papergrid

Date: / /

⇒ Normal find Element approach with dynamic dropdown →

→ If select tag is not there in the UI in that case we can go with this approach.

→ If select tag is not there and dropdown is dynamic in that case we ~~have~~ have to go for screen debugger concept.

→ To make a screen pause you have to follow few steps :-

a) Right click on the web page.
b) click on sources and press function(fn + f8) or
c) only f8.

* Scenario 33 → Launch chrome Browser
→ Access flipkart
→ In the home page move to electronic and select electronics GST store.

```
WebDriver driver = new ChromeDriver();
driver.get(" flipkart url");
driver.manage().window().maximize();
WebElement category = driver.findElement(By.xpath("//
                span[text() = 'Electronics']"));

Actions act = new Actions(driver);
act.moveToElement(category).perform();
Thread.sleep(2000);
driver.findElement(By.xpath("//a[text()='Electronics
                GST store']")).click();
```

* Scenario 34 → Launch chrome Browser
→ Access flipkart
→ Move to Electronics, select audio
→ fetch all the audio list
→ After that click on Headphone Pouch & case covers.

```
WebDriver driver = new ChromeDriver();
driver.get("flipkart.url");
driver.manage().window().maximize();
WebElement category = driver.findElement(By.xpath
        ("//span[text() = 'Electronics']"));
Actions act = new Actions(driver);
act.moveToElement(category).perform();
Thread.sleep(2000);
WebElement audio = driver.findElement(By.xpath
        ("//a[text() = 'Audio']"));
act.moveToElement(audio).perform();
Thread.sleep(2000);
List<WebElement> allproducts = driver.findEle-
        ments(By.xpath("//a[@class='_3490ry
        ]"));
Thread.sleep(2000);
Iterator<WebElement> itr = allproducts.iterator();
Thread.sleep(2000);
while(itr.hasNext())
{
String values = itr.next().getText();
sop(values);
}
```

papergrid

Date: / /

```
Thread.sleep(2000);
driver.findElement(By.xpath("//a[text()=
        'Headphones pouch & Case covers']")).
                                    click();
```

* Scenario 35 →

⇒ With the help of keyBoard stroke →

→ If the dropdown is not developed with the help of select tag we can go with keyboard stroke.
→ Keyboard stroke can handle static as well as dynamic dropdown.
→ In this case we have to go with robot class.

* Scenario 35 → Lauch Chrome Browser
→ Access demoapp.qspider.com
→ Select dropdown
→ click on search with select
→ In the country dropdown select a country called as poland.

```
WebDriver driver= new ChromeDriver();
driver.get("demo app.qspider.com");
driver.manage().window().maximize();
Thread.sleep(2000);
driver.findElement(By.xpath("//section[text()=
        'Dropdown']")).click();
```

```
Thread.sleep(2000);
driver.findElement(By.xpath("//a [text()='Search
                with select']")).click();
Thread.sleep(2000);
driver.findElement(By.xpath("//div[@class='css-13cym-
                wt-control']) [3]")).
                click();
Thread.sleep(2000);

Robot r = new Robot();
r.keyPress(KeyEvent.VK_DOWN);
r.keyRelease(KeyEvent.VK_DOWN);

r.keyPress(KeyEvent.VK_DOWN);
r.keyRelease(KeyEvent.VK_DOWN);

r.keyPress(KeyEvent.VK_ENTER);
r.keyRelease(KeyEvent.VK_ENTER);
```

papergrid
Date: / /

V. Imp.

* Synchronisation →

→ Synchronisat^n is nothing but time mismatch issue in b/w automation tool speed and web browser speed.

→ Synchronisat^n issue is happened because automat^n tool speed is faster than web browser speed.

→ We are having 4 ways to handle synchronisation.
a) Thread. sleep.
b) Implicit wait
c) Explicit wait
d) fluent wait

⇒ a) Thread. sleep →

→ It is one of the way to handle synchronisat^n.
→ This support provides by JAVA.

⇒ Drawbacks of thread. sleep →

Thread. sleep is not a good approach to handle synchronisation because –
a) Unnecessarily code length will be increased.
b) If any element loaded before the given time, still it will wait for the given time. and because of this execution time will be increased.

