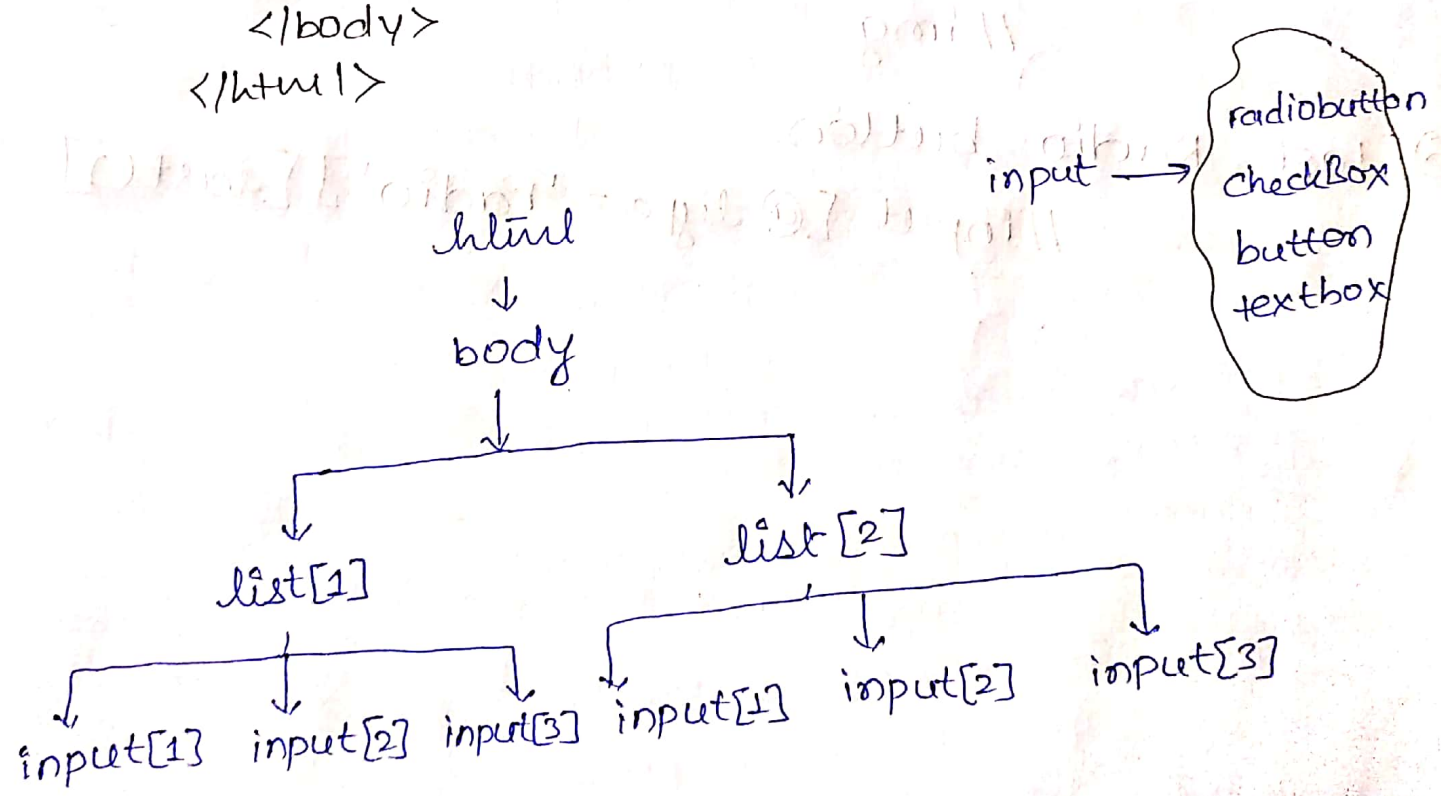
```
(Xpath)[index]
```

22nd Mar '19

From the given below front-end, identify all the radio buttons.

○male ○female ○others ○1st sem ○2nd sem ○3rd sem

```
<html>
  <body>
    <list>
      <input type="radio"> male
      <input type="radio"> female
      <input type="radio"> others
    </list>
    <list>
      <input type="radio"> 1st sem
      <input type="radio"> 2nd sem
      <input type="radio"> 3rd sem
    </list>
  </body>
</html>
```




```
// input[@type='radio']
```

Xpath to match

① All the first radio buttons

```
//input[@type = 'radio'][1]
```

② All the second radio buttons

```
//input[@type = 'radio'][2]
```

③ All the checkboxes

```
//input[@type = 'checkbox']
```

④ All the links present in a website

```
//a
```

⑤ All the images

```
//img
```

⑥ Last Radio button

```
//input[@type = 'radio'][last()]
```

Synchronization or wait in Selenium

Synchronization is a process of matching the speed of selenium with the software under test or application under test.

There are two types of wait :-

a) Implicit wait
b) Explicit wait

Implicit wait

Automate the below scenario

1. Navigate to magento.com
2. click on My Account
3. Enter the Email & password
4. click on Login
5. click on Logout

```
public class Launch
{
    p s v m (string[] args)
    {
        WebDriver driver = new chromeDriver();
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait
                        (20, TimeUnit.SECONDS);
        String url = "https://magento.com";
        driver.get(url);
        WebElement myacct = driver.findElement
                        (By.linkText("My Account"));
        myacct.click();
        WebElement email = driver.findElement(By.
                        id("email"));
```

```
email.sendKeys("vineetanand61@gmail.com");
WebElement pwd = driver.findElement(By.id("pass"));
pwd.sendKeys("Welcome123");

WebElement login = driver.findElement(By.id("
                                        send2"));
login.click();
WebElement logout = driver.findElement(By.
                                    linkText("Log Out"));

logout.click();
}
}
```

Notes :-

* Generally Thread.sleep is not recommended in Industry, while writing the test code. Because, when we use Thread.sleep in our test code, then Selenium will wait or sleep for the specified time and then it will perform the action.
* Even if the element is loaded within 5 sec, then also Selenium would forcefully wait for the specified time.

This problem can be overcome by using Implicit wait which can be applied using the below syntax :-

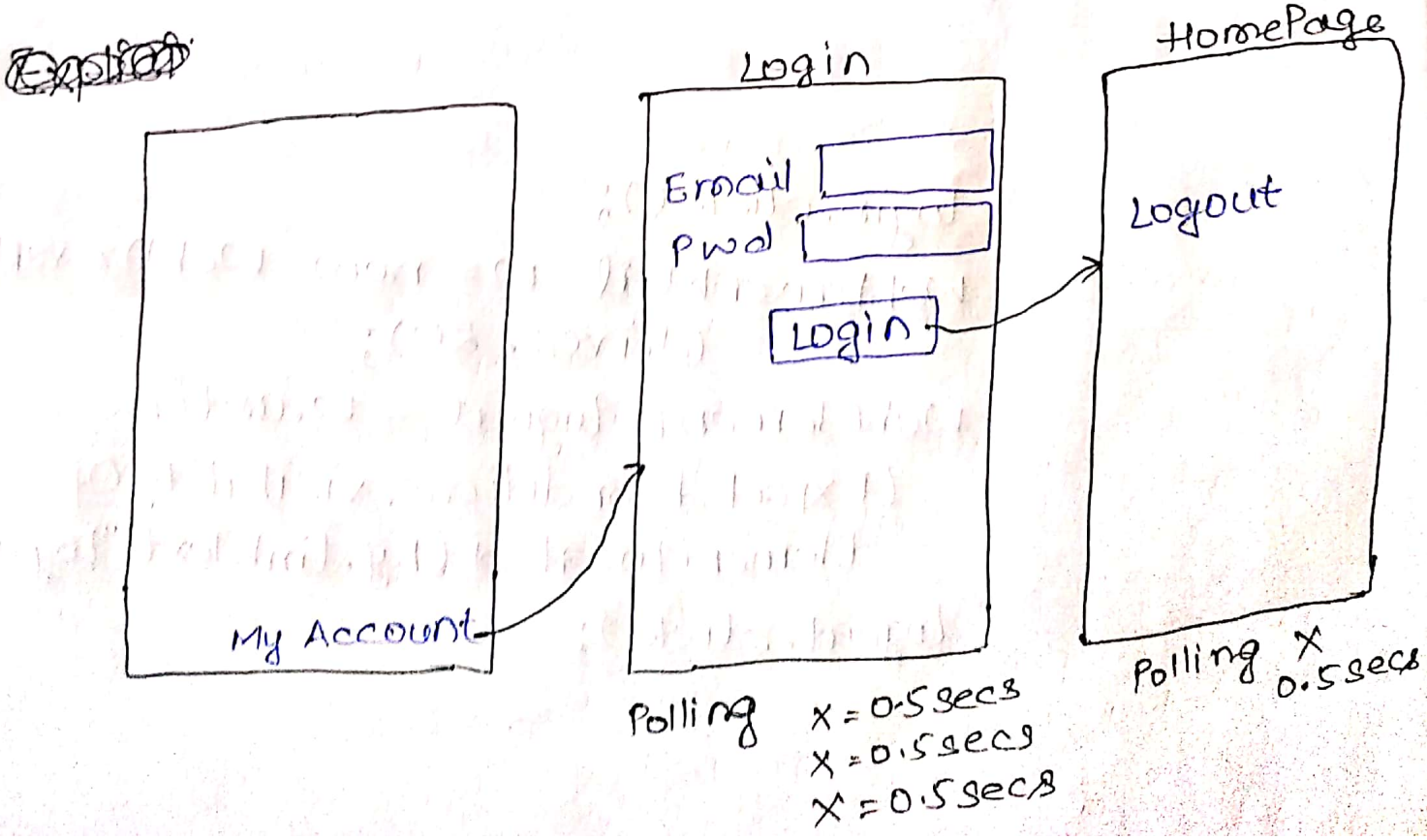
```
driver.manage().timeOuts().implicitly
        wait(20, TimeUnit.SECONDS);
```

* In the above syntax, the Implicit wait is for 20 seconds, it doesn't mean that the Selenium will wait completely for 20 sec to perform action rather for every 0.5 secs. Polling would occur & Selenium will maximum wait for 20 seconds.

