**Automation testing**

Testing on application features with the help of automation tool & execute test script is called as automation testing.

**Disadvantage of Manual testing**

1. Compatibility testing is difficult

2. Test cycle duration will be increased

3. More efforts are required

4. Repetition testing is time consuming

**Advantage of automation testing**

1. Reusability of test script

2. Compatibility testing is easy / possible.

3. Less human efforts are required

4. Project duration will be reduced

5. To overcome the drawback of regression testing

6. Cost of project will be reduced

7. It is reliable & efficient

**➢ Some of the Automation tool**

1. Selenium

2. QTP

3. Sahi / Sahi pro

4. Selendroid

5. Appium

To run application in multiple browser (C.T) i.e., writing test script by using single browser but run the same script in multiple browser we need to use runtime polymorphism by using upcasting in selenium.

Web driver = new Chrome Driver ();

Create an object of Chrome Driver class with reference of web driver interface.

**Q. How to open a Browser in selenium.**

System.set.property(“ webdriver.chrome.driver”, “ path of .exe file”);

Web driver = new chrome driver();

• Create an object of chrome driver class & store it in 1 reference variable with reference of web driver interface.

• Define that we need of set path of chrome driver .exe file

**1. Selenium Architecture**



1. Search content is a super most interface which contains abstracts method & inheritance to web driver.

2. Web driver is an interface which contains abstracts method of search context & its own abstract method.

3. All the abstract method are overridden or implemented in selenium remote web driver class.

4. Selenium remote web driver: - It is a class which implements all the abstract method of both interfaces.

5. Selenium remote web driver class is extended to browser such as F.F driver, Chrome driver… etc.

Selenium flavors

1. Selenium IDE (integrated development environment) o We can run script in only firefox browser

o Record & playback option

o We can’t do compatibility testing

2. Selenium RC (remote control) o Support CT (Cross browser)

o We can run script in java only.

3. Selenium webDriver (Selenium tool)

o Support CT (Cross Browser)

o We can run script in multiple language

4. Selenium grid

5. Selendroid

* + **Q when we should do automation testing?**



Automation testing tool will able to perform testing an application but to perform any action as a TE we need to give command those command are called scripting.

➢ **Advantages of selenium**

1. Open source

2. Multi language support

3. Cross browser / compatibility is possible

4. Cross platform is auto platform

➢ **Disadvantage of selenium**

1. We can automate only web-based application

2. We can’t automate stand-alone application.

3. Can’t automate captcha.

4. Selenium will not support file uploading

5. Ad hoc test cases can’t automate.

6. Can’t read bar code.

To create a webpage we need to use tagname <html> and to create a component or element we need to use tagname <input>

To create a listbox we use <select> tag

To create a link we use <a> tag (anchor)

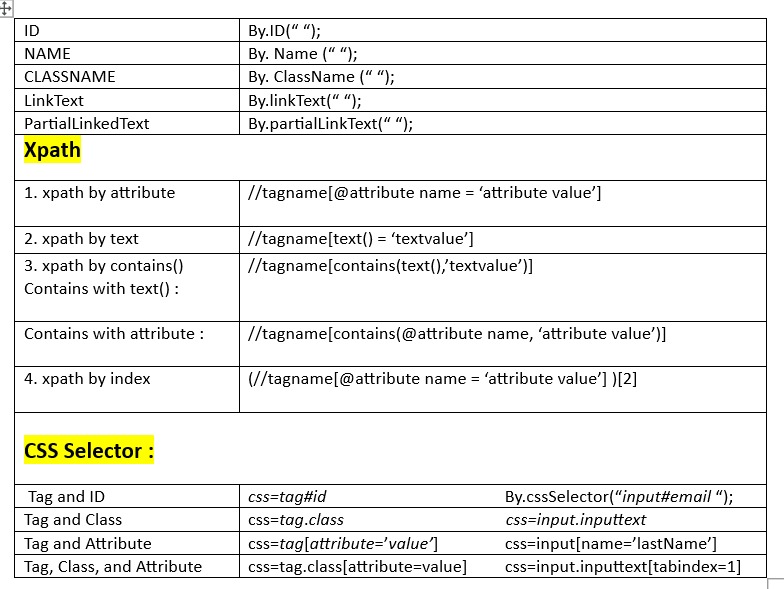
To create a web table we use <table> tag

**Locators :**

1. Locators are used to identify an element present in a webpage with the help of locator types.
2. To identify an element present in a webpage we need to use findElement() method which is present in WebDriver interface.
3. findElement() method will identify an element with the help of “By” class which contains static methods.
4. All the static methods present in a By class are known as locator types.

