

Selenium:

Disadvantages of manual testing:

1. More human efforts are required.
2. Test cycle duration is increased.
3. Compatibility testing (Executing same test cases on different browser) is difficult.
4. Regression testing is also time consuming as we run same test cases after the build update to check effect of newly added module on old module.

What is Automation testing?

Automation testing is defined as testing the application features by using automation tool and executing test script on that tool.

A testing in which after the triggering (execution of code) the content over the application acts on the basis of the code written by automation tester.

The different types of automation tools are Selenium, ATP, Selendroid, Appium etc. Automation testing tool is used to perform the testing but as a tester we have to provide some commands to this tool and that commands are called as "Scripting."

Advantages of Automation testing:

1. Less human efforts are required.
2. Test cycle duration decreases.
3. Compatibility testing becomes easier. We can perform compatibility testing by changing 2-3 lines of our script/code.
4. Reusability of code is allowed.
5. Regression testing is also become easier as we can use same test script.
6. Project cost reduced due to automation.
7. It is reliable and efficient.

Advantages of Selenium:

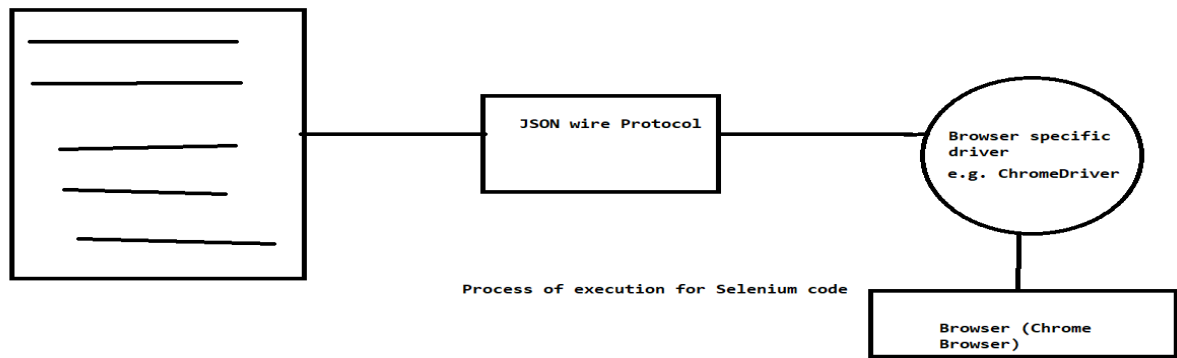
1. It is open source automation tool.
2. It supports multiples programming languages such as java, python, C sharp etc.
3. Cross browser testing is possible
4. Cross platform testing is also possible

Disadvantages of Selenium:

1. We can't automate desktop based application (ex. Team viewer, MS-office)
2. We can't automate standalone applications (ex. Calculator).
3. We can't automate captcha code using selenium.
4. We can't read barcode using selenium tool.
5. It doesn't support file uploading.
6. Ad-hoc testing (we know application but can't have test data) can't be performed.

Different java concepts used in selenium:

- | | |
|------------------|------------------------|
| 1. Inheritance | 7. Arrays |
| 2. Polymorphism | 8. Collections |
| 3. Interface | 9. Loops |
| 4. Up-casting | 10. Control statements |
| 5. Abstraction | 11. String class |
| 6. Encapsulation | |



JSON wire Protocol: It is used to make sure the components that we have for execution (driver, browser and the code) are compatible for the execution of code.

Browser specific driver: It is used to provide commands to the browser which can act accordingly. The version for browser's driver and browser should be compatible in-order to perform the action.

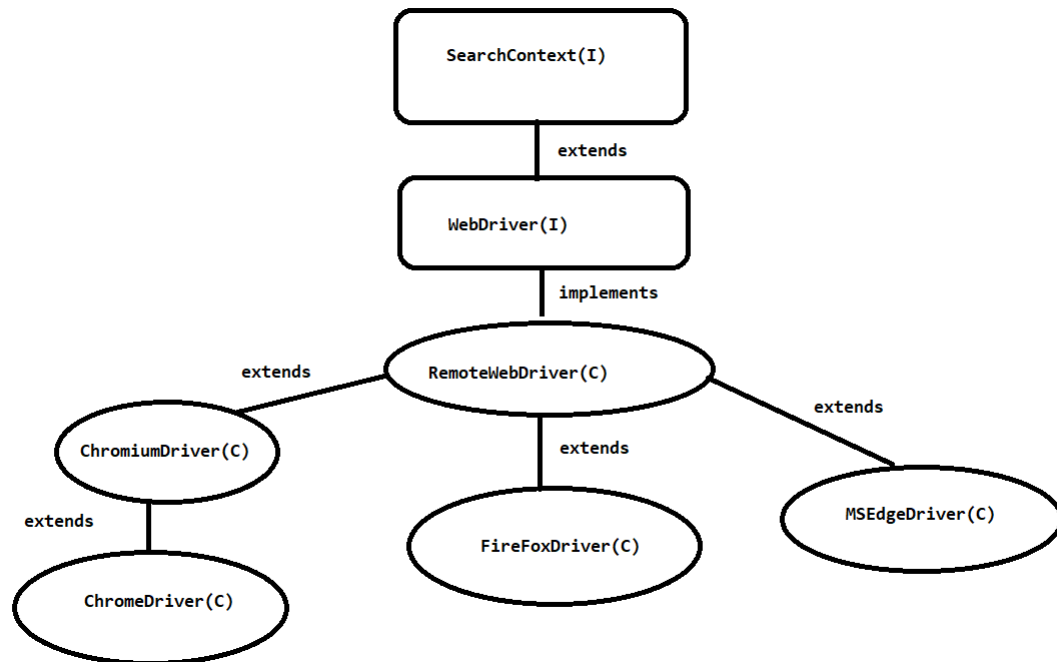
Architecture of Selenium

1. Search Context: It is a super most interface in selenium. It consists of all abstract methods and that methods are inherited to the Webdriver interface.

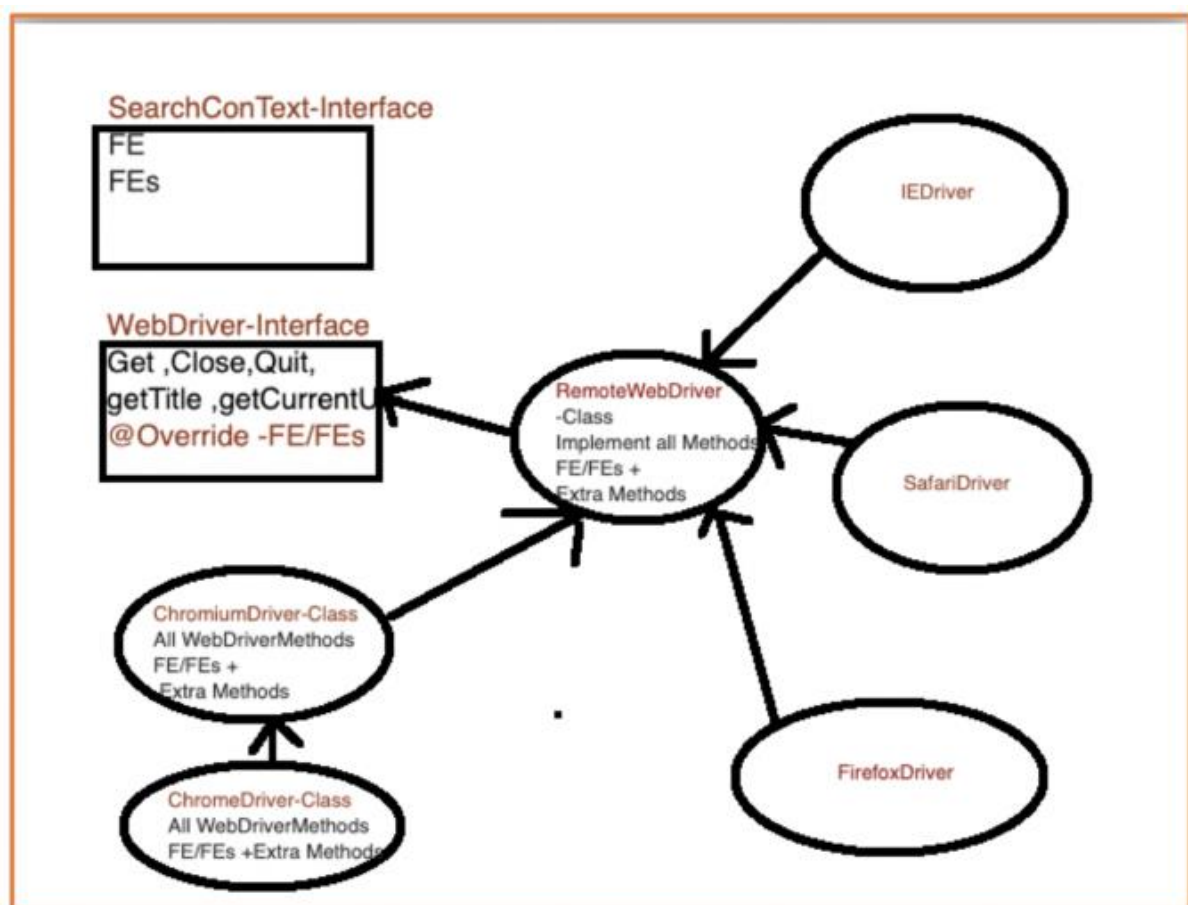
2. Webdriver: It is an interface present in selenium which consists of two types of abstract methods that is abstract methods of search context and his own abstract methods.

3. Selenium Remote Webdriver: It a class which implements all the abstract methods of both the interfaces that is search context and Webdriver. This implementation class is extended to the different browsers such as chrome. Firefox, internet explorer etc.

4. Browser driver: For compatibility testing we need to use runtime polymorphism to perform up casting in selenium. For example, to open a browser using script, we create an object of chrome driver with reference of Webdriver interface. `WebDriver driver = new ChromeDriver ()`



Architecture of Selenium



❖ **Webdriver and its methods:** Webdriver is an interface which is used to perform different actions on browser such as open browser, close browser, navigation on browser etc. to perform these actions Webdriver have inbuilt methods and that methods are called as Webdriver Methods.

- 1) Get method:** Get method is used to open an application or to provide the URL to browser. It doesn't return any value. It will just open the browser and connect to given URL. If we provide wrong URL in get method then we will get Webdriver exception.
- 2) Close Method:** This method is used to close current tab of browser. Immediate opening and closing of browser does not provide proper simulation then we have to add wait timer. So that after opening of browser it will wait for some time and then it will close it. This wait is given in milliseconds. To provide this wait time we have to use Thread class of java. In thread class sleep method is present which provides the wait time. But if we enter wait time then thread class generates the exception then to handle this we have to use throws InterruptedException before closing the method.
- 3) Quit Method:** This method is used to close all tab of the browser. This is an alternative to close method. If we interrupt current running script manually then we get exception saying that unreachable browser exception. So avoid manual interruption while script is running.
- 4) Get title method:** Each website has its own title. This method is used to get the title of particular website. This method will returns string type of information. We can display this string in our console. We can also compare the expected title with title received by using get title. For this purpose we have to use string equals or equal ignore case function.
- 5) Get current URL:** This function is used to get URL of current tab of browser. The return type of this method is String information
- 6) Maximize:** This method is used to maximize the browser. We can't minimize the browser using selenium tool but we can change the size of browser using set function.

7) Navigate: This method is used to perform various operations on browser such as move forward, move backward and refresh the webpage. This method can be used as alternate method to get method

8) Set Size: This function is used to set the size of browser as per given dimensions. We have to pass dimension argument to set size function. But before that we have to create the object of dimension class of selenium tool and we have to pass expected height and width of browser as a parameter to that dimension class.

9) Set Position: This function is used to set the position of browser as per given parameter. We have to pass point argument to set position function. But before that we have to create the object of point class of selenium tool and we have to pass expected x and y coordinates of new position of browser as a parameter to that point class.

10) Get Size and Position: this method is used to get current size and position of browser.

❖ **Web elements methods:** It is an interface used to perform action on element present on browser. If we want to perform multiple actions on same element then we can find that element at once and store it in reference variable having data type as web element. It will remove every time finding of same element. We just use that reference variable and perform the desired action on it. Some important web element methods are given below.

public interface WebElement
extends **SearchContext, TakesScreenshot**

✚ **Send keys ():** This method is used to enter the value in the text field present on the webpage. We have to pass string argument to this method.