* If the web element is loaded within 20 secs then immediately action would be perform & if even after 20 seconds, the web element is not loaded then NoSuchElementException would be displayed.

* Implicit wait is applied on all the find Elements method.

The default polling interval is 0.5 seconds.



Explicit Wait

```
public class Launch
{
    p s v m (String [] args)
    {
        WebDrives driver = new ChromeDriver();
        drives.. manage().window().maximize();
        drives.manage().timeOuts().
                implicitlyWait(20, TimeUnit.SECONDS);
        String url = "https://magento.com";
        driver.get(url);
        WebElement myacct = driver.findElement
                (By.LinkText("My Account"));
        myacct.click();

        — —
        — —
        — —
        — —

        login.click();
        WebDriverWait w = new WebDriverWait
                (driver, 20);
        WebElement logout = w.until
          (ExpectedConditions.visibilityOf
           ElementLocated(By.linkText("Log out"));
        logout.click();

}
```

Implicit Wait	Explicit Wait
1> The TimeUnit is Days, Hours, minutes, seconds, microseconds etc.	The TimeUnit is in seconds.
2> Implicit wait can be applied on findElement() and findElements() method	The Explicit wait can be applied on any method.
3> If the element is not identified then NoSuchElement Exception would be displayed.	If the webElement is not identified then TimeOut Exception would be displayed.
4> No conditions are specified.	conditions are specified.

WAS to launch the Firefox browser and reduce the size of browser by 50%.

```
public class launch {

    p s v m (String [] args)
    {
        WebDriver driver = new FirefoxDriver();
        Dimension dim = driver.manage().window().
                        getSize();
        S.O.P("Height is : " + dim.getHeight());
        S.O.P("width is : " + dim.getWidth());

        Dimension ndim = new
            Dimension (dim.getWidth()/2,
                       dim.getHeight()/2);

        S.O.P(ndim.height);
        S.O.P(ndim.width);
        driver.manage().window().setSize
                                (ndim);

    }
}
```

25th Mar '19

Write a note on get() methods or Write a script to demonstrate

① getTagName()
② getAttribute("attribute name")
③ getText()

Scenario to be Automated

Open chrome Browser
Go to Magento.com
click on My Account
Get the tagName of my account
Get the href value of my account link
Display the text of my account Link
Perform the action using get

```
class Launch
{
    p s v m (String args[])
    {
        WebDriver driver = new ChromeDriver();
        drives.manage().window().maximize();
        String url = "https://magento.com";
        driver.get(url);
        WebElement myacct = driver.findElement(
                            By.linkText("My Account"));
        String tagName = myacct.getTagName();
        String href_value = myacct.getAttribute("href");
        String text = myacct.getText();
        S.O.P(tagName);
        S.O.P(href_value);
        S.O.P(text);
        driver.close(); {
```

WAS to demonstrate boolean functions

There are three boolean functions state below

① isSelected()
② isDisplayed()
③ isEnabled()

Scenario to be Automated :-

a.) Navigate to www.cleartrip.com

b.) check whether round trip is selected or not if it is selected give a message round trip is selected if not, select it

c.) Once round trip is selected, check whether the ~~return~~ Return on date is displayed or not, if displayed, send some date, if not then give a message that Return on is not displayed.

```
public class Launch
{
    p s v m (String args[])
    {
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        String url = "https://www.cleartrip.com";
        driver.get(url);
```

```
WebElement roundtrip = driver.findElement(By.id
                                ("Round Trip"));

if (roundtrip.isSelected() == true)
  {
    S.O.P ("round trip is selected");
  }
else
  {
    roundtrip.click();
  }
Thread.sleep (4000);
WebElement returndate = driver.findElement
            (By.id ("ReturnDate"));
if (returndate.isDisplayed() == true)
  {
    returndate.sendKeys ("Wed, 24 Apr, 2019");

  }
else
  {
    S.O.P("return date is not displayed");
  }
driver.close();

{
```

WAS to demonstrate isEnabled() method.

```
public class Demo
{
    p s v m (String args[])
    {
        WebDriver driver= new ChromeDriver();
        driver.manage().window().maximize();

        String url= "https://account.magento.
                com/customer/account/create/";

        driver.get(url);
        WebElement submit = driver.findElement
                (By.id("registerSubmit"));

        S.O.P(submit.isEnabled());
        Thread.sleep(40000);
        S.O.P(submit.isEnabled());

    }
}
```

o/p:- false
false *

* (if not fill the form in given time)

WAS to count the total no. of links present in the magento application. Display the total no. of links and link.text of each link..

```
class Demo
{
    p s v m (String. args[])
    {
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        String url = "https://magento.com";
        driver.get(url);
        List<WebElement> links = driver.
                findElements(By.tagName("a"));
        int linksize= links.size();
        S.O.P(linksize);
        for(int i=0 ; i<linksize ; i++)
        {
            String text = links.get(i).getText();
            S.O.P(text);
        }
        driver.close();
    }
}
```

O/P :- 243

The same above program can also be written using for each loop. which is given below :-

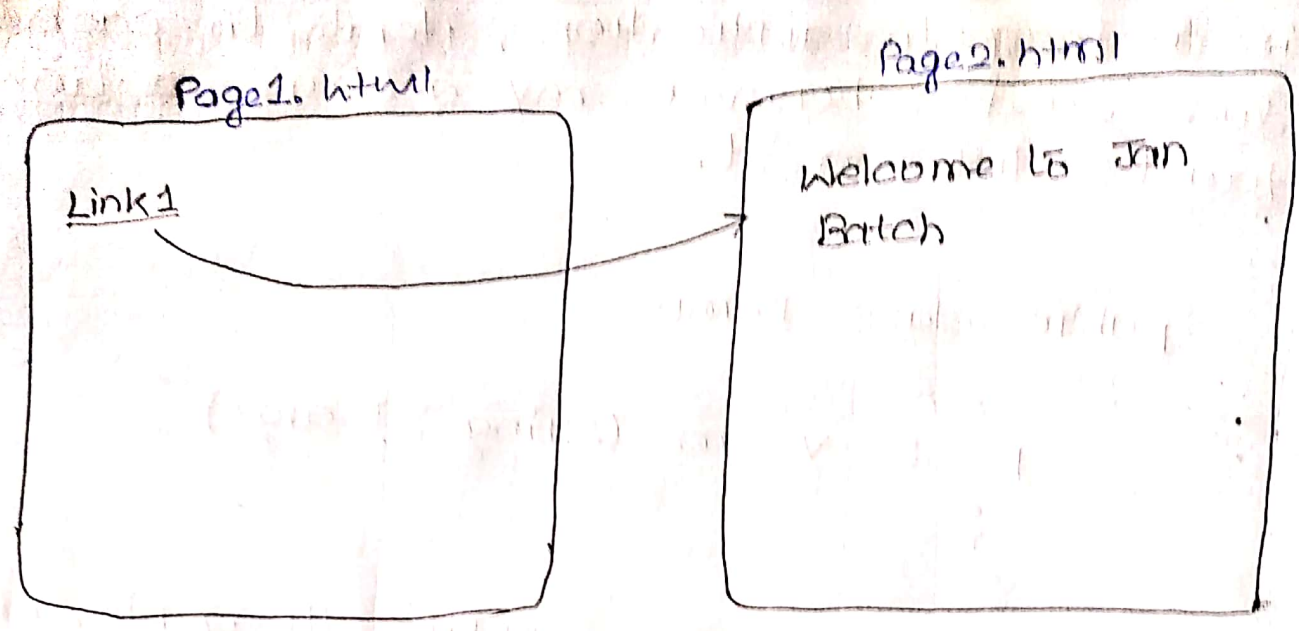
```
for (WebElement e : links)
  {
      s.o.p (e.getText());
  }
```

Differences between findElement() and findElements().

findElement	findElements
① It is used to find only one element.	It is used to find multiple element.
② Return type of find Element is WebElement.	Return type of find Elements is List<WebElement>
③ If the element is not found then NoSuchElementException is displayed.	If the element is not found then empty list is return.