Note → Because of this drawbacks we are going to use implicit wait.

papergrid

Date: / /

6) Implicit Wait →

→ It is one of the way to handle synchronisation.
→ This support provide by selenium.
→ Implicit wait works very smartly in the automatn script.
⇒ In this we have to declare implicit wait only for 1 time just after launching the browser.
⇒ In case of implicit wait we can provide the maximum time interval, which means if any element loaded before the max^m time it will not wait for the remaining time and it will perform the next operation.

Syntax → driver.manage().timeouts().implicitlyWait (Duration.of Seconds(10));

Structure →

```
ophons opt = driver.manage();
Timeouts to = opt.time Outs();
to.implicitlyWait(Duration.of Seconds (10));
```

Note → Implicit wait we can declare multiple times in a script but it is not a recommended approach.

→ In a script some times implicit wait might fails in that case we have to provide thread.sleep to analyze really synchronisatn issue is there or not, if it is there it will be solved by thread.sleep and after that we will remove Thread.sleep and provide explicit Wait.

c) Explicit Wait →

→ It is one of the way to handle synchronisation.
→ This support provides by selenium.
→ In a automatn script whenever implicit wait fails we should go for explicit wait that's y we can use explicit wait multiple times in a script.

Syntax → WebDriverWait = new WebDriverWait (driver, Duration.ofSeconds(10));
wait.until(ExpectedConditions.particularyCondition());

⇒ Poll time →
→ In case of explicit wait a default time monitor interval in explici is present which is called as poll time.
→ The default seconds are 0.5sec.

d) fluent Wait →

fluent wait is exactly same as explicit wait, in case of fluent wait we can customize the time interval in more unique manner.

Date: / /

* Difference B/w Implicit Wait & Explicit Wait

Implicit Wait	Explicit Wait
→ Implicit wait we should declare only one time in a script just after launching the browser.	Explicit wait we should declare multiple times whenever implicit wait fails.
→ Implicit wait works only with findElement and findElements.	Explicit wait works with all the methods.
→ Incase of implicit wait no need to create any object.	Incase of explicit wait we need to create an object of WebDriverWait class.
→ Syntax → driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));	Syntax → WebDriverWait wait = new WebDriverWait(driver, Durat^n.ofSeconds(10)); wait.until(ExpectedConditions.particularCondition());

28/2/24

* Alert Pop-up

→ Alert pop-up is completely developed by Java script. that's y it is also called as Java script pop-up.

→ Alert pop-ups ~~is~~ are also called as ~~Jo~~ confirmation popup.

→ Whenever alert popup is visible entire webpage will be disabled.

→ Alert popups are not inspectable.

→ To handle the alert popup we have to switch the webdriver focus to the alert popup.

Syntax → driver.switchTo().alert();

→ The return type of the alert() is Alert(Interface).

→ Inside Alert Interface few methods are there to perform multiple operations in alert pop-up.

a) accept() → This method helps us to click on 'Ok' button present in a alert popup.

Syntax → alt.accept();

b) dismiss() → This method helps us to cancel a alert popup.

Syntax → alt.dismiss();

c) getText() → This method helps us to fetch the alert message.

Syntax → alt.getText();

d) Sendkeys() → This method helps us to send a message inside a alert popup.

Syntax → alt.sendkeys();

* Scenario 36 → Access demoapps
→ Collapse Web Element
→ click on Javascript
→ click on alertbox
→ fetch the alert message and click on ok.

```
WebDriver driver = new ChromeDriver();
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds
                                        (10));
driver.get("demoapps.qspider.com");
driver.manage().window().maximize();
driver.findElement(By.xpath("//section[text()='Web
                                Elements']")).click();
driver.findElement(By.xpath("//section[text()='popups'
                                ]")).click();
driver.findElement(By.xpath("//section[text()='Java
                                Script']")).click();
driver.findElement(By.xpath("//button[text()='Alert
                                Box']")).click();
Alert alt = driver.switchTo().alert();
String alertMsg = alt.getText();
sop(alertMsg);
Thread.sleep(3000);
alt.accept();
```

papergrid

Date: / /

* Scenario 37 → Access demoapps
→ Collapse webelements
→ click on popups
→ click on javascript
→ click on confirm
→ click on confirm box
→ fetch the alert message
→ click on ok
→ Again click on confirm box
→ click on cancel.

```
WebDriver driver = new ChromeDriver();
driver.manage().timeouts().implicitlyWait(Duration.of
                                    seconds(10));
~~driver.manage().timeouts()~~
driver.get("demoapps");
driver.manage().window().maximize();
driver.findElement(By.xpath("//section[text()='Web
                                    Element']")).click();
driver.findElement(By.xpath("//section[text()='popups'
                                    ]")).click();
driver.findElement(By.xpath("//section[text()='JavaScript']
                                    ")).click();
driver.findElement(By.xpath("//a[text()='confirm']")).
                                    click();
driver.findElement(By.xpath("//button[text()='confirm
                                    Box']")).click();
Alert alt = driver.switchTo().alert();
String alertMsg = alt.getText();
Sop(alertMsg);
```