- ✚ **Click ()**: This method is used to click on button links available on webpage. We also use click option to select the radio button and check box. We are not passing any argument to this method.
- ✚ **Clear ()**: This method is used to clear/remove the value in the text field present on the webpage. We are not passing any argument to this method.
- ✚ **Is enabled ()**: This method is used to check whether the element is enabled for operation or not. We are not passing any argument to this method. This method will return Boolean result in terms of true or false. If element is enable for action then this method will return true and if element is disable for action then this method will return false. Following java code is an example of this method.
- ✚ **Is Selected ()**: This method is used to check whether the element such as radio button or check box is selected or not. We are not passing any argument to this method. This method will return Boolean result in terms of true or false. If element is selected previously then this method will return true and if element is not selected then this method will return false
- ✚ **Is Displayed ()**: This method is used to check whether the element is displayed on the webpage or not. We are not passing any argument to this method. This method will return Boolean result in terms of true or false. If element is displayed then this method will return true and if element is not displayed/present on webpage then this method will give exception. To handle this exception we have to use try and catch block to handle risky code.
- ✚ **Get Text ()**: This method is used to get text present on the web element on the webpage. We are not passing any argument to this method. This method will return string information present on web element.

public class RemoteWebDriver

extends java.lang.Object

implements [WebDriver](#), [JavascriptExecutor](#), [HasInputDevices](#),
[HasCapabilities](#), [HasVirtualAuthenticator](#), [Interactive](#), [PrintsPage](#),
[TakesScreenshot](#)

public class ChromiumDriver

extends [RemoteWebDriver](#)

1) a) **public class ChromeDriver**
extends [ChromiumDriver](#)

b) **public class EdgeDriver**
extends [ChromiumDriver](#)

2) **public class FirefoxDriver**
extends [RemoteWebDriver](#)

3) **public class InternetExplorerDriver**
extends [RemoteWebDriver](#)

4) **public class SafariDriver**
extends [RemoteWebDriver](#)

Basic operations:

```
public static void main(String[] args) throws InterruptedException {  
    System.setProperty("webdriver.chrome.driver",  
"E:\\\\desktop\\Katraj\\Oct  
Batch\\Selenium\\Chromedrivers\\chromedriver.exe");
```

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

to go to a particular url

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

```
Thread.sleep(2000);
```

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

to perform browser's back button operation

```
driver.navigate().back();
```

```
Thread.sleep(2000);
```

to perform browser's forward button operation

```
driver.navigate().forward();
```

to perform refresh operation

```
Thread.sleep(2000);
```

```
driver.navigate().refresh();
```

To perform the maximize operation

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

```
Thread.sleep(5000);
```

to close the browser

```
driver.close();
```

WebElement: The element which is available over the webpage, numerous number of web element combines together to form a webpage.

Web Element like Dropdown, button, text, input field.

To perform the operation over the webpage we need to identify the web element and for that purpose we require their locator for every element.

In Selenium WebElement is an interface which contains two methods:

a. `findElement();`

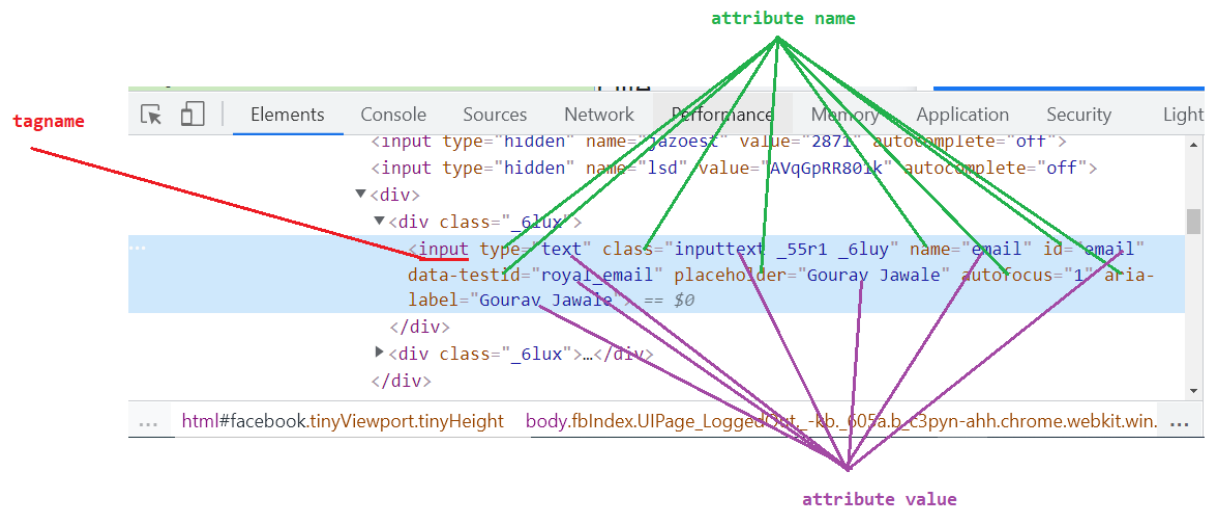
b. `findElements();`

`findElement()` Locator in selenium:

is used to find the first WebElement that matches the current context.

Return type of `findElement()` method is **WebElement** (it is an Interface).

Locators are used to identify the web elements present on web page or browser



❖ **Locators:** Locators are mainly used to find the elements present on the webpage. To find the element present on webpage we have to use find element method of WebElement interface. This method uses by class. This by class consists of various static methods. All the static methods present in the by class are called as Locator types. All the locators takes String type input and there return type is 'by' where as return type of find element method is web element.

Types of Locator:

- A. X Path
- B. Id
- C. Name
- D. Class name
- E. Tag name
- F. CSS selector
- G. Link test
- H. Partial link test

❖ **Locator using ID:** This method is used to find element when element has a attribute with ID. We can't use locator using ID in two cases. When ID is not present and when ID is duplicates. That means two ID have same values then selenium perform action for first ID only. This can be located by using method `id()` of `By` class.

example:

```
WebElement username = driver.findElement(By.id("email"));
```

❖ **Locator using Class Name:** Class name is used when ID is not present or ID has duplicate values. We can't use locator using class name in two cases. When class name is not present or when class name has duplicate values.

```
WebElement loginbutton = driver.findElement  
    (By.className("button"));  
  
loginbutton.click();
```

- ❖ **Locator using Name:** When class name is not present or when class name has duplicate values then we are going to use locator by name. We can't use locator by name when name is not present or when name has duplicate values.

example:

```
WebElement username = driver.findElement
                        (By.name("txtUsername"));
username.sendKeys("Admin");

WebElement password = driver.findElement
                        (By.name("txtPassword"));
password.sendKeys("admin123");
```

- ❖ **Locator using Link Text:** If tagname, ID, Class name, Name is not present or having some duplicate values in html code then we should use Link text or partial link text method to access the web element. It is mainly used to find the element having link and text present on that link. We can't use these methods for normal text codes. In link text we identify the element by using entire text present on the link. This locator is used when we have a link available over the page.

```
WebElement forgotlink = driver.findElement
                        (By.LinkText("Forgot your password?"));
forgotlink.click();
```

- ❖ **Locator using Partial Link Text:** If tagname, ID, Class name, Name is not present or having some duplicate values in html code then we should use Link text or partial link text method to access the web element. It is mainly used to find the element having link and text present on that link. We can't use these methods for normal text codes. In partial link text we identify the element by using some part of text present on the link.

```
WebElement forgotpasswordlink = driver.findElement
                        (By.partialLinkText("Forgot you"));
forgotpasswordlink.click();
```

- ❖ **Locator using Tagname:** We can use tagname to find the locator using by class. But if our code / webpage have multiple elements with same tagname then selenium perform action on only first element. This is disadvantage of tagname. To avoid we have to go to next locator that is ID.

```
WebElement tagnameexample = driver.findElement  
                                (By.tagName("a"));  
tagnameexample.click();
```

- ❖ **Locator using X Path:** X path is again divided into following sub types

- a. X path by Attributes
- b. X path by Text
- c. X path by Contains
- d. X path by Index
- e. Absolute X path
- f. Relative X path

a) X path by attributes: To find web element by using this method we have to follow following steps.

- 1) Open browser using selenium tool.
- 2) Enter valid URL.
- 3) Give wait time so that we can easily differentiate the changes happening.
- 4) Once browser is open then move cursor on one particular web element such as username field.
- 5) Right click on it and select inspect option. Then html code of that page will open. To select HTML code of particular web element we have to move cursor on that element then color of code become blue.
- 6) Press control + F to open find tab. Then write your x path expression in that tab. To write x path expression use following

Syntax: //tagname[@attribute_name = 'attributevalue']

7) Once you complete your x path expression then color of code changes from blue to yellow and it shows 1 of 1. Hence we can conclude that we successfully find the element and make its expression.

8) Then go to your code and after get methods write following method to find element using code.

```
driver.findElement (By.xpath ("Provide your x path here")).sendKeys ("Your Username")
```

9) Repeat step 5 to step 8 for each web element. To find a proper tag name use following hierarchy. Follow the flow id / name/ class/ placeholder.

10) To click any button, use click function after x path expression

This method provides flexibility to use any attribute for locating the element. Through this method we can also see how many nodes are getting match.

Example:

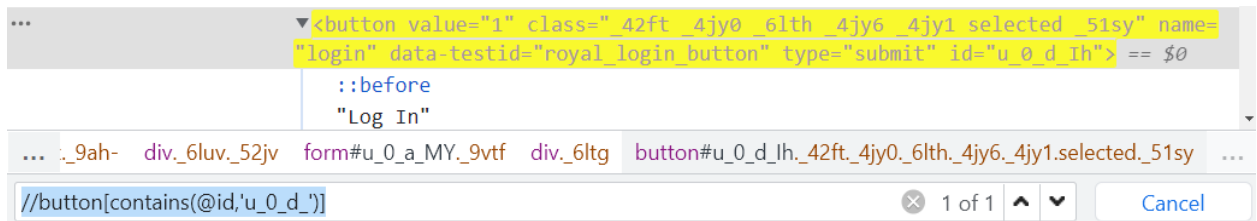
```
WebElement username = driver.findElement  
    (By.xpath("//input[@aria-label='Email address or  
    phone number']"));  
  
username.sendKeys("abc@abc.com");  
  
driver.findElement(By.xpath("//input[@id='pass']"))  
    .sendKeys("password");
```

Customization of Xpath:

- ❖ **X path by contains:** This method will check whether an attribute value matches with some of the text.
when the space created by non breakable method then we can't use X path by text. Hence we are going to use X path by contains. It has two types.

1. By using attributes: //tagname [contains(@Attribute name, 'Attribute value')]
2. By using Text: //tagname [contains(text(), 'Text Value')] All remaining procedure will be same as per above examples/types.

Example:



```
driver.findElement(By.xpath("//button[contains(@id,'u_0_d_1h')])).click();
```

❖ **X path by text:** This method is used to form the xpath when we have angular braces(>word<) on either side of the word

X path by test locator is used when developer create an element using tagname and text only. When we can't find attribute then we have to go with this type. We have to use following formula to create X path expression. Rest of the process is same as per the previous locator.

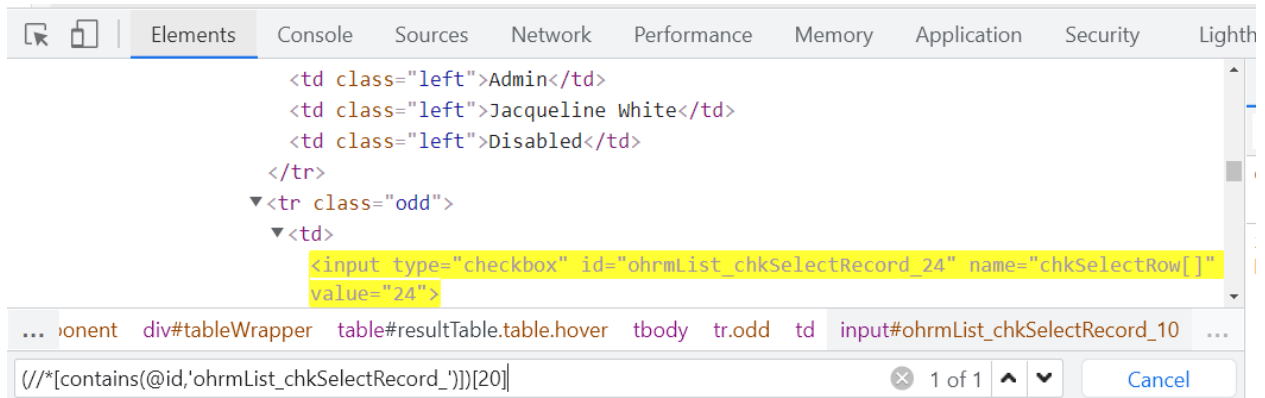
//tag name [text () = 'text value']



```
driver.findElement(By.xpath("//a[text()='Forgotten password?']")).click();
```

- ❖ **X path by Index:** If we have multiple nodes match then we can have indexing of xpath by covering the whole xpath with () and in square bracket we can provide the index number.