WAS to count the total no. of links present in a magento application, iterate to each link and whenever my account link is found, click on it.

```
public class Demo
{
    p s v m (String [] args)
    {
        --

        String url = "https://magento.com";
        driver.get(url);
        List<WebElement> Links = driver.
            findElements (By.tagName("a"));
        int linksize = Links.size();
        S.O.P(linksize);

        for (WebElement l : Links)
        {
            String text = l.getText();
            S.O.P(text);
            if (text.contains("My Account"))
            {
                l.click();
                break;
            }
        }
    }
}
```



Page1.html

```
<html>
  <body>
    <a href = "page2.html" target = "blank"> Link1.
                                            </a>
  </body>
</html>
```

Page2.html

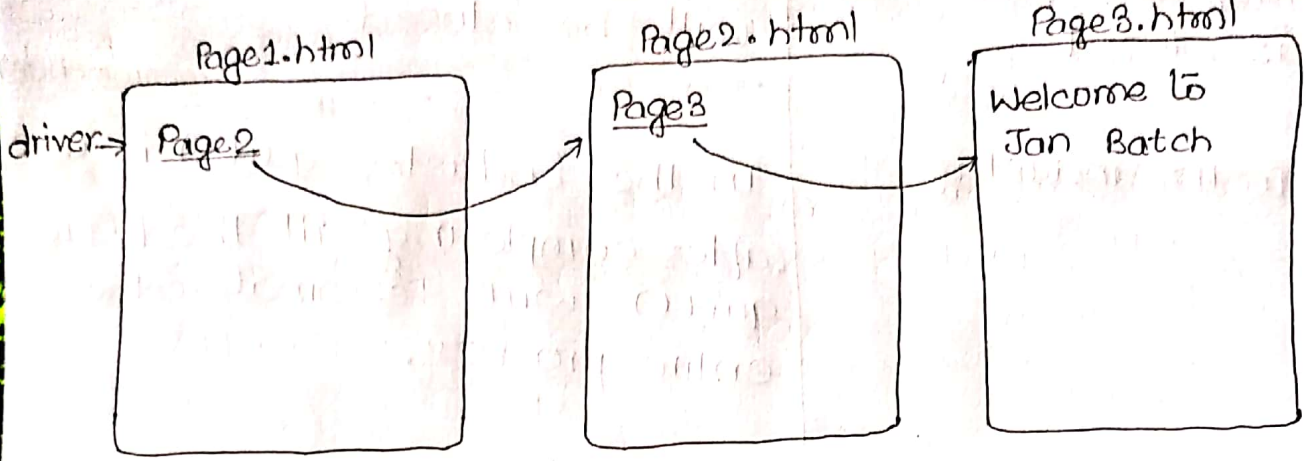
```
<html>
  <body>
    <h2> Welcome to Jan Batch </h2>
  </body>
</html>
```

Difference b/w close() & quit()

close()	quit()
Only the parent tab is closed.	Both parent & child tab will be closed.
~~In the industry~~	In the industry quit() is recommend, after completing all the task quit() will terminate the entire process.

26th Mar '19

Switching from one tab to another tab (or) Switching from one window to another window



Page1.html

```
<html>
  <body>
    <a href = "page2.html" target = "blank">Page 2</a>
  </body>
</html>
```

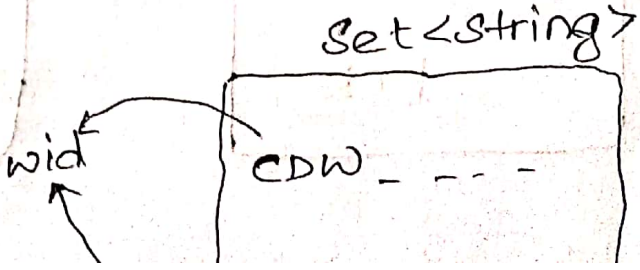
Page2.html

```
<html>
  <body>
    <a href = "page3.html" > Page3 </a>
  </body>
</html>
```

Page3.html

```
<html>
  <body>
    <h2> Welcome to jan Batch </h2>
  </body>
</html>
```

```
class Launch
{
    p s v m (String [] args)
    {
     WebDriver drives = new ChromeDriver();
     driver.get("file//..");
     WebElement page2 = driver.findElement(
                        By.linkText("Page2"));

     page2.click();

     Set<String> handles = driver.getWindow
                           Handles();

     S.O.P(handles);

     for(String wid : handles)
       {
         driver.switchTo().window(wid);
       }

     WebElement page3 = driver.findElement
            (By.linkText("page3"));

     page3.click();

     driver.quit();

    }

}
```

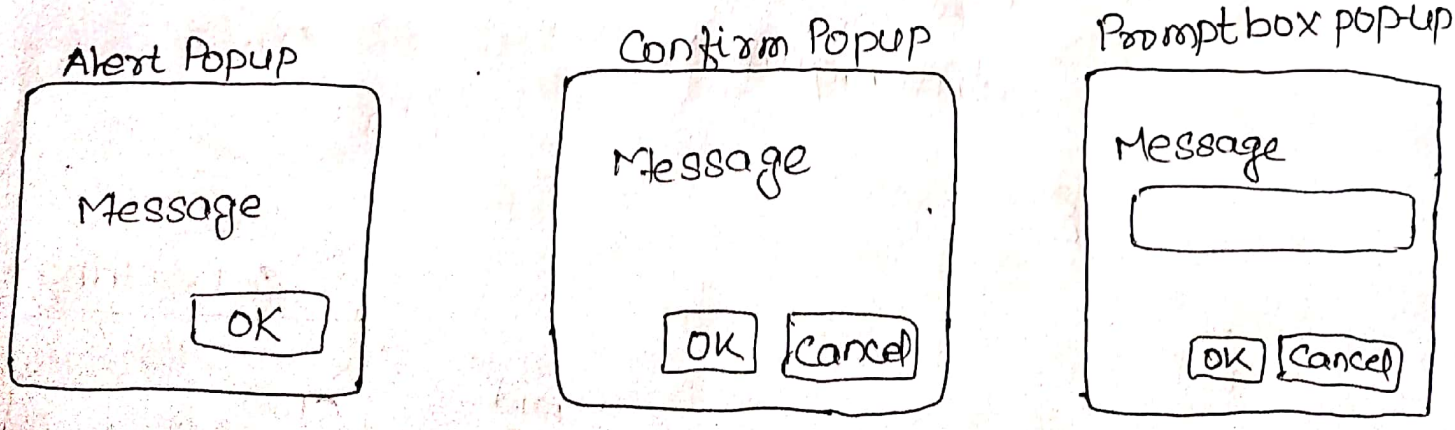


Note :-

* In the above program, the for loop is design in such a way that it will always refer to the last tab.
* If you want to switch for specific tab, then the above program is not suitable.
* In order to switch for specific tab, instead of using for each loop, use ArrayList whose syntax is given below:-

```
Set<String> handles = driver.getWindowHandles();
S.O.P(handles);
ArrayList<String> al = new ArrayList<string>
                       (handles);
driver.switchTo().window(al.get(1));
```

Handling pop-ups



WAS to handle Alert Box PopUP

```
class AlertEx
{
    P S V m (String [] args)
    {
        driver.manage().window().maximize();
        String url = "https://echoecho.com/javascript4.
                                                    htm";
        driver.get(url);
        WebElement alert = driver.findElement(By.name
                                    ("B1"));

        alert.click();
        Alert a = driver.switchTo().alert();
        Thread.sleep(5000);
        a.accept();
        Thread.sleep(5000);
        drives.close();
    }
}
```

WAS to handle confirm Box Pop up

```
WebElement alert = driver.findElement
        (By.xpath("//input[@name='B2']"));

alert.click();
Alert a = driver.switchTo().alert();
Thread.sleep(5000);
a.dismiss();
```

WAS to handle Prompt Box pop up

```
WebElement prompt = driver.findElement
                    (By.xpath("//input[@name='B3']");
prompt.click();
Alert a = driver.switchTo().alert();
Thread.sleep(5000);
a.sendkeys("haibye");
S.o.P(a.getText());
a.accept();
Thread.sleep(5000);
```

Note:-

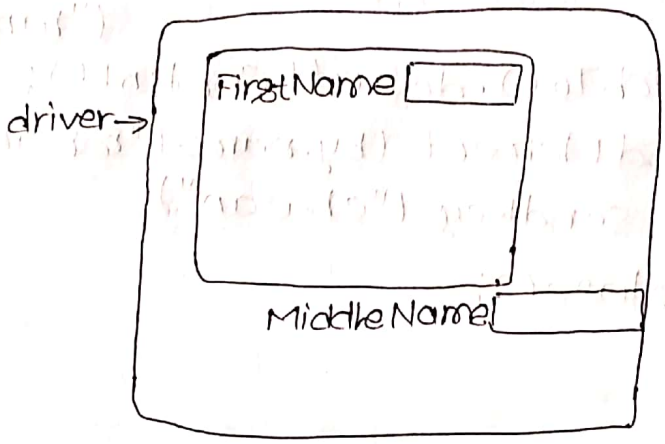
* accept() is used to click on ok button of alerts.
* dismiss() is used to click on the cancel button of alert.
* getText() is used to get the Text of an element.

Handling frames

~~In order to handle frames, we can use two~~
~~three ways :-~~
~~1) drivers switch to~~

A webpage embedded within another webpage or a webpage inside another web page is referred to as frames.

In order to create frames in html, we use iframe tag.



First.html

```
<html>
  <body>
    FirstName <input type="text" id="fname" name="fname">
  </body>
</html>
```

Middle.html

```
<html>
  <body>
  <iframe src="First.html" id="nameinfo" name="nameinfo">
  </iframe>
  MiddleName <input type="text" id="mname" name="mname">
</body>
```

```
public class Launch
{
    p s v m (String args[])
    {
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        String url="file:// ..... ";
        driver.get(url);
        driver.switchTo().frame(0);
        driver.findElement(By.id("frame")).sendKeys
                                          ("ram");
        driver.switchTo().defaultContent();
        driver.findElement(By.name("mname")).
                    sendKeys("charan");
        driver.close();
    }
}
```

Note :-

Switching to frames can be done using three ways :-

1. driver.switchTo().frame(Index);
2. driver.switchTo().frame(name/id);
3. ~~driver~~ By using Xpath

Ex :- 1. driver.switchTo().frame(0);

2. driver.switchTo().frame("nameinfo");

3.

27th Mar '19

Mouse and Keyboards Events

Mouse Event

In order to handle mouse and keyboard events follow the steps listed below :-

1. Identify the web element on which we want to perform an action.
2. Create an object of actions class.
3. To the constructor of actions class pass the reference of the browser.