29/1/24

Note

```
alt.accept();
Thread.sleep(3000);
driver.findElement(By.xpath("//button[text()='
                                  confirm Box']")).click();
alt.dismiss();
```

29/1/24 |20|24|

* Main frame →

→ frames are also called as 'iframe'.

→ whenever we are dividing a webpage into multiple parts each and every part is consider as 'frame'.

→ To access this frames we have to switch the webdriver focus to that particular frame to perform operations.

Syntax → driver.switchTo().frame(index no)

Note → two methods are there to access to the child frames.

a) frame(index)

b) frame(String id value) { if id is there or name }

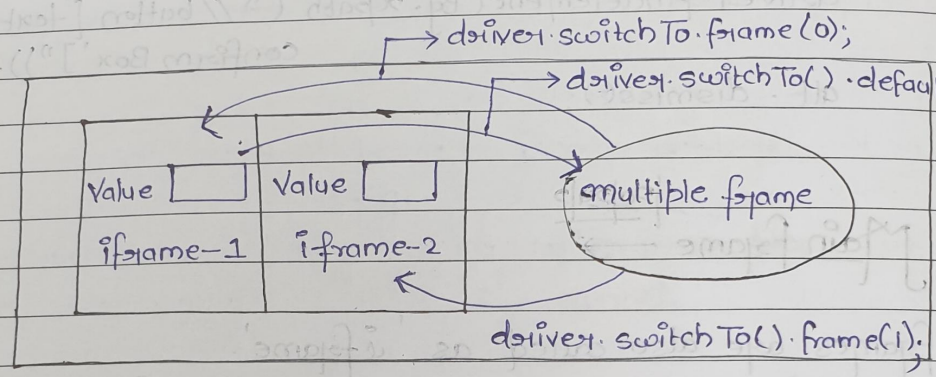
→ One subframe to another subframe we cannot switch directly, in this case we have to switch back to the mainframe or immediate parent frame.

Syntax → driver.switchTo().defaultContent();

or

driver.switchTo().parentFrame();

→ from mainframe or parent frame again we have to switch to the childframe.



* Scenario 38 → Launch Chrome Browser
→ Access demoapps
→ Collapse webElements
→ click on frames
→ click on iframes
→ click on multiple iframe
→ fill the credentials in signup
→ fill the credentials in login.

```
WebDriver driver = new ChromeDriver();
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds
                                        (10));
driver.get("demoapps");
driver.manage().window().maximize();
driver.findElement(By.xpath("//section[text()='WebElements
                                        ]")).click();
driver.findElement(By.xpath("//section[text()='frames']"))
                                        click();
```

papergrid

Date: / /

```
driver.findElement(By.xpath("//section[text()='iframes']")).click();
driver.findElement(By.xpath("//a[text()='Multiple iframe']")).click();
Thread.sleep(3000);
driver.findElement(By.xpath("//input[@name='email']")).sendKeys("Admin@gmail.com");
driver.findElement(By.xpath("//input[@name='password']")).sendKeys("Admin1234");
driver.findElement(By.xpath("//input[@name='confirm password']")).
        sendKeys("Admin@1234");
Thread.sleep(3000);
driver.findElement(By.xpath("//button[text()='signup']")).click();
Thread.sleep(3000);
driver.switchTo().defaultContent();
Thread.sleep(3000);
driver.switchTo().frame(1);
driver.findElement(By.xpath("//input[@name='username']")).
        sendKeys("SuperAdmin@gmail.com");
driver.findElement(By.xpath("//input[@name='password']")).sendKeys
        ("SuperAdmin@1234");
driver.findElement(By.xpath("//button[text()='Login']")).click();
```