**There are different types of locators:**

1. ID
2. Class
3. Name
4. Tagname
5. Linked Text
6. Partial Linked Text
7. CSS Selector
8. X-path



**Disadvantages of Absolute xpath**

1. Absolute xpath is too lengthy and time consuming.
2. Identifying an element by developing html tree diagram is difficult.

**Absolute xpath**

Absolute xpath is used to navigate from root of parent to its **immediate** child. To achieve absolute xpath, we need to use ‘/’.

**/html/body/div[2]/input[1]**

**Relative xpath**

Relative xpath is used to navigate from parent to **any child**. To achieve relative xpath, we need to use ‘//’.

//div[2]//input[1]

html/body/section/div[1]/div[1]

//section[2]//div//div

|  |  |
| --- | --- |
| absolute xpath | Relative XPath |
| Starts with single forward / | Starts with double forward // |
| Slower, This is because of longer travesersing path. | relative XPath expressions can be faster, especially in large and complex DOM structures. |
| Absolute XPath provides the full path from the root element (the <html> tag) to the desired element. | Starts from a (any parent)specific element and navigates through the DOM hierarchy to locate the desired element. |
|  |  |
| Absolute XPath offering precision but less flexibility.  One drawback with the absolute xpath is that if there is any change in page structure ,our absolute xpath will become invalid. | Relative XPath is more flexible and adaptable to changes in the web page structure, |
| Relative XPath navigates elements based on relationships. Not impacted by change in page structure |
| Uses tag/nodes | Uses attributes |
| //html/body/div[1]/div[2]/button | /div[2]/button |

**How to handle dynamic XPath**

dynamic XPath expressions in automation testing is a common challenge, as web pages often generate elements with changing attributes or values. Dynamic XPath refers to XPath expressions that are subject to change due to factors like session IDs, timestamps, or random values. Here are strategies to handle this situation:

**Using Partial Matching:**

Utilize XPath functions like contains() or starts-with() to match a portion of the dynamic attribute value. For instance:

xpath

//input[contains(@id, 'dynamicPart')]

---- ------- ------ ------ ---- ------- ------ ------ ---- ------- ------ ------

**Webdriver:**

It is an interface use to perform action of **browser**.

**WebDriver Methods-:**

1.get()

2.close()

3.quit()

4.minimize()/maximize()

5.navigate()

6.gettitle()

7.getCurrentURL()

1. get- to enter url in browser or to open an application.

driver.get("https://vctcpune.com/");

2. close: to close the current tab of the browser opened by Selenium tool.

driver.close();

3. quit: to close the all the tabs present in browser opened by Selenium tool.

driver.quit();

4. maximize()/ minimize() - used to maximize/ minimize the browser window

note: can't minimize the browser(we can change size & position of the browser).

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

driver.manage().window().minimize();

5. navigate: this method is use to open an application, move forward, backward & refresh the webpage.  
 driver.get("URL”);

driver.navigate().to("URL");

driver.navigate().back();

driver.navigate().forward();

driver.navigate().refresh();

6. getTitle: this method is use to get title of a webpage. return type of getTitle method is String.

driver.get("https://vctcpune.com/");

Thread.*sleep*(300);

//System.out.println(driver.getTitle());

String Title = driver.getTitle();

System.*out*.println("Title is "+Title);

7. getCurrentURL: this method is use to get URL of a webpage. return type of getCurrentURL method is String

driver.get("https://vctcpune.com/");

Thread.*sleep*(400);

//System.out.println(driver.getCurrentUrl());

String url = driver.getCurrentUrl();

System.*out*.println("Url is "+url);

**WebElement methods:**

WebElement methods: WebElment methods are used to take actions on elements present in webpage