While finding element we get multiple x paths. If we search for text type then you might get multiple paths such as 1 of 1 or 1 of 3 etc. in such case we have to use X path index

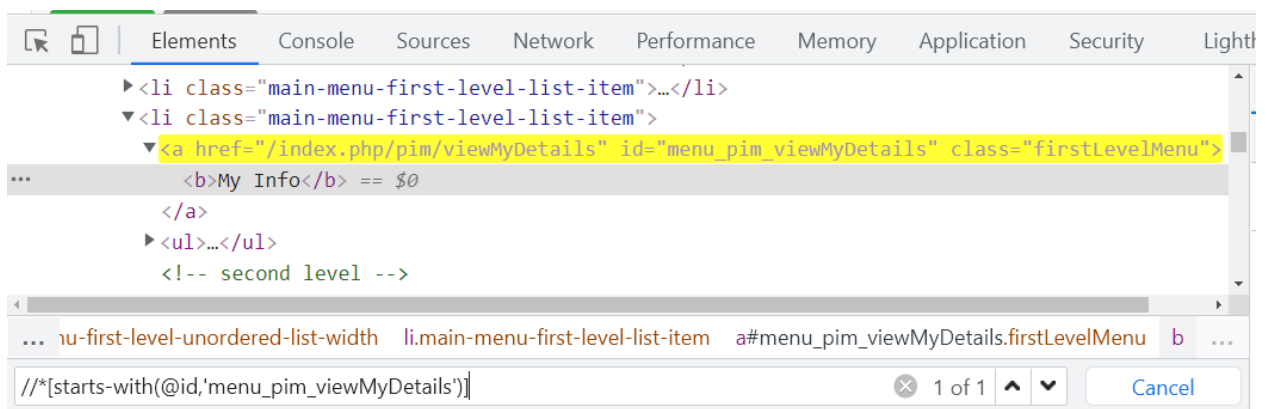


```
driver.findElement(By.xpath("(//*[contains(@id,'ohrmList_chkSelectRecord_')])[20]").click();
```

Here the highlighted xpath is getting change with respect to time hence it is called as Dynamic xpath.

- ❖ **x-path using starts-with:**

It matches with the node if the attribute value starts with the given attribute value.



Example 2:



- ❖ **Relative X path:** To find the X path by using absolute method we need to find HTML tree diagram first. In that diagram we have to move forward from parent to any child class. We need to use double forward slash (//) in between parent and any child. We are using same example an html tree diagram which is used in absolute X path. The xpath which we have formed by traversing (parent to child node) and in that we have used // then it is called as Relative xpath.

Disadvantages:

- 1) Identifying the element using HTML tree is difficult.
- 2) It's a time lengthy and time consuming process



- ❖ **Absolute X path:** To find the X path by using absolute method we need to find HTML tree diagram first. In that diagram we have to move forward from parent to next immediate child. We need to use single forward slash (/) in between parent and its immediate child. We have to follow this approach until our expected element not found.

Disadvantages:

- 1) Identifying the element using HTML tree is difficult.
- 2) It's a time lengthy and time consuming process

The xpath which we have written by using '/' then it is known as absolute xpath.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>OrangeHRM</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <style type="text/css">...</style>
    <link rel="shortcut icon" href="/webres_61c2fbeda25b10.20973602/themes/default/images/
    o">
  ... html body div#wrapper div#content div#divLogin div#divLoginImageContainer div#divLoginFo
  /html/head/style 1 of 1
```

CSS and Xpath Axes

Methods name are:

Ancestor--- ancestor

---ancestor-or-self

Descendent--- descendant

Descendant-or-self

Sibling preceding-sibling

Following-sibling

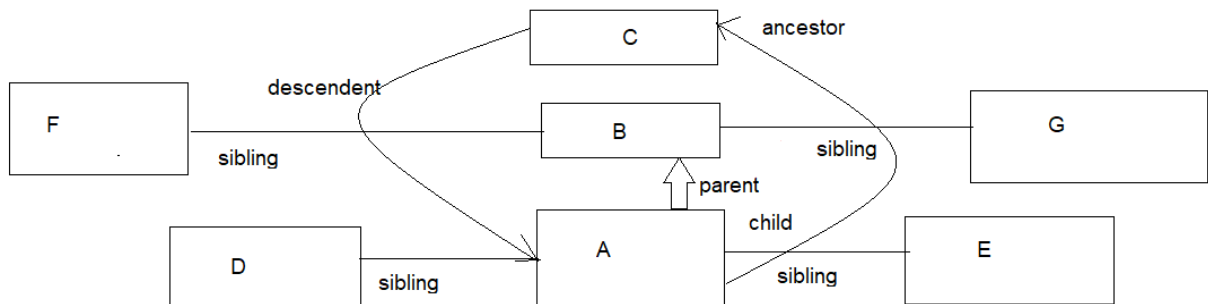
Parent parent

Child child

Syntax:

/axesMethodName::tagname

url used <https://www.hyrtutorials.com/p/add-padding-to-containers.html>



1. following-sibling :

email

Password

Repeat Password

Elements Console Sources Network Performance Memory Application Security

```
<br>
<label>Last Name</label>
<br>
<input maxlength="15" name="name" type="text">
<br>
...
<label>Email</label> == $0
<br>
<input required type="text">
<br>
▶<div>...</div>
```

... outer div.post.hentry article div#post-body-299858861183690484.post-body.entry-content div form div.c

//label[text()='Email']/following-sibling::input[1]

1 of 1

2. parent method:

```
<div class="container">
  <h1>Register</h1>
  "Please fill in this form to create an account. or if you already have
  <a href="https://www.blogger.com/blogger.g?blogID=2026392825261642617&
  in into account</a>
  <br>
  <hr>
  <label>First Name </label>
  ...
</div>
```

... outer div.post.hentry article div#post-body-299858861183690484.post-body.entry-content div form div.conta

//label[text()='Email']/parent::div

or can be

//label[text()='Email']/following-sibling::input[1]/parent::div

3. child:

//div[@class='container']/child::input[@type='text']

```
<label>First Name </label>
<br>
<input maxlength="10" name="name" type="text">
<br>
<label>Last Name</label>
<br>
<input maxlength="15" name="name" type="text">
<br>
<label>Email</label> == $0
<hr>
...
outer div.post.hentry article div#post-body-299858861183690484.post-body.entry-content div form div.conta
```

//div[@class='container']/child::input[@type='text']

4. preceding-sibling:

`//*[text()='Alice Duval']/preceding-sibling::td/child::input`

<input type="checkbox"/>	Aaliyah Haq	ESS	Aaliyah Haq
<input type="checkbox"/>	Aatmaram	ESS	Alice Duval
<input type="checkbox"/>	Admin	Admin	Anton Heathcote
<input type="checkbox"/>	AlexxGun	ESS	Fiona Grace
<input type="checkbox"/>	Alice Duval	ESS	Alice Duval
<input type="checkbox"/>	Anthony Nolan	ESS	Anthony Nolan
<input type="checkbox"/>	Aravind	ESS	Dominic Chase
<input type="checkbox"/>	Cassidy Hope	ESS	Cassidy Hope


```
<input type="checkbox" id="ohrmList_chkSelectRecord_37" name="chkSelectRow[]" value="37">
</td>
<td class="left">...</td>
<td class="left">ESS</td>
<td class="left">Alice Duval</td>
<td class="left">Enabled</td>
</tr>
<tr class="odd">...</tr>
<tr class="even">...</tr>
```

... Header > div.inner > form#frmList_ohrmListComponent > div#tableWrapper > table#resultTable.table.hover > tbody > tr.odd > td.left

`//*[text()='Alice Duval']/preceding-sibling::td/child::input` 1 of 2 Cancel

5. Descendant: To locate the grand children:

`//div[@class='main section']/descendant::article`

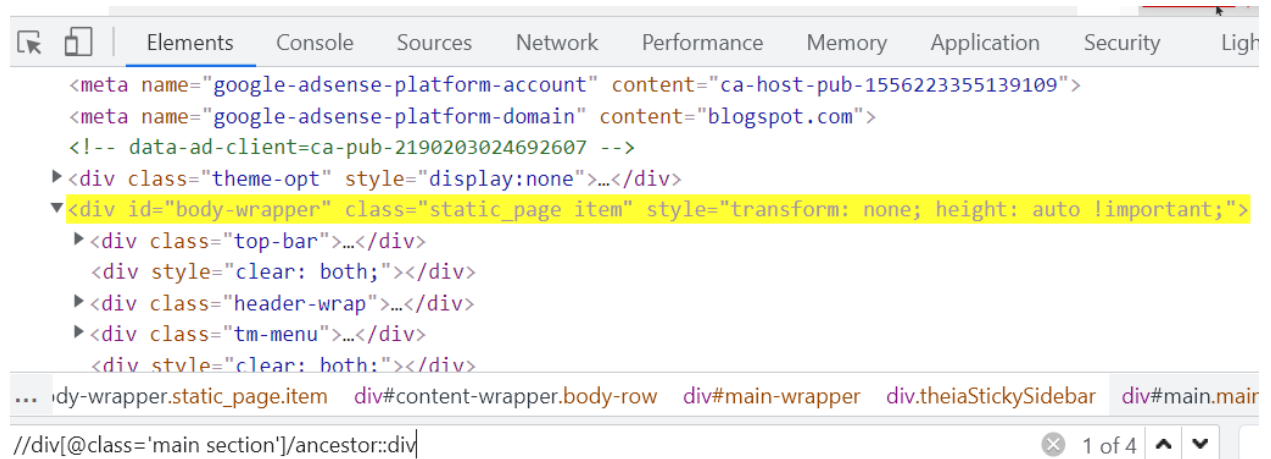
```
<div id="ads-blog">...</div>
<div class="main section" id="main"> == $0
  <div class="widget Blog" data-version="1" id="Blog1">
    <div class="blog-posts hfeed">
      <div class="post-outer">
        <div class="post hentry">
          <div class="post-header">...</div>
          <article>...</article>
          <div class="post-footer">...</div>
        </div>
      </div>
    </div>
  </div>
```

... > body-wrapper.static_page.item > div#content-wrapper.body-row > div#main-wrapper > div.tl

`//div[@class='main section']/descendant::article`

6. Ancestor:

//div[@class='main section']/ancestor::div



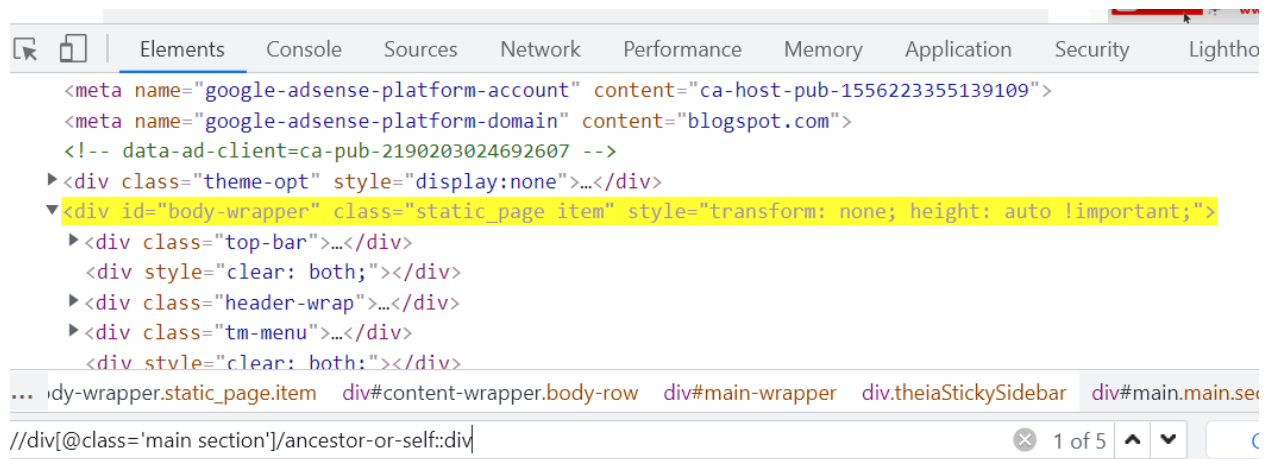
```
<meta name="google-adsense-platform-account" content="ca-host-pub-1556223355139109">
<meta name="google-adsense-platform-domain" content="blogspot.com">
<!-- data-ad-client=ca-pub-2190203024692607 -->
<div class="theme-opt" style="display:none">...</div>
<div id="body-wrapper" class="static_page item" style="transform: none; height: auto !important;">
  <div class="top-bar">...</div>
  <div style="clear: both;"></div>
  <div class="header-wrap">...</div>
  <div class="tm-menu">...</div>
  <div style="clear: both;"></div>
  ...
  <div class="main-section">...</div>
</div>
```

... body-wrapper.static_page.item div#content-wrapper.body-row div#main-wrapper div.theiaStickySidebar div#main.main

//div[@class='main section']/ancestor::div 1 of 4

7. ancestor-or-self method:

//div[@class='main section']/ancestor-or-self::div