Scenario to be Automated

1. Navigate to Amazon.in
2. Move to Shop by Category
3. Move to Echo & Alexa
4. Move to All new Echo Dot and click on it.

```
public class Launch
{
    P S V m (String args[])
    {
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        String url = "https://www.amazon.in";
        driver.get(url);
        WebElement shopBy = driver.findElement
                (By.xpath("//span[text()=
                'Shop by']);
```

```
WebElement echo = driver.findElement(By.xpath
        ("//span[text()='Echo & Alexa']");
WebElement echodot = driver.findElement(By.xpath
        ("//span[text()='

Actions a = new Actions(driver);
a.moveToElement(shopBy).build().perform();
Thread.sleep(5000);
a.moveToElement(echo).build().perform();
Thread.sleep(5000);
a.moveToElement(echodot).click().build().
                                perform();

Thread.sleep(5000);
driver.quit();

    }
}
```

The above program can also be written in an efficient way which is given below :-

```
class Launch
{   psvm(String[] args)
    {
        =
        =
        =
        Actions a = new Actions(driver);
        a.moveToElement(shopBy).pause(5000).
        moveToElement(echo).pause(5000).
        moveToElement(echodot).click().
        build().perform();
    {
}
```

KeyBoard Events

Scenario to be Automated

1. Navigate to www.cleartrip.com
2. Identify the from field.
3. Move to from field using mouse events and click on it.
4. Type tri in from field.
5. Press five times a downward arrow key of keyboard.

```
public class ClearTripEx
{
    p s v m (string args[])
    {
        WebDriver driver = new ChromeDriver();
        drives.manage().window().maximize();

        String url = "https://www.cleartrip.com";
        driver.navigate().to(url);

        WebElement from = driver.findElement
                (By.id("FromTag"));

        Actions a = new Actions(drives);

        a.moveToElement(from).click().sendKeys
        ("ban").pause(4000).sendkeys(keys.
         ARROW_DOWN).pause(3000):sendKeys(
         Keys.ARROW_DOWN).pause(3000).
         sendkeys(Keys.ENTER).build().perform();
    }
}
```

Scenario to be Automated

1. Navigate to www.google.com
2. Identify the Search Box & click on it using mouse events.
3. Type abcfortech in uppercase & click Enter

```
public class Launch
{
    public static void main (String [] args)
    {
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        String url = "https://www.google.com";
        driver.get(url);
        WebElement searchbox = driver.
                findElement (By.xpath
                        ("//input[@name='q']"));

        Actions a = new Actions(driver);
        a.moveToElement(searchbox).click().
        sendKeys (Keys.SHIFT).sendkeys
        ("abc for tech").pause (4000).
        sendKeys (Keys.ENTER).build().perform();
    }
}
```

Handling Tables

Manual Testing	300
Selenium	700
LeanFT	1500

```
<html>
  <body>
    <table border="1" id="tab">
      <tbody>
        <tr>
          <td> Manual Testing </td>
          <td> 300 </td>
        </tr>
        <tr>
          <td> Selenium </td>
          <td> 700 </td>
        </tr>
        <tr>
          <td> Lean FT </td>
          <td> 1500 </td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

```
Class TableDemo
{
    P S V m (String args[])
    {
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        String url = " file:// ";
        driver.get(url);
        List<WebElement> rows = driver.findElements
                (By.xpath("//table[@id='tab']/tbody/tr"));

        int rsize = rows.size();
        System.out.println(rsize);
        List<WebElement> cols = driver.findElements
                (By.xpath("//table[@id='tab']/tbody/tr[1]/
                                                    td"));

        int csize = cols.size();
        System.out.println(csize);

        for(int i=1 ; i<=rsize ; i++)
        {
            WebElement cname = driver.findElement
                (By.xpath("//table[@id='tab']/tbody/
                    tr["+i+"]/td[1]"));

            S.O.P(cname.getText());

        }

    }
}
```

① Go to https://datatables.net/extensions/select/examples/initialisation/simple.html

② From the table given in the page, print the size of rows

③ Print the size of columns

④ Extract the salary

⑤ Calculate the sum & Display it.

```
public class Launch
{
    p s v m (String args[])
    {
        WebDriver driver = new ChromeDriver();
        —
        —
        —
        —

        int sum = 0;
        for(int i=1; i<=row.size(); i++)
        {
            WebElement price = driver.findElement
                (By.xpath("//td[text()='Airi Satou']
                    /../../tr[" + i + "]/td[6]"));

            String amount = price.getText();
            String tot = amount.replace("$", "");
            String p = tot.replace(",", "");
            int d = Integer.parseInt(p);
            sum += d;
        }
        S.O.P(sum);
    }
}
```

28th Mar'19

"FrameWork"

It is a set of rules or guidelines followed by an automation test engineer during automation process in order to increase efficiency.

Framework includes three phases :-

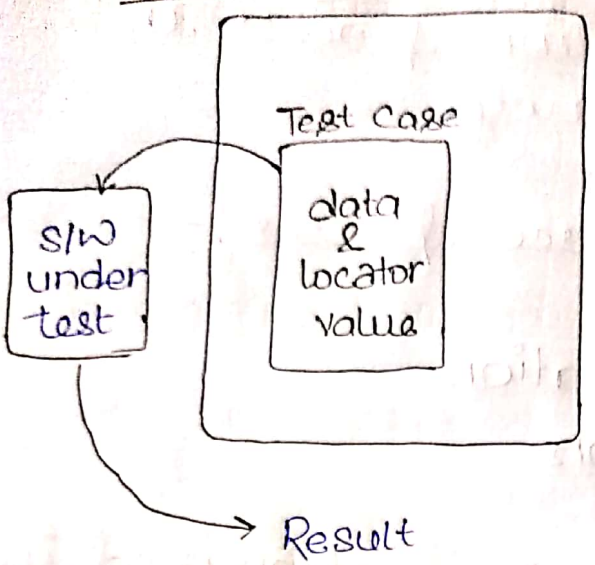
a) Framework Design
b) Framework Implementation
c) Framework Execution

Framework Design is again categorized in six phases :-

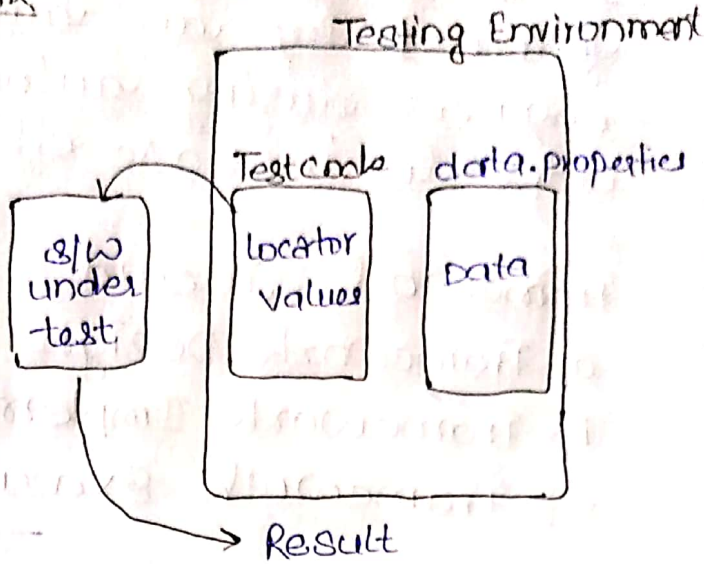
1. Data-Driven Framework
2. Keyword-Driven Framework
3. Hybrid-Driven Framework
4. PageObject Model
5. Page Factory Model
6. Test Next Generation (Text NG)

Data-Driven Framework

Non-data Driven Framework



Data-Driven Framework



Framework ~~design~~ Implementation

In this phase, the test-engineer will convert the manual test case into an automation script (test code).

Test Case id : TC0001

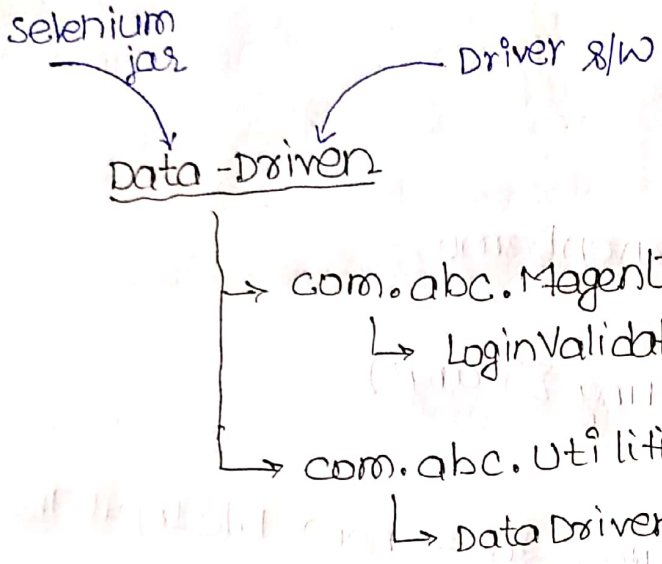
pre-requisite/ precondition :-
chrome should be installed and we should have valid log in details.

procedure :- open chrome Browser
Navigate to magento.com
click on My Account
Enter the email, Enter the password
click on login & click on logout

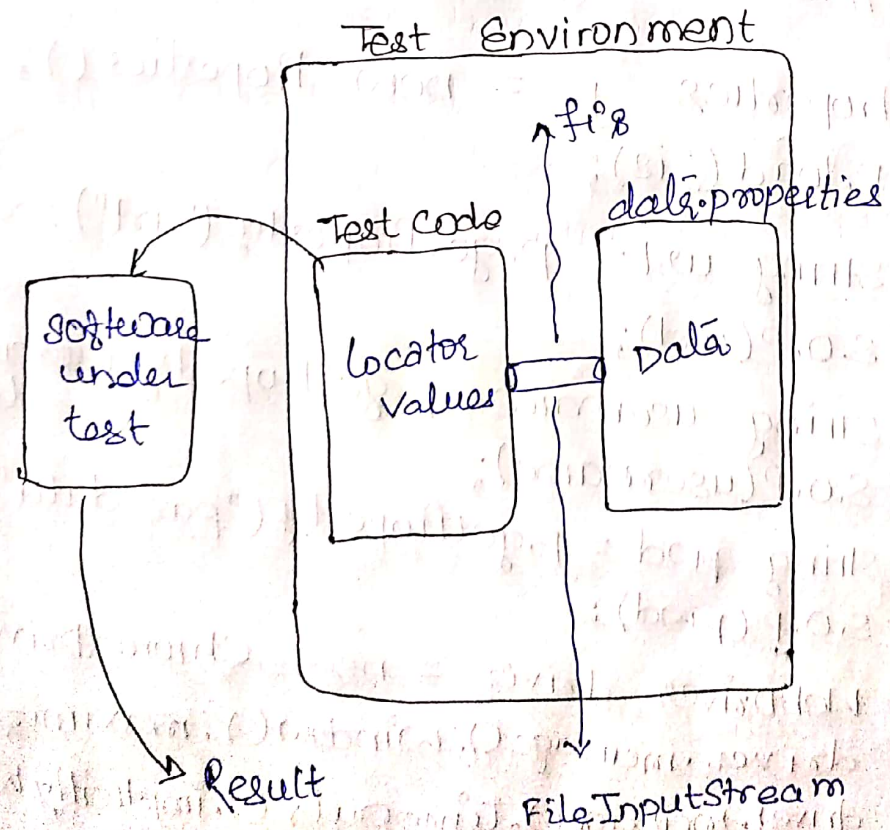
Expected o/p :- It should login & logout successfully.

Actual o/p :-

Result :-



Data-Driven framework



DataDriven.properties

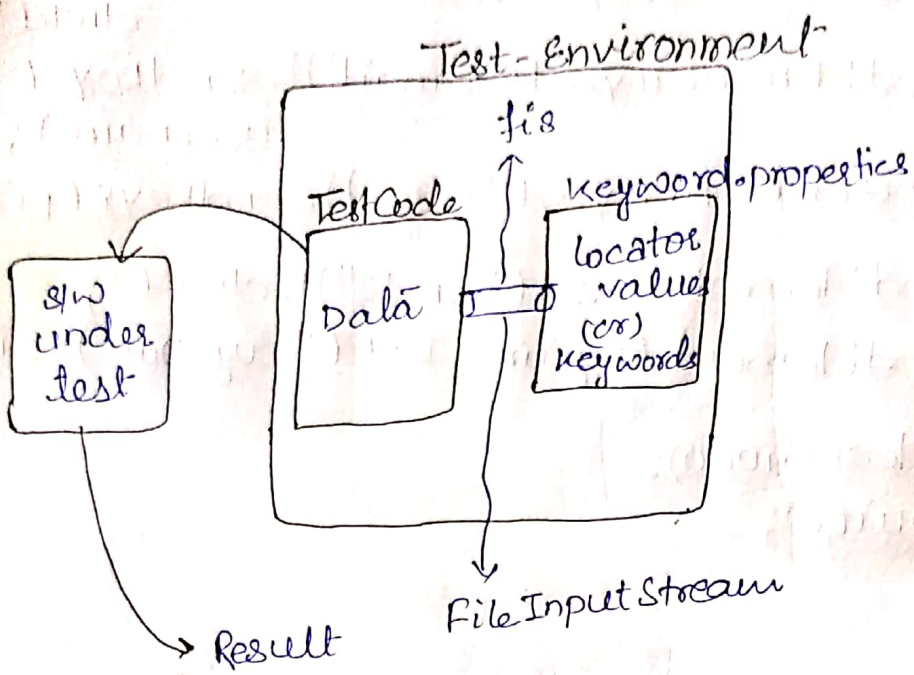
```
url = https://magento.com
emaildata = vineetanand61@gmail.com
passdata = Welcome123
```

DataDrivenDemo.java