**1. Sendkeys():** This method is use to enter value in the input/text field

**2. Clear():**This method is use to clear value in the text field.

**3. Click():** Click method is use to click on buttons, links also use to select radio buttons & checkboxes.

**4. getText:** This method is use to get text present in a webpage. Return type of getText function is String.

**5. isEnabled():** This method is use to verify element is enabled or disabled. Return type of isenabled function boolean if element is enabled then it returns true otherwise it returns false

**6. isSelected():** This method is use to verify radio button/checkbox is selected or not. Return type of isSelected function is boolean. If radio button/checkbox is selected then it returns true otherwise it returns false.

**7. isDisplayed():** This method is use to verify element is present or not. Return type of isDisplayed function is boolean. if element is present then it returns true otherwise it returns exception.

**LIST BOX/ DROPDOWN HANDLING**

Process to handle list box/dropdown (Verify tag name is 🡪select)

1. Identify list box to be handled and store it in reference variable

**WebElement states=driver.findElement(By.xpath("//select[@name='States']"));**

1. Create an object of Select class which will accept WebElement as argument

**Select s= new Select(states);**

1. By using one of the select class **methods** we can select values form list box like

**1. selectByVisibleText: selectByVisibleText(String arg0)**

**2. selectByIndex: selectByIndex(int arg0)**

**3. selectByValue: selectByValue(String arg0)**

**Screenshot**

To take screenshot using selenium webdriver, we need to type cast driver object to Takes Screenshot **interface**.

**🡪 ((TakesScreenshot) driver)**

Then we need call function **getscreenshotAs();** there we need to pass the argument

**🡪Outputtype. FILE**

This method will return object of type

🡪**FILE.**

**ie File src= (TakesScreenshot) driver). getscreenshotAs (Output Type.FILE)**

When this will execute, it will take screenshot, but it will be available inside temporary memory

- To store screenshot in local drive (desired location ), we need to call a method

i-e. copy file () which is present inside FileHandler class.

🡪**FileHandler.copy (src, newfile("destinati path"));**

**Iframe/frame**

**Displaying a webpage as a part another webpage** is known as iframe. Iframe will be created by using tagname iframe.

Procedure to handle iframe🡪

To handle iframe we need to switch selenium focus from main page to frame by using