```
<meta name="google-adsense-platform-account" content="ca-host-pub-1556223355139109">
<meta name="google-adsense-platform-domain" content="blogspot.com">
<!-- data-ad-client=ca-pub-2190203024692607 -->
<div class="theme-opt" style="display:none">...</div>
<div id="body-wrapper" class="static_page item" style="transform: none; height: auto !important;">
  <div class="top-bar">...</div>
  <div style="clear: both;"></div>
  <div class="header-wrap">...</div>
  <div class="tm-menu">...</div>
  <div style="clear: both;"></div>
  ...
  <div class="main-section">...</div>
</div>
```

... body-wrapper.static_page.item div#content-wrapper.body-row div#main-wrapper div.theiaStickySidebar div#main.main.se

//div[@class='main section']/ancestor-or-self::div 1 of 5

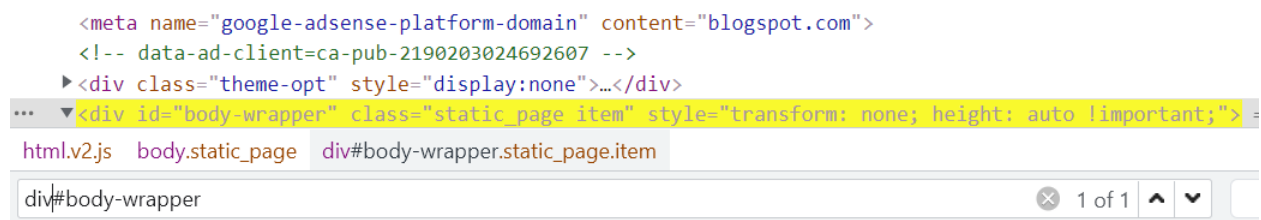
Like wise we have descendent –or-self , following and preceding methods.

CSS Selector:

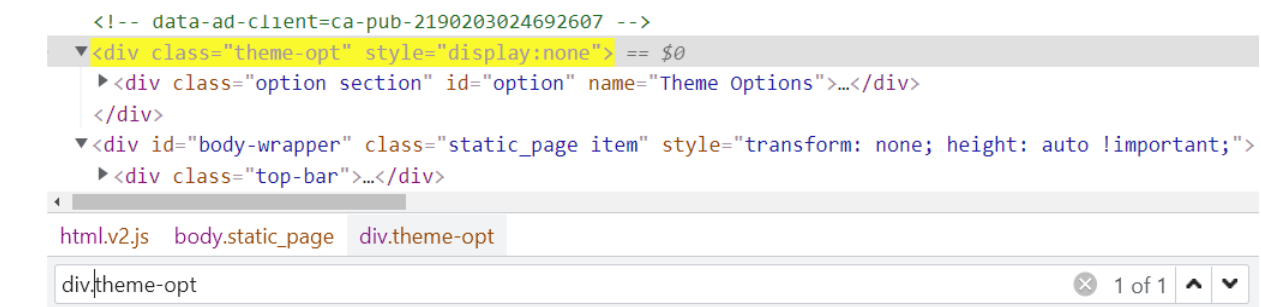
id

Xpath: `//tagname[@id='value']`

CSS: `tagname#value`



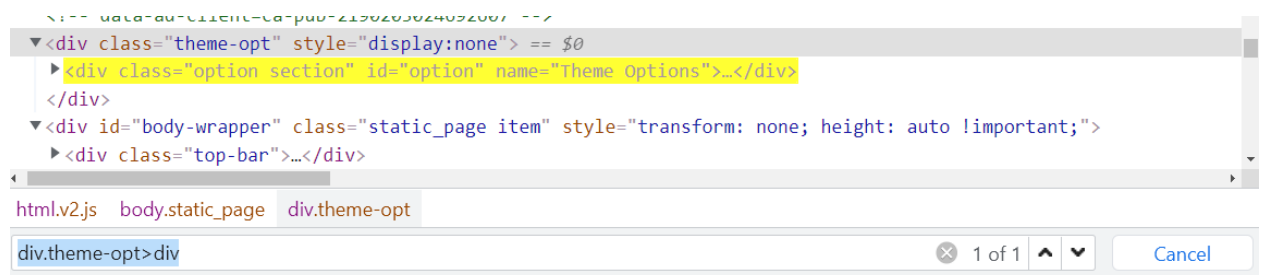
Class:



Xpath: `//div[@class='theme-opt']`

CSS: `div.theme-opt`

Parent to child traversing:



Xpath: `//div[@class='theme-opt']/div`
CSS: `div.theme-opt>div`

Some valid CSS:

```
input[type=checkbox]
```

or

```
[type=checkbox]
```

Starts with in CSS

```
input[type^=ch]
```

ends with in CSS

```
input[type$=box]
```

Difference between CSS and Xpath:

1. CSS expression is compact as compared to xpath hence it doesn't take more time to evaluate.
2. In CSS there is direct location of that element irrespective to the xpath where we have so many relationship that come in the form of parent to child traversing. Hence CSS is much faster than Xpath.

Text capture using selenium:

We can get the text of a webElement by using `getText()`.

```
WebElement message = driver.findElement
    (By.xpath("//*[@id='spanMessage']"));

String actualmessage = message.getText();

System.out.println(actualmessage); //Invalid Credentials
```

FindElements method:

method is used to find all elements within the current context. It returns all the elements that match .

To locate multiple webelements over the webpage we have to use `findElements`.

Return type of this method is `List<WebElement>` i.e list of webelement.

Example:

```
List<WebElement> checkboxes =
driver.findElements(By.xpath("//*[contains(@id,'ohrmList_c
hkSelectRecord_')]"));
// to get number of checkboxes
int numberOfcheckbox = checkboxes.size();

System.out.println(numberofcheckbox);

for(WebElement checkbox :checkboxes)
    checkbox.click();}
```

Handling dropdown using findElements:

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

    System.setProperty("webdriver.chrome.driver",
        "E:\\desktop\\Katraj\\Oct
        Batch\\Selenium\\Chromedrivers\\chromedriver.exe");

    WebDriver driver = new ChromeDriver();

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

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

    List<WebElement> day = driver.findElements
        (By.xpath("//*[@id='day']//option"));

    for(WebElement actualdate:day)
    {
        String datevalues = actualdate.getText();
        System.out.println(datevalues);

        if(datevalues.equals("16"))
        {
            actualdate.click();
        }
    }

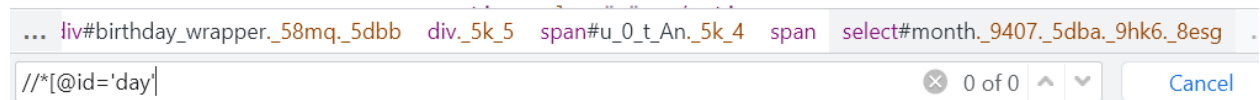
    List<WebElement> month = driver.findElements
        (By.xpath("//*[@id='month']//option"));

    for(WebElement monthvalue:month)
    {
        String actualmonth = monthvalue.getText();
        System.out.println(actualmonth);
        if(actualmonth.equals("Sep"))
        {
            monthvalue.click();
        }
    }
}
```

List box or Dropdown handling :

1. Using Select class: We can handle a dropdown using select class only for those elements which has tagname as 'Select' tag.

```
<select aria-label="Day" name="birthday_day" id="day" title="Day"
class="_9407_5dba_9hk6_8esg" aria-describedby="js_hi">
  <option value="1">1</option>
  <option value="2">2</option>
```



We have following ways to use select class:

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

```
System.setProperty("webdriver.chrome.driver",
"E:\\desktop\\Katraj\\Oct
Batch\\Selenium\\Chromedrivers\\chromedriver.exe");
```

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

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

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

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

Create the object of select Class and pass web element as argument inside the constructor of select class

```
Select daysel = new Select(day);
```

```
daysel.selectByVisibleText("6");
```

```
Thread.sleep(2000);
```

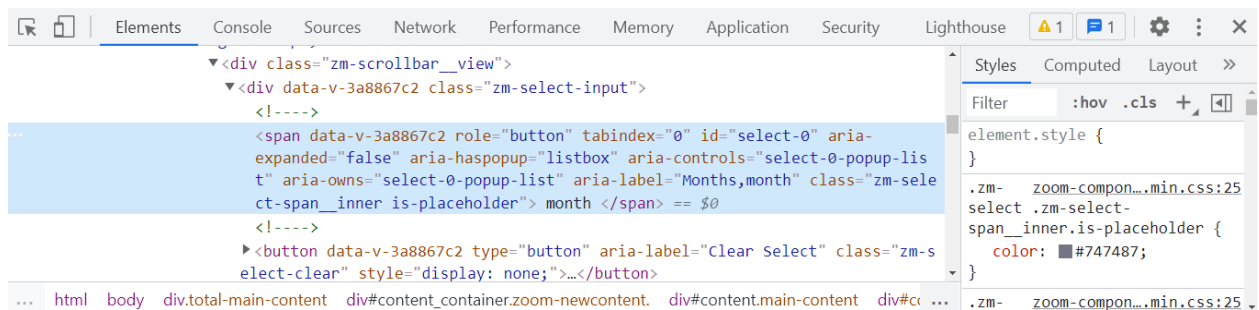
```
daysel.selectByValue("18");
```

```
Thread.sleep(2000);
```

```
daysel.selectByIndex(28);
```

2. Using Bootstrap: Handling dropdown which doesn't have select tag, this kind of dropdown is also known as Bootstrap dropdown.

Example:



```
driver.get("https://zoom.us/signup");
```

```
driver.findElement(By.xpath("//*[@id='select-0']")).click();
```

```
driver.findElement(By.xpath("//*[@id='select-item-select-0-3']")).click();
```

Screenshot using selenium:

- 1) To take the screenshot we have to call `getScreenshotAs` method. And this method present inside the `TakesScreenshot` interface.
- 2) So we have to Type cast the driver object to the `TakesScreenshot` interface. i.e chrome driver class will upacasted in `TakeScreenshot` interface.
- 3) Then use `getScreenshotAs` method and pass argument as (*output Type file*).
- 4) Store the result of above operation in one reference (source) having data type as file.
- 5) This variable consists of default path at which our screen shot is present (*Source*).
- 6) To store the screenshot we have to create an object of `File` class and then provide the *path* as an argument .hence reference variable of `File` class will be our *destination* .
- 7) So to copy the file from this (*Source*) to another location (*destination*), we have to use copy method of `Filehandler` class.
- 8) And just providing the *source* and *destination* as an argument of *copy* method.