```
public class DataDrivenDemo
{
    p s v m (String [] args)
    {
        FileInputStream fis = new FileInputStream
                              ("C://users//.");
        //location of Properties file

        Properties p = new Properties();
        p.load(fis);
        String url = p.getProperty("url");
        S.O.P(url);
        String username = p.getProperty("emailda
                                         ta");
        S.O.P(username);
        String pwd = p.getProperty("passdata");
        S.O.P(pwd);
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait
                              (20, TimeUnit.SECONDS);
```

```
driver.get(url);
driver.findElement(By.linkText("My Account")).
                                        click();
driver.findElement(By.id("email")).sendKeys(
                                        username);
driver.findElement(By.id("pass")).sendKeys(pwd);
driver.findElement(By.id("send2")).click();
driver.findElement(By.linkText("Log Out")).click();
Thread.sleep(4000);
driver.quit();
}
}
```

Keyword-Driven Framework



keyword.properties

```
myacct = //a[text() = 'My Account']
emailkeyword = email
passkeyword = pass
loginkeyword = send2
logoutkeyword = logout
```

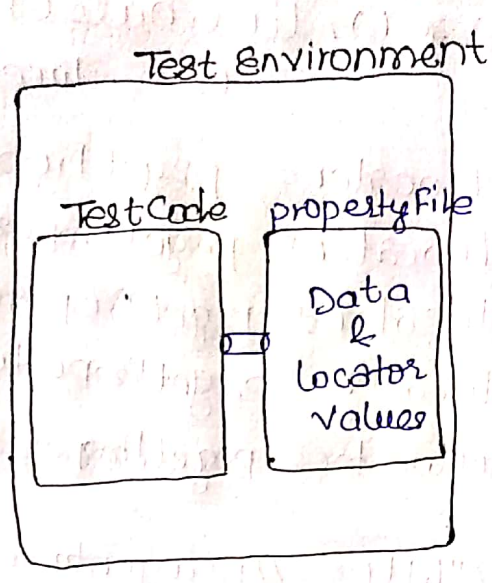
KeywordDrivenDemo.java

```
public class KeywordDrivenDemo
{
    p s v m (String [] args)
    {
        FileInputStream fis = new
            FileInputStream ("C:\\---");
        Properties p = new Properties();
        P.load(fis);
```

```
WebDriver driver = new ChromeDriver();
driver.manage().window().maximize();
driver.manage().timeOuts().implicitlyWait
                    (20, TimeUnit.SECONDS);

String acctlocator = p.getProperty("myacct");
String emaillocator = p.getProperty("emailkeyword");
String passlocator = p.getProperty("passkeyword");
String loginlocator = p.getProperty("loginkeyword");
String logoutlocator = p.getProperty("logoutkeyword");

driver.get("https://magento.com");
driver.findElement(By.xpath(acctlocator)).
                                    click();
driver.findElement(By.id(emaillocator)).
                    sendkeys("vineet@gmail.com");
driver.findElement(By.id(passlocator)).sendkeys
                    ("welcome");
driver.findElement(By.id(loginlocator)).click();
driver.findElement(By.linkText(logoutlocator)).
                    click();

Thread.sleep(4000);
driver.quit();

}

{
```

29th Mar '19

Hybrid Driven framework



Selenium jar → Hybrid-Driven ← Drivers s/w

Hybrid-Driven
- com.abc.MagentoTest
  - TestCode.java
- com.abc.utilities
  - hybrid.properties

hybrid.properties

urldata = https://magento.com
emaildata = vineetanand61@gmail.com
passdata = Welcome123
myacct = My Account
emailkeyword = email
passkeyword = pass
loginkeyword = send2
logoutkeyword = Log Out

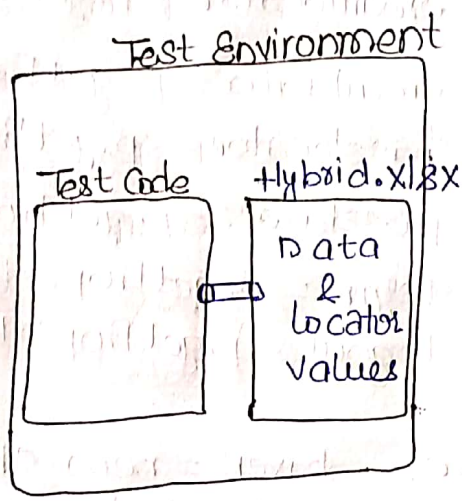
HybridDrivenDemo.java

```
public class HybridDrivenDemo
{
    p s v m (String args[])
    {
        FileInputStream fis = new FileInputStream
                              ("C:\\User\\---");
        Properties p = new Properties();
        p.load(fis);

        String url = p.getProperty("urldata");
        String myacct = p.getProperty("myacct");
        String emaillocator = p.getProperty("emailkeyw
                                            ord");
        String emaildata = p.getProperty("emaildata");
        String passlocator = p.getProperty("passkeywor
                                           d");
        String passdata = p.getProperty("passdata");
        string login = p.getProperty("loginkeyword");
        String logout = p.getProperty("logoutkeyword");