**driver.switchTo().frame("String NameorId”)**

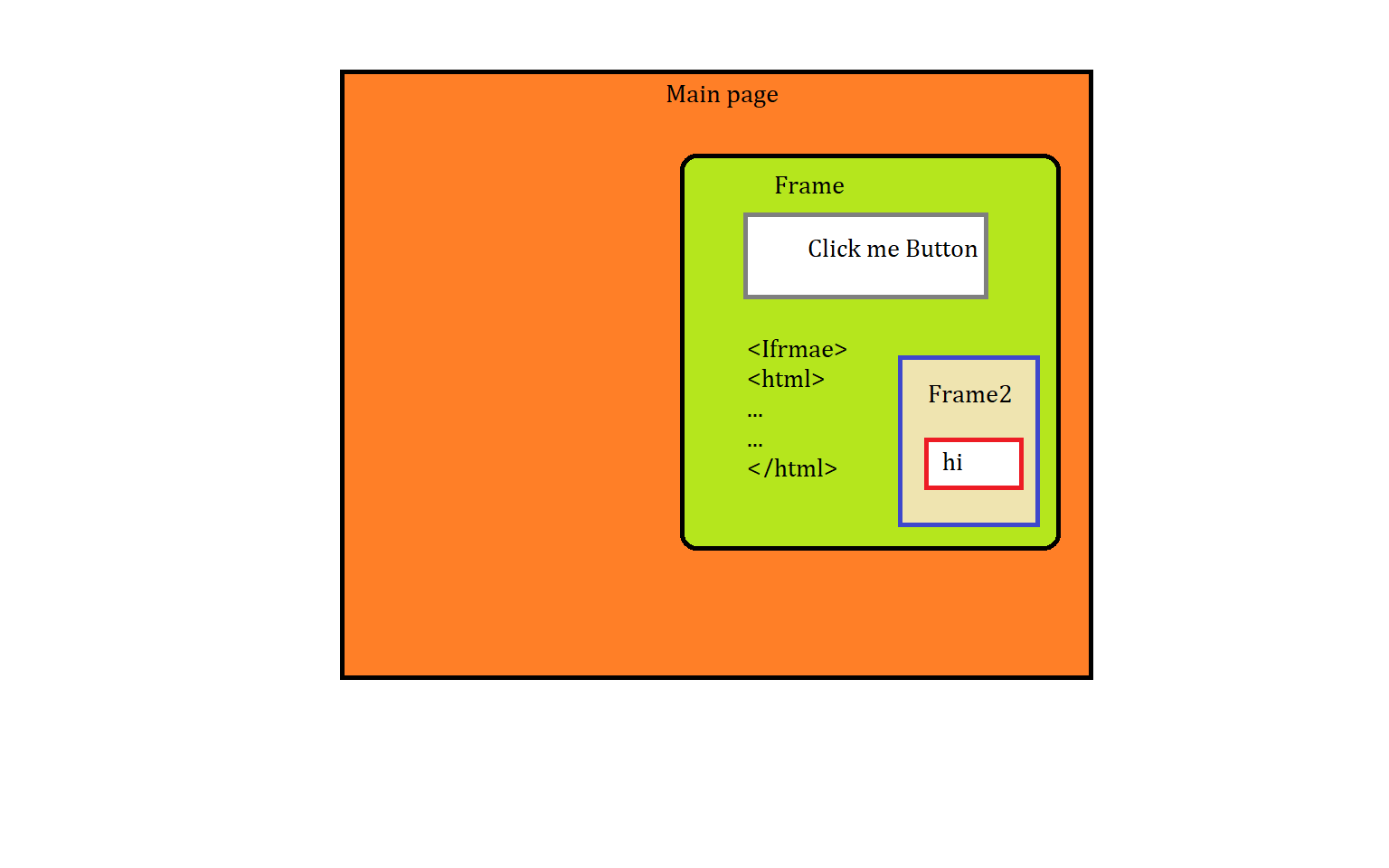
We can switch to frame by 3 ways- i.e. id or name, index, or Webelemt

Once action perform on elements present on iframe, selenium will not navigate to main page by default.

To navigate from iframe to main we need use method like parent or default content.

**Driver.switchTo.parentframe();** → Child frame to immediate parent

**Driver.switchTo.defaultcontent ();** → Child frame to main page



**Popups:**

Popups are small or separate window which will be displayed when we perform action on any components present in a webpage.

These popus can be handled by selenium directly, but sometimes we may need to use 3rd (Auto IT) party tools to handle these popups.

If we are able to inspect element present in a popup then we can use selenium directly to handle that popup.

If we are unable to inspect element present in a popup then we need to use 3rd party tool to handle that popup.

**Types of popups:**

\*1. Hidden-Division Popup

\*2. Alert popup

\*3. Child browser Popup

4. Authentication Popup

5. File Upload Popup

6. File Download Popup

**1. Hidden-Division Popup:**

1. These popups are colorful.
2. We can inspect the elements present in pop up.
3. As we can inspect element present in popup then using selenium we can handle it & no need to switch.

**2. Alert Popup:**

1. We cannot inspect the elements present in pop up, does not have any colors

2. These popups will contain **ok** button or **cancel** button & **Text**.

3. Sometimes this type of popup also contains”?” or “**!”** symbol.

Exceptions:

UnhandledAlertException🡪

If we not handled alert popup and tried to take next action.

NoAlertPresentException🡪

If there is no alert in flow and you trying to handle the alert

**Procedure to handle Alert popup**

1. To handle alert popup we need to switch selenium focus from main page to alert popup by using syntax

**Alert alt = driver.switchTo().alert();**

1. Alert is an interface which contains abstract methods like:

1. **accept():** use to click on ok button. alt.accept();

2. **dismiss():** use to click on cancel button. alt.dismiss();

3. **getText():** use to get text present in an alert popup.

String text = alt.getText();

**3. Child browser Popup/window popup**

1.We can inspect elements present in popup.

2.This popup will contain address field(url),maximize,minimize and close options.