To save the screenshot in the system(pc)

```
TakesScreenshot scrshot =(TakesScreenshot)driver;
```

```
File source = scrshot.getScreenshotAs(OutputType.FILE);
```

By this screenshot will taken in the form of file

```
File destination = new File ("E:\\desktop\\Katraj\\  
OctBatch \\Selenium\\Screenshots\\facebook.png");
```

By this path will decide for store the file
But between source and destination no any connection so to it will happen by
`FileHandler.copy`

```
FileHandler.copy(source, destination);
```

Screenshot with the help of method:

```
public static void captureScreenshot(WebDriver driver,String filename) throws IOException {
```

```
TakesScreenshot scrshot = (TakesScreenshot)driver;
```

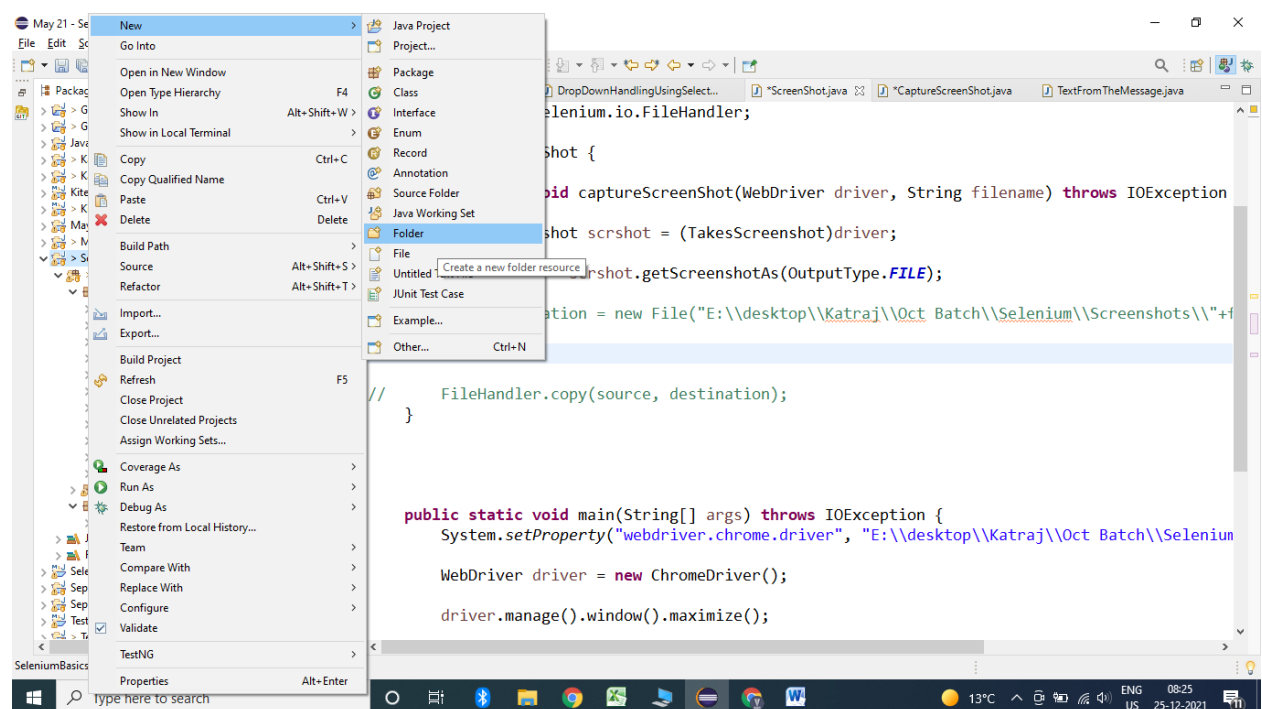
```
File source = scrshot.getScreenshotAs(OutputType.FILE);
```

```
File destination = new File("E:\\desktop\\Katraj\\Oct Batch\\Selenium\\Screenshots\\"+filename+".png");
```

```
FileHandler.copy(source, destination);}
```

To save the screenshot in the project :

Create a new folder inside the project:



Example:

```
public static void captureScreenshot(WebDriver driver,String  
filename) throws IOException {
```

```
TakesScreenshot scrshot = (TakesScreenshot)driver;
```

```
File source = scrshot.getScreenshotAs(OutputType.FILE);
```

```
String path =  
System.getProperty("user.dir")+"\\CapturedScreenshot\\"+filename  
+".png";
```

project path

foldername in project

filename.png

```
File destination = new File(path);
```

```
FileHandler.copy(source, destination);
```

```
}
```

Here system.getproperty() will provide the path of project in the machine.

Scrolling in selenium:

We can perform the scroll using selenium with the help of *JavaScriptExecutor* interface. *WebDriver* doesn't have necessary unimplemented methods to scroll over the page hence we have to type cast the driver object type to *JavaScriptExecutor* and call *executeScript* method.

Example- *To scroll by pixel:*

Orangehrsmlogin code

```
WebElement loginbutton = driver.findElement(By.className("button"));
```

```
loginbutton.click();
```

```
driver.findElement(By.xpath("//*[@id='menu_admin_viewAdminModule']"))).click();
```

```
JavascriptExecutor js = (JavascriptExecutor)driver;
```

To scroll by pixel value in forward and reverse direction

```
js.executeScript("window.scrollTo(0,1000)");
```

```
Thread.sleep(2000);
```

```
js.executeScript("window.scrollTo(0,-500)");
```

To perform scroll to a particular webElement:

To scroll until a webElement:

```
WebElement checkbox = driver.findElement(By.xpath("//*[@id='ohrmList_chkSelectRecord_36']"));
```

```
js.executeScript("arguments[0].scrollIntoView();", checkbox);
```


Scroll by using JavascriptExecutor:

```
WebDriver driver = new ChromeDriver();

driver.get("https://zoom.us/signup");

driver.findElement(By.xpath("//*[@id='select-0']")).click();

WebElement octmonth = driver.findElement(By.xpath
    ("//*[@id='select-item-select-0-9']"));

JavascriptExecutor js = (JavascriptExecutor)driver;

js.executeScript("arguments[0].click();", octmonth);

}
```

Sendkeys using java script executor:

```
WebDriver driver = new ChromeDriver();

driver.get("https://opensource-demo.orangehrmlive.
    com/index.php/dashboard");

// by name

WebElement username = driver.findElement (By.name("txtUsername"));

JavascriptExecutor js = (JavascriptExecutor)driver;

js.executeScript("arguments[0].value='text to be written inside the
    textfield';", username);
```

Actions class:

To *perform mouse and keyboard operations* we use actions class of selenium.

To perform the mouse action we need to create the object of Actions class which accept the web driver argument.

Methods or functions of Actions class

- **Move to element:** It is used to move the pointer(Hover) control to specific element And it accept the Web Element as an argument.
- **Click :** It is used to perform the *left click* operation of mouse.
- **Double click :** It is used to perform the *left double click* operation of mouse
- **Context Click :** It is used to perform the *right click* operation of mouse.
- **Build :** It is used to combine the multiple actions of Actions class in single statement.
- **Perform:** It is used to execute the each action or operation .
- **Drag And Drop :** It is used to drag element on another element.
- **Click and hold :** it is used to click and hold the element

1.Keyboard operations:

- **Example to press Tab key :**

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

```
WebElement firstname = driver.findElement (By.name("firstname"));
```

```
Actions act = new Actions(driver);
```

```
act.click(firstname).perform();
```

```
act.sendKeys("Velocity").sendKeys(Keys.TAB).sendKeys("Corporate").sendKeys(Keys.TAB).sendKeys("9988776655").build().perform();
```

- **Example to press Arrow Down key and Enter:**

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

WebElement searchfield = driver.findElement (By.xpath("//*[@name='q']"));

Actions act = new Actions(driver);

act.click(searchfield).sendKeys("selenium").build().perform();

Thread.sleep(2000);

act.sendKeys(Keys.ARROW_DOWN).sendKeys(Keys.ARROW_DOWN).

sendKeys(Keys.ARROW_DOWN).sendKeys(Keys.ENTER). build().perform();
```

- **WAP to handle Autosuggestions using selenium:**

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

WebElement searchfield = driver.findElement(By.xpath("//*[@name=\"q\"]"));

Actions act =new Actions(driver);

act.click(searchfield).sendKeys("selenium").build().perform();

Thread.sleep(3000);

List<WebElement> searchresults = driver.findElements
    (By.xpath("//*[@jsname='bw4e9b']/span['selenium']"));

for(WebElement searchresult:searchresults)
{
    String value = searchresult.getText();

    System.out.println(value);

    If ( value.contains ("testing") ) {

        searchresult.click();

        break ;
    }
```

- **Example to perform copy and paste:**

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

```
WebElement firstname = driver.findElement (By.name("firstname"));
```

```
Actions act = new Actions(driver);
```

```
act.click(firstname).sendKeys("Velocity").keyDown(Keys.CONTROL)  
.sendKeys("a").keyUp(Keys.CONTROL).build().perform();
```

```
act.keyDown(Keys.CONTROL).sendKeys("c").keyUp(Keys.  
CONTROL).build().perform();
```

```
act.sendKeys(Keys.TAB).keyDown(Keys.CONTROL).sendKeys("v").keyUp  
(Keys.CONTROL).build().perform();  
}
```

<https://github.com/SeleniumHQ/selenium/tree/trunk/java/src/org/openqa/selenium>

2. Mouse specific actions using Actions class:

To perform hover we should use moveToElement():

After login oragnehrm site

```
WebElement admintab = driver.findElement(By.xpath  
("//*[ @id='menu_admin_viewAdminModule']"));
```

```
Actions act = new Actions(driver);
```

```
act.moveToElement(admintab).perform();
```

To perform right click using selenium: we should use `contextClick()` method.

```
WebElement myinfo = driver.findElement(By.xpath  
    ("//*[ @id='menu_pim_viewMyDetails']"));
```

```
Actions act = new Actions(driver);
```

```
act.contextClick(myinfo).perform();
```

to handle the windows operation we have to Robot class

```
Robot robo = new Robot();
```

```
robo.keyPress(KeyEvent.VK_DOWN);  
robo.keyPress(KeyEvent.VK_DOWN);  
robo.keyPress(KeyEvent.VK_ENTER);
```

To perform Double click using selenium: we should use `doubleClick()` method.

```
driver.get("https://chercher.tech/practice/practice-pop-ups-selenium-  
webdriver");
```

```
WebElement doubleclickbutton = driver.findElement(By.xpath  
    ("//*[ @id='double-click']"));
```

```
Actions act = new Actions(driver);
```

```
act.doubleClick(doubleclickbutton).perform();
```

Iframe : It is a special functionality that has been attached to a webpage and Displaying webpage as a part of another webpage is called as I-frame. I-frame will be created by using tagname 'iframe'.

If we have an element inside the iframe then we have to first go inside it and then perform the action. To find out the iframe first we should search a tagname by <iframe>.

Q. How to handle i-frame using selenium webdriver?

1. To handle i-frame using selenium webdriver, we have to change the focus of selenium from main webpage to part of that webpage (to the i-frame). For this purpose we have to use switch to frame function.
2. We can switch to frame by using three methods. By index, by id and by name. generally we are not using by index method. to get name or id we have to find html tree diagram for that frame. And pass that name or id as string in signature of switch to frame function.
3. Then our focus moves from main page to that frame. We can perform action on the element present on that i-frame.
4. Sometimes it is possible to have frame within frame. We need to follow same procedure.
5. If we want to go back to one frame only then use function switch to parent frame. i.e `parentFrame()`.
6. If we want to go directly to main / default page then use switch to default content function. i.e `defaultContent()`.

- **Slider component: by using click and hold method**

```
driver.get("https://jqueryui.com/slider/");

WebElement iframe = driver.findElement(By.xpath
    ("//*[ @class='demo-frame']"));

driver.switchTo().frame(iframe);

WebElement slider = driver.findElement(By.xpath
    ("//*[ @class='ui-slider-handle ui-corner-all ui-state-default']"));

Actions act = new Actions(driver);

act.clickAndHold(slider).moveByOffset(285,0).release().build()
    .perform();
}
```

- **Drag and Drop:**

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

```
WebElement iframe = driver.findElement(By.xpath ("//*[@class='demo-frame']"));
```

```
driver.switchTo().frame(iframe);
```

```
WebElement dragobject =driver.findElement(By.xpath("//\*\[@id='draggable'\]"))
```

```
WebElement dropobject =driver.findElement(By.xpath("//\*\[@id='droppable'\]"))
```

```
Actions act = new Actions(driver);
```

```
act.dragAndDrop(dragobject, dropobject).perform();
```

```
driver.switchTo().parentFrame();
```

```
act.dragAndDropBy(dragobject, 100, 0).perform();
```

```
driver.findElement(By.xpath("//\*\[@class='menuitem'\]/a\[text\(\)='Demos'\]"))  
.click();
```

```
}
```

Handling Pop up in selenium:

Pop ups are small separate windows which open when we performs action on webpage. If we can't handle the popup we are not able to move forward. If we able to perform inspect on web element then we can handle popup by using selenium but if we are not able to inspect web element then such element can't handled by using selenium. We have to use some indirect approaches such as use of interfaces, auto IT toll and robot class .