        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait
                        (20, TimeUnit.SECONDS);

        driver.get(url);
        driver.findElement(By.linkText(myacct)).clic
                                                k();
        driver.findElement(By.id(emailkeyword)).
                           sendkeys(emaildata);
        driver.findElement(By.id(passkeyword)).
                           sendkeys(passdata);
        drivers.findElement(By.id(login)).click();
        driver.findElement(By.linkText(logout)).
                                       click();
        driver.quit();
    }
}
```

Hybrid Driven Using Excel File

Steps to Download POI Jar Files

1. Open any Browser, search for Apache POI
2. click on the first link (https://poi.apache.org)
3. click on Download
4. Scroll down to the end of the Page
5. In the Release Archives, click on binary artifacts
6. Search for poi-bin-3.17-20170915.zip
7. Place the downloaded zip file in Selenium folder and extract



Hybrid-driven with Excel

- com.abc.MagentoTest
  - TestCode.java
- com.abc.Utilities
  - Hybrid.xlsx

Excel

.xls — 97-2003 — HSSFWorkBook

.xlsx — 2007-19 — XSSFWorkBook

locator values of the keyword → object

TestCaseId	procedure	Actions	Objects	data
TC0001	Open chrome Browser	launch		https://magento.com
TC0002	Navigate to Magento.com	navigate		
TC0003	Click on my account	click	//a[text()='My Account']	
TC0004	Enter the email	type	//input[@title='Email']	vineetamand61@gmail.com
TC0005	Enter the password	type	//input[@title='Password']	Welcome123
TC0006	Click on login	click	//button[@type='submit']	
TC0007	Click on logout	click	//a[text()='Log Out']	
TC0008	close the browser	quit		

Steps to add poi jars :-

1. Right click on the project, select build path
2. click on configure build path
3. click on Add External jars
4. Navigate to the poi folder, present in your Selenium Components
5. Select all the jars present in the folder & click on open
6. Again click on add external jars open lib folder in poi folder, select all the jars, click on open
7. Again add the jars present in ooxml-lib folder & following above steps
8. click on Apply & close.

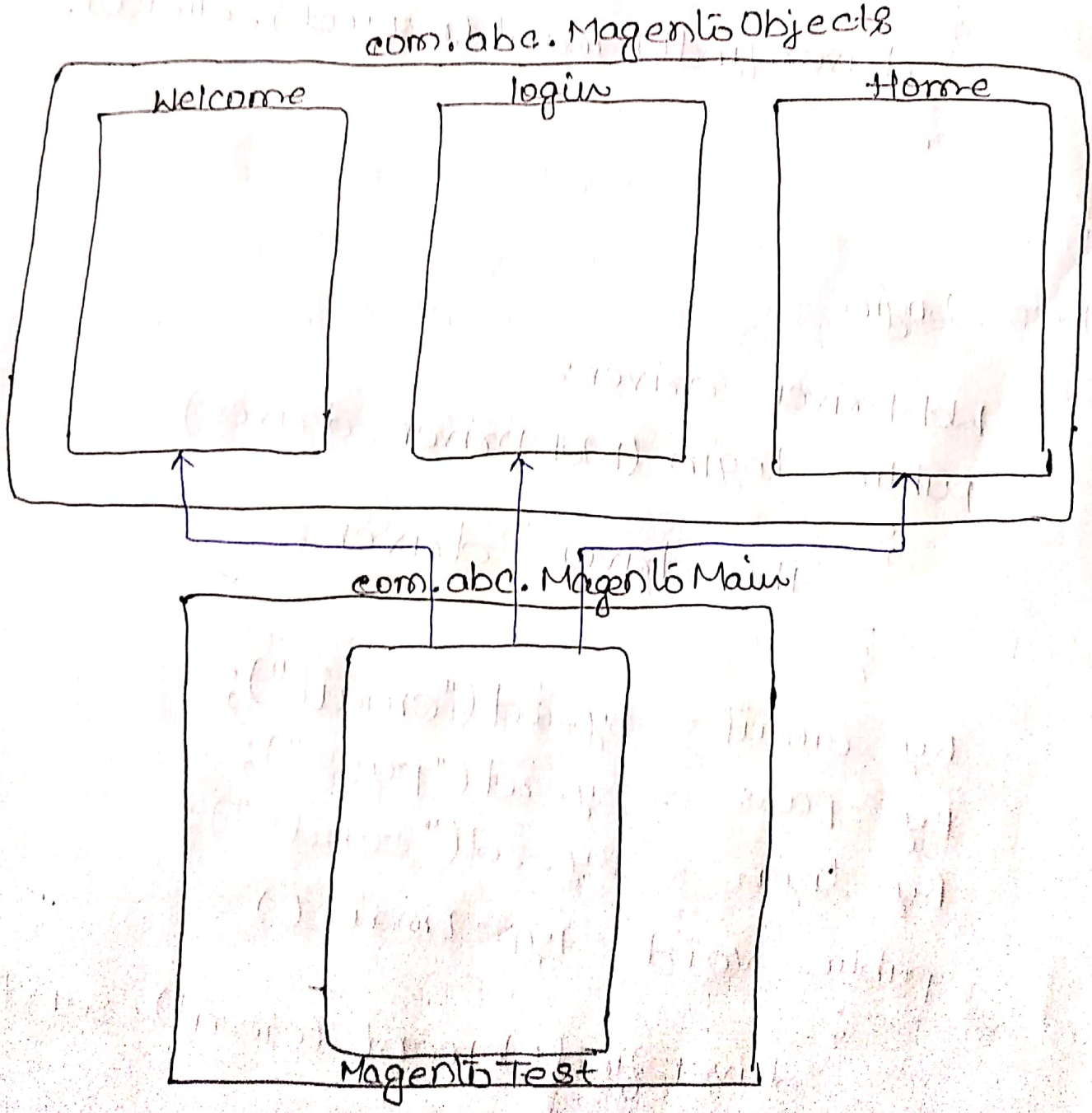
```
public class MagentoTest {
    public static XSSFSheet sheet;
    public static WebDriver driver;
    public static String getCellValue(int rown, int coln)
    {
        XSSFRow row = sheet.getRow(rown);
        XSSFCell cell = row.getCell(coln);
        String data = cell.getStringCellValue();
        return data;
    }
    public static void main(String[] args)
    {
        FileInputStream fis = new FileInputStream
                ("C:\\User\\----");
        XSSFWorkbook book = new XSSFWorkbook
                (fis);
        sheet = book.getSheetAt(0);
        int numberOfRows = sheet.getPhysicalNumber
                OfRows();
        S.O.P(numberOfRows);
        for(int i=0; i< numberOfRows ; i++)
        {
            String action = getCellValue(i,2);
            S.O.P(action);
            switch(action)
            {
                case "Launch":
                    WebDriver driver = new Chrome
                            Driver();
                    driver.manage().window().
                            maximized
                    driver.manage().timeOuts().
                     implicitlyWait(20, TimeUnit.SECON
                            S);
```

```
        break;
        case "navigate.":
            driver.get(getCellValue(i,4));
        break;
        case "click":
            driver.findElement(By.xpath(getCellValue(i,3)).
                                        click();
        break;
        case "type":
            driver.findElement(By.xpath(getCellValue(i,3));
                        sendkeys(getCellValue(i,4));

        break;
        case "quit":
            driver.quit();
        break;
    }
}
```

01st April '19

Page Object Model

Page Object Model
- com.abc.Magenlib Objects
  - Welcome.java
  - Login.java
  - Home.java
- com.abc.MagenlibMain
  - MagenlibTest.java



```
class Welcome
{
    WebDriver driver;
    public Welcome (WebDriver driver)
        {
            this.driver = driver;
        }
    By myacct = By.linkText("My Account");
    public void clickOnMyAccount()
      {
        driver.findElement(myacct).click();
      }
}

class Login
 {
    WebDriver driver;
    public Login (WebDriver driver)
       {
          this.driver = driver;
        }
    By email = By.id("email");
    By pass = By.id("pass");
    By login = By.id("send2");
    public void typeEmail()
        {
          driver.findElement(email).sendKeys("");
        }
```

```
    public void typePassword()
    {
        driver.findElement(pass).sendKeys("");
    }
    public void clickOnlogin()
    {
        driver.findElement(login).click();
    }
}

class Home
{
    WebDriver driver;
    public Home(WebDriver driver)
    {
        this.driver = driver;
    }
    By logout = By.linkText("Log Out");
    public void clickonLogout()
    {
        driver.findElement(logout).click();
    }
}

class MagentoTest
{
    P S V m(String[] args)
    {
        WebDriver driver = new ChromeDriver();
        driver.get("https://magento.com");
```

```
Welcome w = new Welcome(driver);
    w.clickOnMyAccount();

Login l = new Login(driver);
    l.typeEmail();
    l.typePassword();
    l.clickOnLogin();

Home h = new Home(driver);
    h.clickOnLogout();

}

{
```

Page Factory Model

```
class Welcome
{
    WebDriver driver;
    public Welcome (WebDriver driver)
    {
        this.driver = driver;
        PageFactory.initElements (driver, this);
    }
    @FindBy (linkText = "My Account")
    WebElement myacct;
    public void clickOnMyAccount()
    {
        myacct.click();
    }
}
Class Login
{
    WebDriver drives;
    public login (WebDriver drives)
    {
        this.driver = drives;
        PageFactory.initElements (drives, this);
    }
    @FindBy (id = "email")
    WebElement email;
    @FindBy (id = "pass")
    WebElement pwd;
    @Find By (id = "send2")
    WebElement login;
```

```
    public void typeEmail()
      {
        email.sendKeys("----");
      }
    public void typePassword()
      {
        pwd.sendKeys("----");
      }
    public void clickOnLogin()
      {
        login.click();
      }
}
class Home
{
    WebDriver driver;
    public Home(WebDriver driver)
      {
        this.driver = driver;
        PageFactory.initElements(driver, this);
      }
    @FindBy(linkText = "Log out")
      WebElement logout;
    public void clickOnLogout()
      {
        logout.click();
      }
}
```

```
class MagenloTest
{
    p s v m (String args[])
    {
        WebDriver driver = new ChromeDriver();
        driver.get("https://magenlo.com");
        Welcome w = new Welcome(driver);
        w.clickOnMyAccount();

        Login l = new Login(driver);
        l.typeEmail();
        l.typePassword();
        l.clickOnLogin();

      Home h = new Home(driver);
        h.clickOnLogout();

    }

}
```

Test Next Generation (Text NG)

It is a framework which is used to test the software under test & obtain the report.

Steps involved in installing Test NG Pluging

① In the Eclipse, click on the help button in menu bar.

② click on Eclipse Marketplace / Install New S/w.

③ In the ~~find~~ work with ~~text~~ box, type ~~testng~~ ~~and click enter~~.

③ In the work with text box, type https://beust.com/eclipse/

④ Click on Add, in the name box type TestNG and in the location box type the same url as above.

⑤ Click on Add.

⑥ Once the searching process is done, it will display TestNG in the Name text box.

⑦ Click on the check Box

⑧ click on Next & Click on Terms & Cond.

⑨ click on Finish

Steps involved downloading TestNG jars

1. Search for MVN Repasitory in chrome Browser
2. Click on first Link (https://mvnrepository.com)
3. In search box, type TestNG, click on search.
4. Click on first Link (TestNG)
5. Click on 6.14.3
6. ~~Under~~ in files click on jar
7. Place the downloadable jar file in selenium component folder
8. Add the jar file to the project.

02nd April '19

Program