**Procedure to handle Child browser popup**

1. To handle Child browser popup we need to switch selenium focus from main page to Child browser popup by using syntax

* driver.switchTo().window("String windowId")

1. To get ID of main page we need to use syntax

String IdOfMainPage=driver.getWindowHandle(); //return type of getWindowHandle method is String

1. To get address of main page as well child browser we need to use synatx

set<String> IdOfAllpages=driver.getWindowHandles();

//return type of getWindowHandles method is set<String>

1. To get address of child browser we need to use synatx

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

ArrayList al=new ArrayList(ids);

System.out.println(al.get(0)); // address of main page

System.out.println(al.get(1)); // address of child1 browse

org.openqa.selenium.NoSuchWindowException🡪 we have not switched to window and tried some actions

**4. Authentication Popup--autoIT tool / Robot class**

1. Pop up displayed on Page load
2. We cannot inspect the elements present in pop up.
3. We can move the pop up(Except chrome & edge)

example🡪

username: admin password: admin

**5. File upload🡪 AutoIT tool/ Robot class**

1. Find the upload button, store in reference variable
2. Reference variable .sendKeys(“filePath”)

**6. File download🡪 AutoIT tool/ Robot classs**

The various mouse actions that are provided by the **Actions** class are-

1. click () – clicks at the current location
2. doubleClick() – performs a **Double click** at the current mouse location
3. contextClick() – performs a **Right click** at the current mouse location
4. dragAndDrop(WebElement source, WebElement target) – drags an element from the source location and drops in target location
5. moveToElement(WebElement target) – moves to the target element

**Keyboard Actions**

The various keyboard actions that are provided by the Actions class are-

1. keyUp(WebElement target, java.lang.CharSequence key) – performs a key release after focusing on the target element
2. keyDown(WebElement target, java.lang.CharSequence key) – performs a key press after focusing on the target element
3. sendKeys(WebElement target, java.lang.CharSequence… keys) – types the value.

**Scrolling**

**A Scroll** is a JavaScript method. The JavaScriptExecutor provides an interface that enables QAs to run JavaScript methods from Selenium scripts. Hence, to scroll up or down with Selenium, a JavaScriptExecutor is a must.

**Scroll functions can be defined as follows :**

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("window.scrollBy(0,250)", "");

The **scrollBy()** method involves two parameters, x, and y, that represent the horizontal and vertical pixel values, respectively.

Setting Window Size:

* Use driver.manage().window().setSize(new Dimension(width, height)) to set the window size to a specific width and height in pixels.
* For example: driver.manage().window().setSize(new Dimension(1024, 768)).

Setting Window Position:

* Use driver.manage().window().setPosition(new Point(x, y)) to move the browser window to a specific position on the screen, where x and y are the coordinates.
* For example: driver.manage().window().setPosition(new Point(0, 0)).

**Chrome Options Class**

**What is Chrome Options Class?**

The Chromeoptions Class is a concept in Selenium WebDriver for manipulating various properties of the Chrome driver. The Chrome options class is generally used with Desired Capabilities for customizing Chrome driver sessions. It helps you perform various operations like opening Chrome in maximized mode, disable existing extensions, disable pop-ups, etc.

The ChromeOptions class in Selenium WebDriver allows the customization of ChromeDriver sessions by modifying browser properties.

**Below are the list of available and most commonly used arguments for ChromeOptions class**

**start-maximized**: Opens Chrome in maximize mode

**incognito**: Opens Chrome in incognito mode

**headless**: Opens Chrome in headless mode

**version**: Prints chrome browser version

**disable-extensions**: Disables existing extensions on Chrome browser

**disable-infobars:** Prevents Chrome from displaying the notification ‘Chrome is being controlled by automated software

**disable**-**popup-blocking:** Disables pop-ups displayed on Chrome browser

**make-default-browser:** Makes Chrome default browser

**Synchronization**

Synchronization/wait: **matching selenium test script speed with browser speed**

Selenium – Synchronization - matching selenium test script speed with browser speed.