There are 2 types of pop up:

1. Alert pop up.
2. Child browser popup.

1. Alert pop up: We can't inspect the elements present on alert popup. This popup consists of fields such as ok or cancel button and text is also present on that popup. Sometimes it also contains (?) or (!) symbols so we can only click on it but can't perform right click on that button.

To handle this popup follow given steps:

I. At first we need to switch the focus of selenium from the main page to the alert popup. For this purpose use syntax: `Alert alt = driver.switchTo.alert();`

II. Alert is an interface which consists of three abstract methods. **Accept**, **dismiss** and **get text**. **Accept method is used to click on OK button**,

dismiss method is used to click on CANCEL button whereas

get text method is used to get text present on the alert popup. Get text method will return string type of information

```
driver.get("https://chercher.tech/practice/practice-pop-ups-selenium  
webdriver");
```

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

```
alertbutton.click();
```



```
Thread.sleep(2000);
```

to get the text from the alert pop up

```
String textonalert = driver.switchTo().alert().getText();
```

```
System.out.println(textonalert);
```

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

```
driver.findElement(By.xpath("//*[@name='confirmation']")).click();
```

```
String confirmationtext = driver.switchTo().alert().getText();
```

```
System.out.println(confirmationtext);
```

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

To upload a file using selenium

```
WebElement uploadbutton = driver.findElement(By.xpath  
("//*[@name='upload']"));
```

```
uploadbutton.sendKeys("C:\\Users\\A\\Desktop\\TestDataSeptBatch.xlsx");  
}
```

2. Child browser popup: This pop up comes with the close button. We can inspect element present on child browser popup. But it is different from hidden division popup. These popup are generated in new tab or new window. This is child window of our main page. This popup contains address bar, maximize, minimize and close option as normal webpage.

To handle these popup follow the steps mentioned below:
<https://skpatro.github.io/demo/links/>

- At first we need to switch the focus of selenium from the main page to the alert popup. For this purpose use
syntax: `Alert alt = driver.switchTo().windows("String ID");`

- To get ID of main webpage use get windows handle function. But we required ID of child hence used getWindowHandles(). This will return set of ID's. III. As we know that index is absent in set collection hence we have to store these id in arraylist and get respective id by using index of that webpage.
- Then focus will shift to child page and you can perform actions on element present on that child webpage.
- To move back to previous page we have to use ID of main page, then we can perform action on main page

Example: simple program

```
driver.get("https://groww.in/");

driver.findElement(By.xpath("//*[@class='absolute-center
btn51ParentDimension']//span[text()='Login/Register']")).click();

driver.findElement(By.xpath("//*[@id='login_email1']")).sendKeys("abc@abc.co
m");

driver.findElement(By.xpath("//*[@class='rodal-close']")).click();

}
```

Window handling: To manage the multiple windows in selenium we have to handle it by the help of window id which is unique for every window.

To get parent id : driver.getWindowHandle();

To get multiple id : driver.getWindowHandles();

The return type of getWindoHandles is **String** whereas the the return type of the getWindowHandles method is **set of Strings**.

Example:

```
public static void main(String[] args) {  
    System.setProperty("webdriver.chrome.driver","E:\\desktop\\Katraj\\Oct  
Batch\\Selenium\\Chromedivers\\chromedriver.exe");  
  
    WebDriver driver = new ChromeDriver();  
  
    driver.manage().window().maximize();  
  
    driver.get("https://opensource-demo.orangehrmlive.com/index.php/dashboard");  
  
    String parentid = driver.getWindowHandle();  
  
    System.out.println(parentid);  
  
    driver.findElement(By.xpath("//*[text()='OrangeHRM, Inc']")).click();  
  
    Set<String> allwindid = driver.getWindowHandles();  
  
    for (String winid : allwindid) {  
        System.out.println(winid);  
  
        if (!(winid.equals(parentid)))  
        {  
            driver.switchTo().window(winid);  
            driver.get("https://www.google.co.in/");  
        }  
    }  
  
}
```

```

public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver","E:\\desktop\\Katraj\\Oct
Batch\\Selenium\\Chromedrivers\\chromedriver.exe");

    WebDriver driver = new ChromeDriver();

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

driver.get("https://opensourcedemo.orangehrmlive.com/index.php/dashboard"
);

    String parentid = driver.getWindowHandle();

    System.out.println(parentid);

driver.findElement(By.xpath("//*[text()='OrangeHRM, Inc']")).click();
driver.findElement(By.xpath("//*[text()='OrangeHRM, Inc']")).click();
driver.findElement(By.xpath("//*[text()='OrangeHRM, Inc']")).click();

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

int numberofwindows= allwindid.size();

    String [] windowid = new String[numberofwindows];

    int i=0;
    for(String id :allwindid)
    {
        windowid[i]= id;
        i++;
    }

    driver.switchTo().window(windowid[2]);

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

}

```

Close() vs quit(): close it to close the current window which is in the focus but if we call quit() then it means closing all the already opened window.

Waits in selenium:

Synchronization : Matching of execution speed of test script with application or browser speed is known as synchronization. And The action which got perform inorder to maintain the synchronization between the webpage and the code is refers to waits in selenium.

There are two types of waits :

a. static wait.: This is a wait which waits for the specified time irrespective of the actions is getting perform over the webpage. To perform this we have to thorws InterruptedException first.

Example: Thread.sleep(time in milliseconds);

b. Dynamic wait: These are the waits which applies to the webpage based on condition because of which they are specific to the webelement condition.

There are 3 types of dynamic waits:

1. Implicit wait
2. Explicit wait
3. Fluentwait

1. implicit wait: This is also known as global wait because once we write into the class it will be applicable for all the webElements present in Script .

It wait till the elements not present over the page till the maximum configured time. When we are not able to perform the action (element is not available) then after the maximum configured time it throws NoSuchElementException.

In this wait we can wait for TimeUnit such as second,minute ,hour , day etc.

Example:

```
public static void main(String[] args) throws InterruptedException {  
    WebDriver driver = new ChromeDriver();  
    driver.manage().timeouts().implicitlyWait(60, TimeUnit.SECONDS);  
    driver.get("https://www.google.com/");  
    Actions act = new Actions(driver);  
    WebElement searchbox = driver.findElement(By.xpath("//*[@name='q']"));  
    act.sendKeys(searchbox, "India").perform();  
    act.sendKeys(Keys.ENTER).perform();  
    driver.findElement(By.xpath("//*[text()='State Bank of India']")).click();  
}
```

2.Explicit wait:

Explicit wait is use when we don't know the exact time taken by web element to load on the browser.

In explicit wait there are two parameters for waiting time

Maximum Waiting Time which is in seconds.

Condition (static method of Expected Conditions Class)

Example:

- If waiting time is 15 seconds and condition is Visibility of element located by , so selenium will verify visibility of element (condition) on every 500 milliseconds intervals and if element is visible then waiting will be over.
- If Condition is not true within maximum waiting time 15 seconds. Then we wil get TimeoutException.
- Explicit wait will applicable to specific elements
- If suppose 15 seconds then in between 15 seconds element will check $15 / 500 \text{ Milliseconds} = 15 \text{ sec} / 0.5 \text{ sec} = 30 // 30$ times it check that element

- In this we provide the time only in seconds.
- For applying explicit wait we have to create an object of WebDriverWait.

Example

```

public static void main(String[] args) throws InterruptedException {
    System.setProperty("webdriver.chrome.driver", "E:\\desktop\\Katraj\\Oct
    Batch\\Selenium\\Chromedrivers\\chromedriver.exe");

    WebDriver driver = new ChromeDriver();

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

    driver.get("https://chercher.tech/practice/explicit-wait");

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

    WebDriverWait wait = new WebDriverWait(driver, 50);

    // wait.until(ExpectedConditions.elementToBeClickable(By.xpath("//*[@id='
    disable']"))));
    //
    // wait till button get enable
    // driver.findElement(By.xpath("//*[@id='enable-button']")).click();

    // wait till alert is available
    driver.findElement(By.xpath("//*[@id='alert']")).click();

    wait.until(ExpectedConditions.alertIsPresent());

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

    driver.findElement(By.xpath("//*[@id='checkbox']")).click();
    // wait till checkbox got selected

    wait.until(ExpectedConditions.elementSelectionModeToBe(By.xpath("//*[@id='ch
    ']), true));

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

    // condition in which text to be change by value
    wait.until(ExpectedConditions.textToBe(By.xpath("//*[@id='h2']"),
    "Selenium WebDriver"));
}

```

3.Fluent wait: This is an advance wait of explicit wait in which we can configure time unit for timeout of the wait which was not available in explicit wait.

We can also configure the polling interval also in this wait.

Fluent Wait can define the maximum amount of time to wait for a specific condition and frequency with which to check the condition before throwing an “ElementNotVisibleException” exception.

Example:

```
public static void main(String[] args) throws InterruptedException {  
    System.setProperty("webdriver.chrome.driver", "E:\\desktop\\Katraj\\Oct  
Batch\\Selenium\\Chromedrivers\\chromedriver.exe");
```

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

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

```
    driver.get("https://chercher.tech/practice/explicit-wait");
```

```
    FluentWait<WebDriver> wait = new FluentWait<WebDriver>(driver)
```

```
        .withTimeout(Duration.ofSeconds(60))
```

```
        .pollingEvery(Duration.ofSeconds(9));
```

```
    driver.findElement(By.xpath("//*[@id='populate-text']")).click();
```

```
    wait.until(ExpectedConditions.textToBe(By.xpath("//*[@id='h2']"), "Selenium  
Webdriver"));
```

```
}
```

Difference between implicit and Explicit

Sr no	Implicit	Explicit
1	NoSuchElementException	Timeout
2	Only one condition	Multiple condition
3	Applicable to all the elements	Applicable to only specific elements
4	No method support	Many methods are available

How to fetch the data from the excel sheet ?

Parameterization: The process of fetching the data from external source and use this data in selenium script, is called as parameterization. These external sources can be Excel sheet, CSV file, Test NG data provider etc. To fetch data from excel sheet we have to follow following steps:

Download Apache POI jar file and add these jar files in your current project. Follow the same procedure which we follow while adding the selenium jar file. Right click on your project name. Click on build path. Click on configure build path then new window will open. Click on external jar and then add external jars. Navigate to your location and add all jars and click on apply and close.

Create a excel sheet and some data on that sheet. Save this sheet at any location in your computer. Shift + Right click on that excel sheet name then you can copy it as a path.

Using XSSFWorkbook class:

- To Fetch the data from the excel sheet we have to use XSSFWorkbook class before that first of all take a path of the excel sheet by using system.getProperty and provide this path as an argument of File class then use FileInputStream class .
- provide reference of FileInputStream class as an argument to XSSFWorkbook class and use gesheet method to get the specific sheet from excel sheet .
- we have to use formatCellValue(); method of DataFormatter class to get the any type of data in the string format .
- and then by using getrow and getcell method we can fetch the data from excel sheet.

```
public static void main(String[] args) throws IOException {  
    String path = System.getProperty("user.dir")+"//Testdata.xlsx";  
  
    File src = new File(path);  
  
    FileInputStream fis = new FileInputStream(src);  
  
    XSSFWorkbook wb = new XSSFWorkbook(fis);  
  
    XSSFSheet sh1 = wb.getSheet("TestSheet1");  
  
    String value = sh1.getRow(5).getCell(1).getStringCellValue();  
  
    System.out.println(value);  
}
```

To get the data from excel sheet with any type e.g. numeric , Boolean . so that we have to use DataFormatter class ,It is a class which has method called as formatCellValue(); which is used to format the data in String form.And Return the string value.

```
public static void main(String[] args) throws IOException {  
  
    String path = System.getProperty("user.dir")+"//Testdata.xlsx";  
  
    File src = new File(path);  
  
    FileInputStream fis = new FileInputStream(src);  
  
    XSSFWorkbook wb = new XSSFWorkbook(fis);  
  
    XSSFSheet sh1 = wb.getSheet("TestSheet1");  
  
    String value = sh1.getRow(6).getCell(0).getStringCellValue();  
  
    DataFormatter df = new DataFormatter();  
  
    String cellvalue = df.formatCellValue(sh1.getRow(6).getCell(0));  
  
    System.out.println(cellvalue);  
}
```

How to fetch the data from the properties file

1. Create the properties file:- Right click on Project—New—File

2. Enter the name of file and click on Finish.

- To read the data from Properties file we have to use FileInputStream class and provide the path of that file as an arguments .
- creates an object of Properties class where load is present then provides reference of FileInputStream to the load method so that file will ready to fetch the data
- then by using getProperty method of Properties class we can get the data from File.

```
public static void main(String[] args) throws IOException {  
  
    Properties prop = new Properties();  
  
    String path = System.getProperty("user.dir")+"\\config.properties";  
  
    FileInputStream fis = new FileInputStream(path);  
  
    prop.load(fis);  
  
    String url = prop.getProperty("testsiteurl");  
  
    System.out.println(url);  
}
```

For using same script with number of time we have to use static return type method

```
public class ReadDataFromProperties {  
  
    public static String readConfigData(String datatoberead) throws IOException{  
  
        Properties prop = new Properties();  
  
        String path = System.getProperty("user.dir")+"\\config.properties";  
  
        FileInputStream fis = new FileInputStream(path);  
  
        prop.load(fis);  
  
        String value = prop.getProperty(datatoberead);  
  
        System.out.println(value);  
  
        return value;  
    }  
}
```

TestNG:

TestNG is a framework which is mainly used to control the execution. It is basically termed as Testing with new generation. The features of **TestNG** are

1. We can represent or define a test case.
2. We can decide the sequence of execution.
3. We can decide the number of execution of test case.
4. After the execution a detailed analysis can be obtained in the form of report.
5. Pass/Fail/skip criteria can be defined for a test case.
6. Parallel execution can be done.