```
class Script
{
    @Test
    public void positiveCrea1()
    {
        WebDriver driver = new chromeDriver();
        drives.get("https://magenlo.com");
        drives.findElement(By.LinkText("My Account")).
                                        click();
        driver.findElement(By.id("email")).sendkeys
                                        ("__");
        drives.findElement(By.id("pass")).sendKeys
                                        ("__");
        drives.findElement(By.linkText("Log Out")).
                                        click();
        driver.close();
    }
    @Test
    public void positive Crea2()
    {
        WebDrivers driver = new FirefoxDriver();
        — —
        — —
        — —
        — —
        — —
        — —
    }
}
```

Note:-

* A testNG class consist of test methods
* Any method which is annotated using @Test is referred as test method.
* A testNG class can have any number of test methods
* A test method is also referred to as a test case.

Reports Generated By TestNG

→ In order to see the report, once the testNG class is executed
Select project, right click & click on refresh
Once we refresh, one folder by name test-output would appear.
Inside the test-output folder, we have two reports, index.html & second one emailable report.html

@BeforeMethod

@BeforeMethod

Such methods which are annotated using @BeforeMethod will be executed before the execution of each test case or test method.

@After.Method

Such methods which are annotated using @After.Method will be executed after the execution of each test case or test method.

Program

```
class Script2
{
    @BeforeMethod
    public void beforeMethod()
    {
        S.O.P("Before method executed");
    }
    @After.Method
    public void after.Method()
    {
        S.O.P("after method executed");
    }
}
```

```
@Test
  public void test1()
    {
      S.O.P("Test 1 Method executed");
    }
@Test
  public void test2()
    {
      S.O.P("Test 2 Method executed");
    }
}
```

Program

```
class Script3
{
    WebDriver driver;
    @BeforeMethod
    public void beforeMethod()
    {
        driver = new chromeDriver();
        driver.manage().window().maximize();
        driver.manage().timeouts().
                implicitlyWait(20, TimeUnit.SECONDS);
        driver.get("https://magenlo.com");
    }

    @AfterMethod
    public void afterMethod()
    {
        driver.close();
    }

    @Test
    public void positiveCrea1()
    {
        driver.findElement(By.linkText("My Account")
                ).click();
        driver.findElement(By.id("email")).
            sendKeys("vineetanand61@gmail.com");
        driver.findElement(By.id("pass")).
            sendKeys("Welcome123");
        driver.findElement(By.id("send2")).click();
```

```
    drives.findElement(By.linkText("Log Out")).click();
  }
  @Test
    public void positiveCrea2()
    {
      drives.findElement(By.linkText("MyAccount")).
                                              Click();
      drives.findElement(By.id("email")).sendkeys
          ("mzaidurrahman@gmail.com");
      ---
      ---
      ---
    }
}
```

@BeforeClass

Such methods which are annotated using @BeforeClass will be executed only once before the execution of before method in the current class.

@AfterClass

Such methods which are annotated using @AfterClass will get executed only once after the execution of after method in the current class.

Program

```
public class Script4
{
    @BeforeClass
    public void beforeclass()
    {
        S.O.P("before class method executed");
    }
    @After Class
    public void after class()
    {
        S.O.P("after class method executed");
    }
    @BeforeMethod
    public void beforeMethod()
    {
        S.O.P("Before method executed");
    }
    @AfterMethod
    public void afterMethod()
    {
        S.O.P("after method executed");
    }
    @Test
    public void Test1()
    {
        S.O.P("Test1 method executed");
    }
```

```
@Test
public void test2()
{
    S.O.P("Test2 method executed");
}
```

Priorities in TestNG :-

In TestNG, the order of execution is based on alphabetical order or ASCII value.

However, we can change the order of the execution based on our requirements using the concept of priority.

Syntax :-

```
@Test (priority = number)
```

Test Method without Priority

```
public class Script5
{
    @Test
    public void z-method()
    {
        S.O.P("z method executed");
    }
    @Test
    public void d-method()
    {
        S.O.P("d method executed");
    }
    @Test
    public void b-method()
    {
        S.O.P("b method executed");
    }
    @Test
    public void a-method()
    {
        S.O.P("a method executed");
    }
    @Test
    public void c-method()
    {
        S.O.P("c method executed");
    }
}
```

O/P:- a method executed
b = =
c = = z method executed

Test Method with Priority

```
public class Script6
{
    @Test (priority=0)
    public void z-method()
    {
        s.o.p("z method executed");
    }
    @Test(priority = 2)
    public void d-method()
    {
        s.o.p("d method executed");
    }
    @Test (priority =33)
    public void b-method()
    {
        s.o.p("b method executed");
    }
    @Test (priority = 99)
    public void a-method()
    {
        s.o.p("a-method executed");
    }
    @Test(priority = 3)
    public void c-method()
    {
        s.o.p("c method executed");
    }
}
```

o/p:-

```
z method executed
d method executed
c method executed
b ---  ---
a ---  ---
```

Multiple Test method with same priority

Whenever we have multiple test methods with the same priority then the order of the execution will be, the methods which have same priority, then it will check for alphabetical order & it will execute based on alphabetical order.

```
public class Script7
{
    @Test (priority =0)
    public void z-method()
    {
        S.O.P ("z method");
    }
    @Test(priority =2)
    public void d-method()
    {
        S.O.P("d method");
    }
    @Test(priority =2)
    public void b-method()
    {
        S.O.P(" b method");
    }
}
```

o/p:-
```
z method
b method
d method
```

Test methods with and without priority

In a TestNG class, if we have both Test method with priority & without priority then the order of the execution would be :-

① The test methods without priority will be executed based on alphabetical order

② The test methods with priority will be executed based on priority.

```
public class Script8
{
    @Test
    public void z_method()
    { S.O.P("z method"); }

    @Test(priority=2)
    public void d_method()
    { S.O.P("d method"); }

    @Test(priority=2)
    public void b_method()
    { S.O.P("b method"); }

    @Test
    public void a_method()
    { S.O.P("a method"); }
}
```

O/P :-
```
a method
z method
b method
d method
```

TestNG Suite

TestNG Suite is nothing but collection of test cases or test methods.

Steps involved for creating TestNG suite :-

① After creating the TestNG class, select the class, right click on it,

② Go to TestNG and click on convert to TestNG

③ Select the name for the XML file & click on finish

```
<suite>
  <test>
    <classes>
      <class>
      </class>
    </classes>
  </test>
</suite>
```

Script6.java

```
public class Script6
{
    @Test
    public void positiveCrea1()
    {
        S.O.P("positive credential 1 executed");
    }
    @Test
    public void positiveCrea2()
    {
        S.O.P("positive Credential 2 executed");
    }
}
```

Script8.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM
<suite name="Suite">
    <test thread-count="5" name="Test">
        <classes>
            <class name="com.abc.TestNGTest
                Script6"/>
        </classes>
    </test>
</suite>
```

03rd April '19

Include & Exclude

Include

It is used to specify within the XML file to inform which and all test cases has to be included in the execution.

Exclude

It is used to specify within the XML file to inform which and all test cases has to be excluded in the execution.

Program

```
class Script10
{
    @Test
    public void testCase1()
    {
        S.O.P("test case 1 executed");
    }
    @Test
    public void testCase2()
    {
        S.O.P("test case 2 executed");
    }
    @Test
    public void testCase3()
    {
        S.O.P("test case 3 executed");
    }
}
```

Script 10.xml

```
<?xml version = "1.0" - - -  ?>
<!DOCTYPE Suite SYSTEM
<suite name = "Suite" >
  - - -
  - - -
   <class name = "com.abc.TestNGTest
                  Script10" />
    <methods>
     <include name = "testCase1" >
      </include>
     </methods>
    </class> </classes>
   </test>
 </suite>
```

O/P :- Test case 1 executed.

Program

```
class Script11
 { ≡ }
```

script 11.xml

```
<methods>
 <exclude name = "testCase1" ></exclude>
</methods>
```

O/P :- Test case 2 executed
Test case 3 executed

Groups

Program

```
public class Script10
{
    @Test (groups = "inbox")
    public void testcase1()
    {
        S.O.P("test case 1 execute");
    }

    @Test (groups = "compose")
    public void testcase2()
    {
        S.O.P("test case 2 execute");
    }

    @Test (groups = "compose")
    public void testCase3()
    {
        S.O.P("test case 3 execute");
    }

    @Test (groups = "inbox")
    public void testCase4()
    {
        S.O.P("test case 4 execute");
    }

    @Test (groups = "logout")
    public void testCase5()
    {
        S.O.P("test case 5 execute");
    }
}
```

Script12.xml

```
<?xml ... >
...
...
<suite ... >
<test thread ... >
  <groups>
    <run>
      <include name="compose" ></include
    </run>
  </groups>
    <classes>
      <class name="com.abc.TestNGTe
                .Script12"/>
    </classes>
</test>
</suite>
```

Op:-

Test case 2 execute
Test case 3 execute

@BeforeTest and @AfterTest

@BeforeTest

* @BeforeTest and @AfterTest is not related to the test case or test method rather it is related to the test tag in the XML.
* Such methods which are annotated using @BeforeTest will be executed before the test tag of the XML.
* Such methods which are annotated using @AfterTest will be executed after the closure tag of test in the XML.