**Wait-**

1.Static wait

2. Dynamic wait

**static wait: java wait**

**1. Thread.Sleep**

Thread.sleep(3000); // Sleep for 3 seconds

Thread.Sleep is a static wait

Wait time is fixed, irrespective of condition. Script will be hold for mentioned seconds.

**Dynamic wait: Selenium**

**2. Implicit Wait**

driver.manage().timeouts().implicitlyWait(Duration.*ofMillis*(6000));

Applicable: complete webpage // wait for full webpage loding

1 parameter: time value (seconds)

//2000ms🡪 100ms🡪1900 release wait time

Advantage of implicit wait is that suppose we have mension time 5000 ms and our page load in 1000 ms then it will save remaining 4000 ms for nest task.

**3. Explicit wait**

WebDriverWait w = new WebDriverWait(driver, Duration.*ofMillis*(15000));

w.until(ExpectedConditions.*visibilityOfElementLocated*(By.*xpath*("(//h6[text()='Sign In'])[2]")));

//w.until(ExpectedConditions.*elementToBeClickable*(By.*xpath*("(//h6[text()='Sign In'])[2]")));

//w.until(ExpectedConditions.*visibilityOf* (By.*xpath*("(//h6[text()='Sign In'])[2]")));

Applicable: single element

1 parameter: time value(seconds)

2 parameter: Condition (isselected, isdisplayed, isenabled)

An explicit wait Applicable for single element.

It will wait until specific element to be load.

**4. Fluent Wait**

FluentWait<WebDriver> w = new FluentWait<WebDriver>(driver).

withTimeout(Duration.ofMillis(15000)).

pollingEvery(Duration.ofMillis(1000)). ignoring(NoSuchElementException.class,TimeoutException.class);

w.until(ExpectedConditions.elementToBeClickable(By.xpath("(//h6[text()='Sign In'])[2]")));

driver.findElement(By.xpath("(//h6[text()='Sign In'])[2]")).click();

Applicable: single element

1 parameter: time value(seconds)//1000ms

2 parameter: Condition (isselected, isdisplayed, isenabled)

3 frequency: time(time in sec)- 5 m sec🡪10ms

Fluent Wait will sets the maximum duration for the Selenium WebDriver to wait until a specific web element visible on webpage. it will try to find the element again and again during the mensioned wait time.

Suppose we have 6000 ms for an element to be available on the web page, but it will check its avilable once in every 500 seconds.

More flexible than explicit wait,allowing setting polling frequency and maximum wait time for a condition.

It also specifies how often WebDriver will check for the condition before raising an 'ElementNotVisibleException'.

-------- ------------- --------- ---------- ----

**How does an explicit wait differ from an implicit wait in Selenium?**

Ans: *Explicit* waits allow the script to pause until a specific condition is met, offering more precise control. Implicit waits, on the other hand, instruct the WebDriver to wait for a certain amount of time when trying to locate elements, applied globally throughout the script.

------------ ---------------- ------------------ -------------

**Implicit Wait**

Implicit wait is used in cases where the WebDriver cannot locate an object immediately because of its unavailability. The WebDriver will wait for a specified implicit wait time and it will not try to find the element again during the specified time period.

Once the specified time limit is crossed, the webDriver will try to search the element once again for one last time. Upon success, it proceeds with the execution; upon failure, it throws exception.

It is a kind of global wait which means the wait is applicable for the entire driver. Hence, hardcoding this wait for longer time periods will hamper the execution time.

**What is the difference between Absolute and Relative XPath in Selenium?**

*Ans: Absolute XPath provides the full path from the HTML document's root, offering precision but less flexibility. Relative XPath navigates elements based on relationships, providing more adaptability.*

**When should I use Absolute XPath over Relative XPath?**

*Ans: Absolute XPath is suitable when precise element location is critical, but it's less flexible and prone to breakage. Use it sparingly when element positioning is stable.*

**What are the advantages of Relative XPath in Selenium?**

*Ans: Relative XPath offers flexibility, robustness against webpage structure changes, readability, and efficiency compared to Absolute XPath.*

**Selenium Framework**

**POM class contains -**  Variables, Constructor, Methods

**Test Class -**

POM Class does not contain main method so to run POM class we create Test Class which contains main method. We don’t run POM class we run Test Class.