To use the TestNg we should first install it:

1. Go to help--- Eclipse market place
2. Click on eclipse market place and type testng
3. Click on install button for testng.
4. Perform the positive options and restart the eclipse.

Different types of annotations and the keywords in TestNG

Annotations in testng: Annotations are the one which describe the pre and post condition for a test case. To represent a test case in TestNG we have to use **@Test** annotation.

1. **@BeforeMethod:** This will execute before every test case.
2. **@AfterMethod:** This will execute after every test case.
3. **@BeforeClass:** This will execute before the execution of anything inside a test class.
4. **@AfterClass:** This will execute after the execution of everything inside a test class.

5. @BeforeTest: This will execute before the execution of <test>. In whatever we write it and that class is at whichever sequence in xml then also first @BeforeTest will get execute.

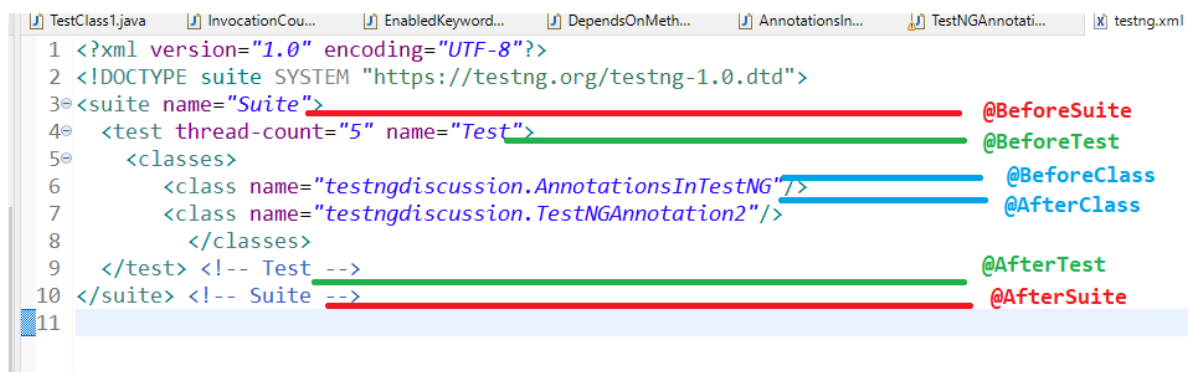
6. @AfterTest: This will execute after the execution of <test>. In whatever we write it and that class is at whichever sequence in xml then also at last @AfterTest will get execute.

7. @BeforeSuite: This will execute before the execution of <suite> irrespective of any class in which we have mentioned it.

8. @AfterSuite: This will execute after the completion of execution of <suite> irrespective of any class in which we have mentioned it.

9.@Parameeter:Parameter in TestNG: Whenever we want to take the data from the xml file then we should use @parameter annotation. To use @parameter we have to define the value into the xml by using `<parameter name="buildversion" value="0.0.56"></parameter>`

Sequence of execution of annotations.



Before suite annotation
before test anotation
Before class annotation
Before method annotation
Test case
After method annotation
~~Before method annotation~~
After class annotation
After test annotation
After suite

Keywords used with annotation in TestNG:

1. priority: We can decide the sequence of execution based on the priority of the test case.

Rules for priority:

- priority of a test case can be +ve , -ve or 0 but it cannot be in decimal. Whichever test case has lowest will execute the earliest.
- If we are not define the priority of a testcase then by default it will be 0.
- We can have duplicate priority of two test case.

2.invocationCount: This is a keyword which executes a testcase multiple times.

3. enabled: This key word will allow the test case to execute every time without any condition if we write enabled= true and will not execute the testcase if we write enabled =false.

4. dependsOnMethods: This is a keyword which execute a test case only when the dependent test case get execute otherwise this test case gets skip.

5.timeOut: if a test contains multiple test methods and we want the test to be complete ,in a timeduration and if it takes more time then test should auomatically fail then we use “timeout” keyword . in this time is in milliseconds

Reporter.log method():

This method will print the content in the console as well as in the emailable-report of testng.

If the Boolean value in reporter.log method is true then text will get print into console and report but if it is false then it will print into only report.

To check the report of execution :

Refresh the project --- inside the test-output folder – open the emailablereport.html this can be opened in the browser .

Inclusion and Exclusion of method from the class:

a. include a method: We can include a specific method from a class by using <include> tag in xml file.

b. Exclusion of method: We can exclude a method from the class execution by using <exclude> and execute the class.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite">
    <test thread-count="5" name="Test">
        <classes>
            <class name="testngdiscussion.InclusionAndExclusionOfTestCase">
                <methods>
                    <include name="login"></include> -->
                    <exclude name="profile"></exclude>
                </methods>
            </class>
            <class name="testngdiscussion.TestClass4"/>
        </classes>
    </test> <!-- Test -->
</suite> <!-- Suite -->
```

Grouping of test cases:

We can group the testcases by their category and include them according to the requirement.

public class GroupingOfTestCases {

```
@Test(groups = "Sanity", priority = 1)
public void login()
{
    System.out.println("Login test case");
    Reporter.log("Login test case", false);
}
@Test(groups = "Regression", priority = 2)
public void dashboard()
{
    System.out.println("Dashboard test case");
    Reporter.log("Dashboard test ", true);
}
```

```

@Test(groups = "Sanity", priority = 3)
public void profile()
{
    System.out.println("Profile test case");
    Reporter.log("profile test ", false);
}
@Test(groups = "Functional", priority = 4)
public void avatar()
{
    System.out.println("avatar test case");
    Reporter.log("avatar test ", false);
}
@Test(groups = {"Sanity", "Regression"}, priority = 5)
public void time()
{
    System.out.println("Time test case");
}
@Test(groups = {"Functional", "Regression"}, priority = 6)
public void recruitment()
{
    System.out.println("recruitment test case");
}

@Test(groups = "Sanity", priority = 7)
public void performance()
{
    System.out.println("performance test case");
}
}
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite">
    <test thread-count="5" name="Test">
        <groups>
            <run>
                <include name="Sanity"></include>
                <include name="Functional"></include>
            </run>
        </groups>
        <classes>
            <class name="testngdiscussion.GroupingOfTestCases" />
        </classes>
    </test> <!-- Test -->
</suite> <!-- Suite -->

```


Assertions: These are the check points through which we evaluate a particular testcase is pass/fail based on actual and expected result.

There are 2 types of assertions:

- a. Hard assert
- b. Soft assert

a. Hard assert: In this if the assertion got failed then we mark that test case as failed without executing any further steps.

Note: If Hard assert got failed then the code below to that fail condition will not execute it directly terminate the test case.

Note : Whenever a test case fails then we get assertion error instead of Exception.

Methods in Hard Assert class are static methods:

a. Assert.assertTrue(String message, Boolean condition,): This method expect the condition to be appear after the evaluation as true then this test case marked as pass other wise fail.

b. Assert.assertFalse(Boolean condition, String message): This method expect the Boolean condition to be false inorder to mark this test case as Pass other wise it will fail the test case.

c. Assert.assertEquals(Any data type actual, Any data type expected): This assertion will check the actual and expected condition/data are equal or not if they are not equal then it fail the test case.

Note: the data in actual and expected result that we are considering should be same data type.

b.SoftAssert: It is a class in which the result of test case will not mark instantly as the assertion got failed it will continue its execution further till we don't call `assertAll()`. It is also known as verify.

The methods inside the softassert is class are nonstatic.

Listeners in TestNG:

Listeners are the one which monitors the test cases based on events which will get occur as a result of testcase i.e testcase pass, fail, skipped, start finish etc.

For implementing listeners we have to implement an interface called `ITestListener(I)`.

```
public class ListenersTest{  
  
    static WebDriver driver;  
    @Test  
    public void logintestcase(){  
  
    }  
}
```

```
public class TestNGListeners extends ListenersTest implements ITestListener{
```

```
    @Override
```

```
    public void onTestStart(ITestResult result) {
```

```
        System.out.println("Test started : "+ result.getName());
```

```
    }
```

```
    @Override
```

```
    public void onTestSuccess(ITestResult result) {
```

```
        System.out.println("Test case passed: "+result.getName());
```

```
    }
```

```
    @Override
```

```
    public void onTestFailure(ITestResult result) {
```

```
        System.out.println("Testcase failed: "+result.getName());
```

```
        try {
```

```
            ScreenShot.captureScreenShot(driver, result.getName());
```

```
        } catch (IOException e) {
```

```
        }
```

```
    }
```

```
    @Override
```

```
    public void onTestSkipped(ITestResult result) {
```

```
        System.out.println("Test case skipped :"+result.getName());
```

```
    }
```

```

@Override
public void onTestFailedButWithinSuccessPercentage(ITestResult result) {

}

@Override
public void onTestFailedWithTimeout(ITestResult result) {

}

@Override
public void onStart(ITestContext context) {
    System.out.println("Test started :"+context.getName());
}

@Override
public void onFinish(ITestContext context) {

    System.out.println("Test completed :"+context.getName());
}
}

```

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite">
  <listeners>
    <listener class-name="testngdiscussion.TestNGListeners"></listener>
  </listeners>
  <test thread-count="5" name="Test">
    <classes>
      <class name="testngdiscussion.ListenersTest"/>
    </classes>
  </test> <!-- Test -->
</suite> <!-- Suite -->

```

Parallel execution: We can execute the testcases, classes and <test> in parallel through TestNG. For that we have to write parallel = “methods”, “classes” and parallel = “tests”.

a. Execution of test case in parallel:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite">
  <test thread-count="5" name="Test" parallel = "methods">
    <classes>
      <class name="testngdiscussion.ParallelTestCaseExecution"/>
    </classes>
  </test> <!-- Test -->
</suite> <!-- Suite -->
```

Test	# Passed	# Skipped	# Retried	# Failed	Time (ms)	Included Groups	Excluded Groups
Suite							
Test	2	0	0	0	52,098		
Class				Method		Start	Time (ms)
Suite							
Test — passed							
testngdiscussion.ParallelTestCaseExecution				launchBrowserForFacebook		1641956450667	51996
				launchChromeForOrange		1641956450667	43727

Test

testngdiscussion.ParallelTestCaseExecution#launchBrowserForFacebook

[back to summary](#)

testngdiscussion.ParallelTestCaseExecution#launchChromeForOrange

[back to summary](#)

Execute the testcases in parallel which are in different classes:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite">
  <test thread-count="5" name="Test" parallel = "methods">
    <classes>
      <class name="testngdiscussion.ParallelTestCaseExecution"/>
      <class name="testngdiscussion.ParallelTestCase2"/>
    </classes>
  </test> <!-- Test -->
</suite> <!-- Suite -->
```

Execution of classes in parallel:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite">
  <test thread-count="5" name="Test" parallel = "classes">
    <classes>
      <class name="testngdiscussion.ParallelExecutionClass1"/>
      <class name="testngdiscussion.ParallelExecutionClass2"/>
    </classes>
  </test> <!-- Test -->
</suite> <!-- Suite -->
```

Execution of test tag in parallel manner:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite" parallel="tests">
  <test thread-count="10" name="Module 1">
    <classes>
      <class name="testngdiscussion.ParallelExecutionTesttag1"/>
      <class name="testngdiscussion.TestClass3"/>
    </classes>
  </test> <!-- Test -->
  <test thread-count="5" name="Module 2">
    <classes>
      <class name="testngdiscussion.ParallelExecutionTestTag2"/>
      <class name="testngdiscussion.TestClass4"/>
    </classes>
  </test> <!-- Test -->
</suite> <!-- Suite -->
```

Parallel execution of single test case but with different url:

```
public class TestTagExecutionActualCase {
    @Parameters("url")
    @Test
    public void navToUrl(String website)
    {
        System.setProperty("webdriver.chrome.driver", "E:\\desktop\\Katraj\\Oct
        Batch\\Selenium\\Chromedrivrs\\chromedriver.exe");

        WebDriver driver = new ChromeDriver();

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

        if(website.contains("facebook")){

            driver.get("https://www.facebook.com/");
        }
        else if (website.contains("google")) {

            driver.get("https://www.google.com/");
        }
    }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite" parallel="tests">
  <test thread-count="5" name="facebook">
    <parameter name="url" value="https://www.facebook.com/"></parameter>
    <classes>
      <class name="testngdiscussion.TestTagExecutionActualCase"/>
    </classes>
  </test> <!-- Test -->
  <test thread-count="5" name="Google">
    <parameter name="url" value="https://www.google.com/"></parameter>
    <classes>
      <class name="testngdiscussion.TestTagExecutionActualCase"/>
    </classes>
  </test> <!-- Test -->
</suite> <!-- Suite -->
```

POM : It is an abbreviation of Page object model . It is basically a design pattern in which we create pages in one package and Pages contains all the methods in which we have handled all the scenarios. Methods inside the page classes will get call inside the test class to represent as a step inside it @Test annotation we used which is a test case.