Program

```
public class Script13
{
    @BeforeTest
    public void beforetest()
    {
        S.O.P("before test executed");
    }

    @AfterTest
    public void aftertest()
    {
        S.O.P("after test executed");
    }
```

```
@BeforeClass
public void .beforeclass()
  {
    S.O.P ("before class executed");
  }

@AfterClass
public void afterclass()
  {
    S.O.P("after class executed");
  }

@BeforeMethod
public void beforeMethod()
  {
    S.O.P("before method executed");
  }

@AfterMethod
public void afterMethod()
  {
```

Script18.xml

```
<suite name="Suite">
  <test thread-count="5" name="Test">
    <classes>
      <class name="abc.TestNGTest.Script13"/>
    </classes>
  </test>
</suite>
```

O/P:-

Note:-

Before Suite
<Suite>
Before Test
<Test>
<classes>
Before Class
<class>
Before Method
Test Case
After Method
</class>
After class
</classes>
</Test>
After Test
</Suite>
After Suite

Various attributes associated with the Test Annotation

1. dependsOnMethods
2. Assert.Fail()
3. invocation count
4. AlwaysRun
5. Enabled

dependsOnMethods

Whenever we have to make one test-case dependent on another test-case then we should be using dependsOnMethods

Assert.Fail()

Whenever we want to intentially failed the test case, then we use Assert.Fail().

invocation Count

Whenever we want to execute a test case multiple no. of times then we can use invocation count.

AlwaysRun

Whenever we want to execute a test case irrespective of whether the dependent test case is executed or not or pass or fail, then we should take a help of AlwaysRun.

Enabled

Whenever we have to hide a test case then we can say Enable is equals to false.

Program

```
public class Script14
{
  @Test
  public void fun1()
  {
     S.O.P("fun 1 is executed");
     Assert.fail();
  }
  @Test(dependsOnMethods = "fun1",
        invocationCount=5, alwaysRun=true)
  public void fun2()
  {
     S.O.P("fun 2 is executed");
  }
  @Test(enabled = false)
  public void fun3()
  {
     S.O.P("fun 3 is executed");
  }
}
```

Parameterization

* In case of parametrization, the data would be stored in the XML file.
* From the XML file, we should fetch the data and we should use it in the test code.
* The parameters can be specified within the test level. If the parameters are specified within the test level, the parameters are available only for the particular test.
* If the parameter has to be made available for all the test, then place the parameters at suite level.

Program

```
public class ParameterDemo
{
    static WebDriver driver;
    @BeforeMethod
    public void beforeMethod()
    {
        driver = new chromeDriver();
        driver.manage().window().maximized();
        driver.manage().timeOuts().
            implicitly wait(20, TimeUnit.SECONDS);
        drivers.get("https://magento.com");
    }
```

```
@AfterMethod
public void afterMethod()
{
    driver.quit();
}

@Parameters ({"username", "password"})
@Test
public void validLogin (String un, String pwd)
{
    driver.findElement(By.linkText("My Account")).
                                        click();
    driver.findElement(By.id("email")).sendKeys
                                        (un);
    driver.findElement(By.id("pass")).sendKeys(pwd);
    driver.findElement(By.id("send2")).click();
    driver.findElement(By.linkText("Log Out")).
                                        click();
}

}
```

Parameter.xml

```
<suite name = "Suite">
    <test -- >
        <parameter name = "username"
                   value = "mzaidurrahman@gmail.
                                        com">
        </parameter>
        <parameter name = "password"
                   value = "Welcome123">
        </parameter>
    <classes>
</suite>
```

Data Providers

[ ][ ] → [ ] Parameter number.
↓
invocation count

```
public class DataProviderDemo
{
    @DataProvider(name = "authentication")
    public Object[][] dataProvider()
    {
        Object[][] obj = new Object[2][2];
        obj[0][0] = "mzaidurrahman@gmail.com";
        obj[0][1] = "Welcome123";
        Obj[1][0] = "vineetanand61@gmail.com";
        obj[1][1] = "Welcome123";
        return obj;
    }

    @Test(dataProvider = "authentication")
    public void validLogin(String un, String pwd)
    {
        WebDriver driver = new ChromeDriver();
        driver.manage().windows().maximize();
        driver.manage().timeouts().implicitlyWai
                                    (20, TimeUnit.SECONDS);

        driver.findElement(By.linkText("My Account")
                                    .click();
        — —
        — —
        — —.
        — —
    }
}
```

Cross Browser Testing

Whenever we want to test the software under test in different browsers or whenever the s/w has to be tested in multiple browsers parallely then we use the concept of Cross Browser Testing.

Program

```
public class CrossBrowserDemo
{
    static WebDriver driver;
    @Parameters ("browser")
    @BeforeMethod
    public void beforeMethod (String browser)
    {
        if (browser.equalsIgnoreCase ("firefox"))
        {
            driver = new FirefoxDriver();
        }
        else
        {
            driver = new ChromeDriver();
            driver.manage().window().
                maximize();
        }
}
```

```
@Test
public void validLogin()
{
    driver.manage().timeouts().--
    driver.get("https://magento.com");
    driver.findElement(By.linkText("MyAccount")
                       .click();
    --
    --
    --
    --
}
}
```

CrossBrowserDemo.xml

```
<suite -- >
  <test thread-count="5" name="firefox"
        parallel="tests">
    <parameter name="browser"
               value="firefox"></parameter>
    <classes>
      <class name="--"/>
    </classes>
  </test>
  <test thread-count="5" name="chrome"
        parallel="tests">
    <parameter name="browser"
               value="chrome">
               </parameter>
    --
    --
  </test>
</suite>
```

09-04-19

Maven

It is a project management tool which will download the jar files automatically into the project.

Steps involved for downloading Apache maven :-

1. Open any browser, search for Apache maven
2. click on first link (https://maven.apache.org)
3. click on download
4. Scroll down to the file section & click on Apache-maven-3.6.0-bin.zip
5. Place the downloaded zip in Selenium component folder & extract.

Right click on this PC icon, click on properties, click on advance system settings, click on Environment Variables.

Under system variables, click on New. Type the variable name as M2_HOME & for the variable value, paste the path of maven home directory.

click on OK.

Again click on New, for the variable name, give the values as MAVEN_HOME for the variable value paste the same path again, click on OK.

Under System Variable, click on a variable path, click on New & paste the path of apache-maven build. click on OK for four times.

Maven Project consist of three phases:-

① Creation of the project
② Adding the dependencies
③ Execution of the project

Creation of the Project :-

1. Go to eclipse, click on File, click on New
2. click on Project & search for maven
3. click on Maven Project
4. Check the create simple project checkbox
5. click on Next.
6. In the group id text box, type the package name com.abc.Magento
7. In the artifact id text box, type the name as MagentoTest
8. Click on Finish
9. copy the page objects, created in the page object model framework (i.e. Welcome.java, Login.java, ~~Logout.java~~ Home.java) & paste it in src/main/java folder of Maven project.
10. Copy the main file of the PageObjectModel MagentoTest.java & paste it in src/test/java folder of Maven project.

Adding dependencies to the Project

① Open the Browser, search for mvn repository
② click on first link (https://mvnrepository.com)
③ In the search box, type selenium & click on Search
④ click on first link which says Selenium java.
⑤ click on 3.141.59 version.
⑥ From the maven text box, copy the code, go to eclipse, in the maven project, we will have a file by the name POM.xml,
⑦ open it, In the POM.xml, before the closure of project, create a new tag by the name dependencies & paste the copied code within the tag.
⑧ Go back to mvn repository, search for TestNG
⑨ click on first link
⑩ click on 6.14.3 version
⑪ Copy the code present in the maven text box & paste it, below the closure of dependency tag of selenium in POM.xml file of maven project.

Execution of the Project

① Select the maven project
② Right click on it, Select Maven & click on Update project.
③ To run the project, select project, right click, select run as maven test.

```
public class MagenloTest {

    @Test
    public void validLogin()
    {
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.manage().timeOuts().implicitlyWait
                    (20, TimeUnit.SECONDS);
        driver.get("https://magenlo.com");
        Welcome w = new Welcome(driver);
        w.clickOnMyaccount();
        Login l = new Login(driver);
        l.TypeEmail();
        l.TypePassword();
        l.clickOnLogin();
        Main m = new Main(driver);
        m.clickOnLogout();
    }

}
```

10th April '19

Version Control Tool (or) Source Code Management Tool (or) Revision Control Tool

Version Control Tools are used to manage the code develop by the developers in a single repository.

It can also be used to be monitor the revisions done to the code.

The Version Control Tool, we are using is Git.

Downloading Git

1. Open any browser, search for Git
2. click on first link (https://git-scm.com/)
3. click on download 2.21.0 for windows.
4. Keep the downloaded file in Selenium component folder.
5. Double click on Git exe file.
6. Follow the installation by clicking on Next.
7. Once the installation is complete, Go to Github.com create one account

Adding Project to the Repository

1. After the completion of account creating in Github, login to Github account
2. Click on start a project
3. Give the repository name as MagentoTesting
4. Click on create repository.
5. Go to Eclipse, in the Quick Access Search Box, Type Git

(6) Click on Git Repositories
(7) Click on Clone a Git repository
(8) In the Source Git Repository window, copy the repository url from the github website ending with .git & paste it in URI of source Git Repository window.
(9) Click on Next
(10) In the branch selection window, click on Next.
(11) In the local Destination Window, click on Finish.
(12) Select the project that you want to add to the repository, right click on it, Select team & click on Share Project
(13) In the Configure Git Repository Window, click on the Repository dropdown & Select the repository displayed.
(14) click on Finish
(15) Again Right click on the project which we have configured with the repository. Select Team & Click on Commit.
(16) Move the files from Unstaged changed to Staged changed dropdown.
(17) Give the Commit message as first-commit.
(18) click on Commit & Push
(19) click on Next