**@FindBy**

It is a annotation. Given by selenium.

Work same as driver.find element method

It contain all locators present in driver.findElement method like id , class, name, tagname, css selector,linked text, partial link text, xpath etc.

**Why we make WebElement as private ?**

**Private WebElement gender**

Variables present in POM classes are usually private in nature and those variables are utilize in methods which are public

Here we are using Encapsulation in POM classes by making variables private and utilizing methods in POM classes. To hide our data from external usage.

**Page Factory**

It is a class contain initElements method

**POM class contain main method? If no then how we run classes?**

No. POM class doesnot contains main method. We are creating test class which contains different different methods. By using this methods we run.

**Why we use driver?**

To perform action on web elements we need driver. But here we are not creating WebDriver object so temporarily we create local driver.

**Why we use this?**

To access global variables present in class

**@FindBy**

* Used to locate a single web element using one specific locator strategy.

**@FindBys**

* Used to locate an element that meets **all** specified locator conditions (acts like a logical **AND** condition).

**What is Lazy Initialization?**

In programming, lazy initialization (or lazy loading) is a technique where the creation or loading of an object or resource is delayed until it is actually needed, rather than being done immediately when the application starts.

**How it Works in Selenium PageFactory:**

* When you use PageFactory with @FindBy annotations, you declare web elements as fields in your page object classes.
* However, the elements themselves are not immediately found and stored in memory when the page object is instantiated.
* Instead, they are initialized as "proxies" or placeholders.
* Only when you try to interact with or access a web element (e.g., by calling click(), sendKeys(), or getText()), the PageFactory will locate the element on the page and return it.

**TestNG**

TestNG is a **java unit framework** use for writing/designing of **Test classes**.

TestNgVersion - 7.10.1

Selenium version – 4.28

**1 Emailable Report:**

Report generation is very important when you are doing the Automation Testing as well as for Manual Testing. By looking at the result, you can easily identify how many test cases are passed, failed and skipped. By looking at the report, you will come to know what the status of the project is. Selenium web driver is used for automating the web-application, but it won't generate any reports. The TestNG will generate the default report.

Green🡪 You are matching with your timeline (daily 10 TC)

Yellow🡪 You are lagging in your execution (daily 8 TC) 🡪(blocking defect)

Red🡪 You are far behind your target (No TC executed)