POM we have implemented on the basis of PageFactory which provides the initialization of the webelements at the constructor and at the method, we just have to use them by using @FindBy annotation and declared them as a private. To run the POM class we need to create separate test class. Test Class contains the all the Navigation steps to execute the test.

As per our implementation we are setting the properties for browser driver and launch it in @BeforeSuite annotation.

Through @BeforeClass annotation we are here creating the objects which are used in the corresponding test classes.

BaseTest is a class here which contains initBrowser() method which is responsible to set the properties of browser and create an object of Browser's driver(ChromeDriver). And as it is the parent class of all the test classes all the objects will be created under @BeforeClass annotations methods which is later used to call the methods of the page classes in the test classes.

Framework:

A structure which is able to perform the activities by its own without depending on any. In this we don't have to worry about the supporting libraries as those will directly get called when it is required.

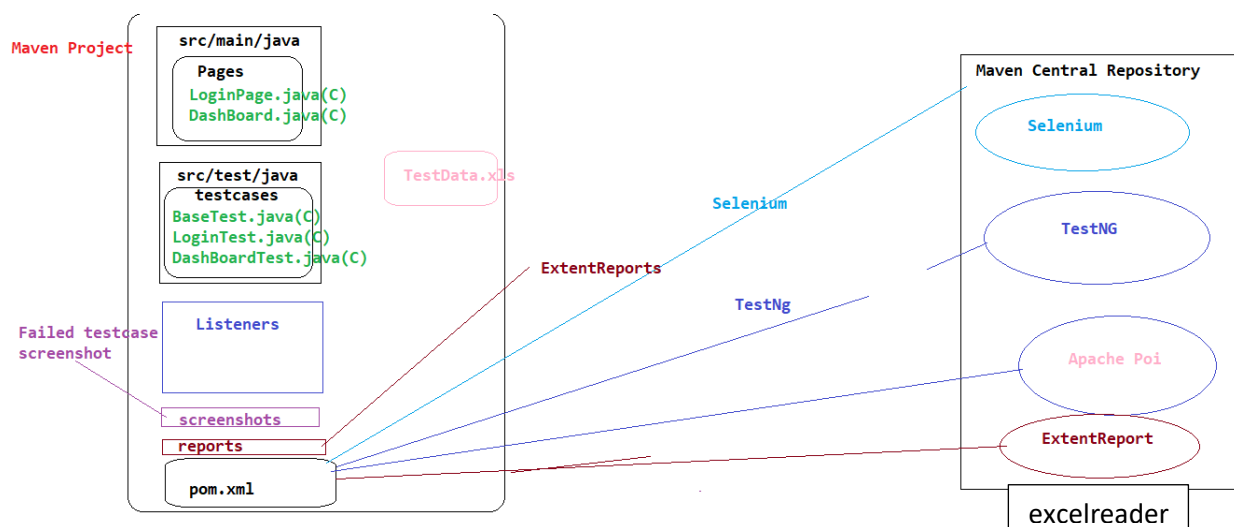
There are 3 types of framework:

1. **Data Driven Framework:** In this we will take the data from the external source and use it in the framework as test data.
2. **Keyword driven framework:** A particular key decides the flow of execution that we will keep somewhere in the project.
3. **Hybrid framework:** Whenever we have the combination of more than one type of framework in the project then it is Hybrid framework.

The project that we are using is Maven Project. Maven is a build automation tool that means it automatically takes the libraries from the maven central repository to the project and use them .

Reason to use maven project:

1. All the jar files in the maven project gets download automatically with the required version once we configured them in pom.xml.
2. We have a well designed structure with respect to POM.
3. We can execute the project multiple ways i.e through eclipse, command prompt and through Jenkins.



Steps for creation of Maven Project:

1. Go to file – New--- Project:
2. Click on Maven and select Maven Project:
3. Click on Next button.
4. type org.apache.maven and select the below selected item from the list and click on Next.
5. Enter the name of group id and artifact id and click on Finish button:
6. Delete the packages from src/main/java and src/test/java.
7. we have to add dependencies and some imp plugin

Dependencies are like 1) maven selenium jar, 2) testNG, 3) WebDriver manager, 4) extent reports, 5) commons-io, 6) apache poi and ooxml

Plugins are like 1) maven-compiler plugin, 2) maven-sure-fire plugin

To execute the maven project:

1. When we are executing the project for the first time then we should use Run as Maven Install.
2. If we have executed once then for the next time we should use Maven Test.

Commands to execute the maven project:

1. **Maven Install:** This command will first download the source if required and then execute the project. If the run was successful then it will generate a jar file of that project.
2. **Maven Test:** This command will execute the project.
3. **Maven Build:** This command will execute the project according to the command.
4. **Maven clean:** This command will delete target folder.

ExtentReports in Maven Project:

Extent report is a third party tool which provides a html based report that contains the test case information and the screenshots if any of the test case got failed.

To apply the extent report we have to use dependencies: extent reports and common-io

Extent Reports code:

```
public class ExtentReportGen {
    static ExtentReports extent;

    public static ExtentReports extentReportGenerator()
    {
        String path =
System.getProperty("user.dir")+"//reports//zerodhareport.html";

        ExtentHtmlReporter reporter = new ExtentHtmlReporter(path);

        reporter.config().setReportName("ZerodhaTestReport");

        reporter.config().setTheme(Theme.STANDARD);

        extent = new ExtentReports();

        extent.attachReporter(reporter);

        extent.setSystemInfo("OS", "Windows");

        extent.setSystemInfo("Organization name", "Velocity");

        extent.setSystemInfo("Executed by", "Oct Batch");

        return extent;

    }
```

Note: For providing extra support features to the browser then we have to use ChromeOptions class. In that there is a method called addArguments().

Like :

```
ChromeOptions options = new ChromeOptions();

//      options.addArguments("--disable-notifications");
//
//      options.addArguments("--start-maximized");
//
//      options.addArguments("--incognito");

options.addArguments("--headless");

driver = new ChromeDriver(options);
```

OOPs concept in Selenium Framework

1. **Interface:** `WebDriver driver = new ChromeDriver();`

In this statement `WebDriver` is nothing but interface in selenium.

1. **UPCASTING:** `WebDriver driver = new ChromeDriver();`

above statement is nothing but UPCASTING in selenium.

2. INHERITANCE

We create a Base Class in the Framework to initialize `WebDriver` interface, `WebDriver` waits, Property files etc., in the Base Class.

We extend the Base Class in Tests Class. that is nothing but Inheritance in Selenium Framework.

POLYMORPHISM

Combination of overloading and overriding is known as Polymorphism.

`Reporter.log`, `Assert.assertTrue()`--- Overloading

Overriding – All the listeners method

3. METHOD OVERLOADING

We use implicit wait in Selenium. Implicit wait is an example of overloading. In Implicit wait we use different time stamps such as SECONDS, MINUTES, HOURS etc.,

A class having multiple methods with same name but different parameters is called Method Overloading.

eg. `driver.switchTo().frame()`: - String name, int index

4. METHOD OVERRIDING

Declaring a method in child class which is already present in the parent class is called Method Overriding.

5. ENCAPSULATION

All the POM classes in a framework are an example of Encapsulation. In POM classes,

we declare the data members using `@FindBy` and initialization of data members which are private through this we are achieving data hiding and calling of this private variables into the public method and those public method will get call inside test classes which is Abstraction.

In pagefactory will be done using Constructor to utilize those in methods.

Encapsulation is a mechanism of binding code and data together in a single unit.

Encapsulation is the process of wrapping up code and data together in a single unit. It is used to hide the data of a class from another class.

Encapsulation can be achieved when you declare all variables as private and a public method in a class to get the values of the variable.

6. ABSTRACTION

In Page Object Model design pattern, we write locators (such as id, name, xpath etc.,) in a Page Class.

We utilize these locators in pom class but we can't see these locators in the tests. Literally we hide the locators from the tests.

Abstraction is the methodology of hiding the implementation of internal details and showing the functionality to the users.

What are the different types of exception in selenium

_IllegalStateException: when driver path is not properly given.

Web Driver Exception : When URL is not in proper way

Unreachable Browser Exception : While executing the test script if the browser get interrupted are corrupt or crash then we get unreachable browser exception.

Not Connected Exception :When selenium jar file unable to interact with browser then we get not connected exception

No Alert Present Exception: When we try to perform the action on other POP ups using alert interface . When we try to use alert pop up but alert pop is not present in screen

Unhandled alert exception :When we try to perform action on main page without handling the alert pop up .

Unexpected tag name exception : When we try to customize list box Tag name of the list box other than Select. (Using select class) .

Invalid state exception : When we try to handle the disabled elements.

No such Element Exception : When XPATH is wrong and synchronization issue we will get this exception.

Stale Exception : While performing the action on web element selenium notice that element Is not present When web elements get refresh When web page is get refresh then DOM document object model get crash at that time we get stale exception .

Difference between no such element exception and stale exception : We get the no such element when selenium is going to use inspect the element We get stale exception when selenium is trying to perform action on element.

nullpointerExeption :- demo d = new demo();

i declare a reference variable i am not initialized it when a reference variable not initialized it point to null "syso (d):" then we get nullpointerExeption

When ask about which framework you are using .

We are using datadriven framework under the maven project . and we are using pom(pom of object module) as a design pattern.

Explain about your framework .

So when I have been assigned as a automation tester on particular domain the framework was already designed. So we are using maven project there are two folder separately ,src/main/java and src/test/java .

inside the src/main/java folder there is a pages package , we are writing all the pageclasses inside that and in pageclass we writes webelements constructor and required methods using pom design pattern with pagefactory class.

In another folder means src/test/java there are packages like testcases, extentlisteners and utility, inside testcases package we are writing all the testclasses ,so in test class we writes all the test which we implementing with the help of testNG @Test annotation. and in testcases package we have a Base class as well, inside that we writes a code to initialize the webdriver , browser , opening url, and also creation of objects of pageclass with using testNg annotations like @Before suite and @ before class etc.

Extentlistners package have Listener class which monitors on all testcases and extentreportsclass to generate the extentreport after the execution.

Utily package have classes like screenshot,excelreader and frequently used functions

Rather than that there is pom.xml to manage the all dependencies and plugin ,then testng.xml to define the order of execution . excel sheet for testcase data ,config,properties contains credential for login ,and inside the report folder we get the emailabe.html report.

If interviewer ask why did you used maven project

In the maven project pom.xml is there and inside it we can add dependencies of specific tools which we have to use and also we can change the version of tools and just after the updating the project all the jars gets download automatically . and main advantage is we can easily export the project in another system and just after updating the project we can execute the project so here we don't have need to download jars files and keep them inside the folder of the system .