Steps to generate Emailable report🡪

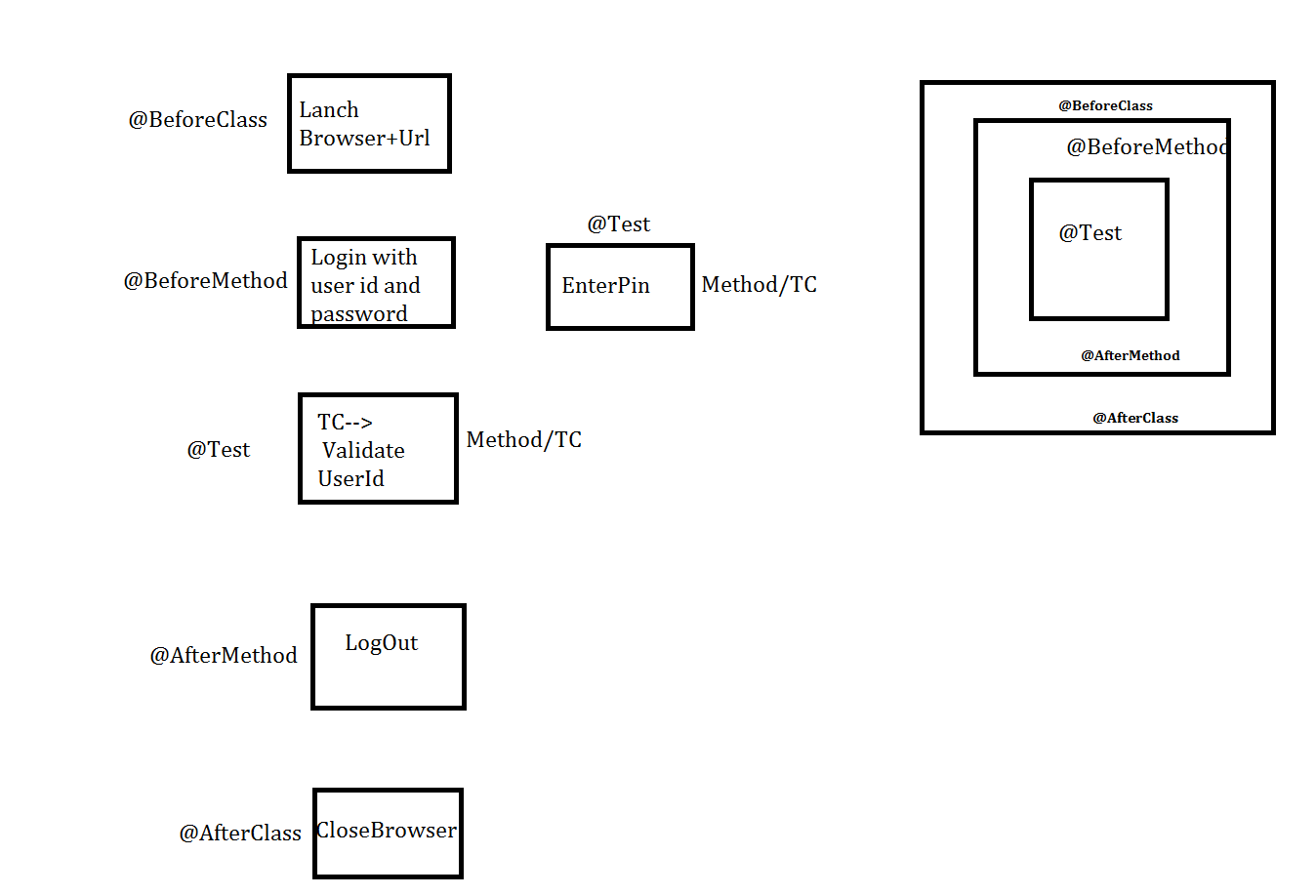
1. Execute Test class and refresh the project.
2. You will get test-output folder.
3. In That folder Right click on the "emailable-report.html" and select the option Open with the web browser or double click on it.

Note:

1. if we use sop() to display text as a output then result will be displayed in console only, not in emailable report.
2. To display text in emailable report we need to use static method log present in Reporter class. eg. Reporter.log("String msg", true)

**2. TestNG Annotation**

1. @Test: - Used for execution of test method/TC.
2. @BeforeMethod: - It is used for execution of method before execution of every test method with an annotation @Test.
3. @AfterMethod: - It is used for execution of method after execution of every test method with an annotation @Test.
4. @BeforeClass: - It is used for execution of method before execution of test class.
5. @AfterClass: - It is used for execution of method after execution of test class.



**3. TestNG Keyword:**

1. invocationCount

2. priority

3. enabled

4. TimeOut

5. dependsOnMethods

1. **invocationCount:**

Sometimes same test method/TC need to be executed multiple times which can be possible by using TestNG keyword "invocationCount" eg. invocationCount=5;

**InvocationCount cant be :**  0, Negative.

**Default InvocationCount** is 1.

1. **priority**:

If we multiple TC in class then It will execute alphabetically. to change test TC execution sequence we need to use TestNG keyword "priority". If P =def then executes alphabatically

**priority can be** : Positive, Negative, Duplicate bydefault=0

**priority can't be** : Decimal, Variables

|  |  |
| --- | --- |
| eg. priority=1 |  |

1. **enabled:**

Disabling a test method/TC in TestNG can be achieved by setting the enabled attribute of the @Test annotation to false. eg. enabled=false

1. **TimeOut**:

If test class contains multiple test methods if one of the test method is time consuming to execute then TestNG bydefault **fail** that TC & executes the other TC.

eg. @Test(timeOut=8000)

**5. dependsOnMethods**:

If 1 TC execution depends on multiple TC then we need to use "dependsOnMethods" TestNG keyword. e.g. (dependsOnMethods = {"c","b"})

**Maven**

**Q. Why we Surefire plugin?**

Ans – to use multiple xml file we need to add surefire plug in.