PLP → Product list page
PDP → Product discription page

papergrid
Date: / /

* Scenario 39 → Launch chrome Browser
→ Access flipkart
→ Search for iphone 11
→ click on the 1st product from the product list page (PLP).
→ click on go to cart (inside PDP page)
→ click on add to cart inside cart page.

```
WebDriver driver = new ChromeDriver();
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

driver.get("flipkart Url");
driver.manage().window().maximize();
WebElement element = driver.findElement(By.xpath("//input[@name='q']"));
element.sendKeys("iphone 11");
element.submit();
driver.findElement(By.xpath("//div[@class='_4rROIT']")[1].click();

Thread.sleep(3000);
String parentId = driver.getWindowHandle();
String title = driver.getTitle();
String parentWindowTitle = parentId + title;
Set<String> allIds = driver.getWindowHandles();
Thread.sleep(3000);

// Standard way → comparison with title and id

Iterator<String> itr = allIds.iterator();
While(itr.hasNext())
{
String all = itr.next();
```

```
String childWindowTitles = driver.switchTo().window(all).
                                getTitle();
 if (childWindowTitles.contains(parentWindowTitle))
 {
  break;
 }
}
// Basic way comparison based on id for( String all: all
 for (String all : allIds)
  {
    if (!parentId.equals(all))
      {
        driver.switchTo().window(all);
      }
  }
 //
Thread.sleep(3000);
driver.findElement(By.xpath("//button[text()='Add to
                                Cart']")).click();
Thread.sleep(3000);
driver.findElement(By.xpath("//span[text()='Add Item'
                                ]")).click();
  }
}
```

* Window pop-up Handling →

To handle window popup we have to use 2 methods -
1) getWindowHandle();
2) getWindowHandles();

1) getWindowHandle() →

getWindowHandle() will help us to fetch the parent window id.

2) getWindowHandles() →
getWindowHandles() will helps us to fetch the all window IDs.

→ The return type of this method is set<String>.

* Difference b/w getWindowHandle() and getWindowHandles()

getWindowHandle()	getWindowHandles()
→ This method will fetch the parent Window ID.	This method will fetch the all the windows ID.
→ Return type of this method is string.	Return type of this method is Set<String>.

papergrid

Date: / /

* Calender pop up →

→ We have to use 2 method called as getInstance() and getTime() to fetch the current date, time and location information in the default format.

```
Calendar cal = Calendar.getInstance();
Date d = cal.getTime();
```

→ To get the date in the required buisness format we have to create an instance of simple date format class. and we have to use a method called as 'format method'.

```
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
String currentDate = sdf.format(d);
```

Note → If we want to fetch or select future dates or past dates we have to use a method called as "add()."

```
cal.add(Calendar.DAY_OF_WEEK, 7); → to fetch 1 week later date
```

* Scenario 40 → Access demoapps.

→ Collapse Web Elements

→ Select date and time picker

→ Select date picker

→ provide the current date in the calendar.

```
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
```

papergrid

Date: / /

```
public class CalendayHandling {
public static void main (String [] args) throwos Exception
 {
 Calenday cal = Calenday.getInstance();
// cal.add (Calenday.DAY_OF_WEEK,7);
 Date d = cal.getTime();
 System.out.println (d);
 SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/
                                        YYYY");
 String currentDate = sdf.format(d);
 System.out.println (currentDate);

 WebDriver driver = new ChromeDriver();
 driver.get("https://demoapps.qspiders.com/ui?
                    scenario=1");
 driver.manage().window().maximize();
 Thread.sleep(3000);
 driver.findElement (By.xpath("//section[.='Web
                    Elements']")).click();
 Thread.sleep (3000);
 driver.findElement (By.xpath("//section[.='Date
                    and time picker']")).click();
 Thread.sleep (3000);
 driver.findElement (By.xpath("//section[.='Date
                    Picker']")).click();
 Thread.sleep(3000);
 driver.findElement (By.xpath ("//input[@place
                    holder=\"Select A Date\"]")).
                    sendKeys (currentDate);
 }
}
```

4/3/24

* Fram

Framew
to wor
for
we

* Types

a) Keyw

becau
so p
can

→ The
a) Vers
b) It is

Note → Th
bec

b) Mo

we
ca
eg
→ "
c) Hy
f
eg

papergrid

Date: / /

```
public class CalendarHandling {
public static void main (String [] args) throws Exception
{
Calendar cal = Calendar.getInstance();
// cal.add (Calendar.DAY_OF_WEEK, 7);
Date d = cal.getTime();
System.out.println (d);
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/
                                              YYYY");
String currentDate = sdf.format(d);
System.out.println (currentDate);

WebDriver driver = new ChromeDriver();
driver.get("https://demoapps.qspiders.com/ui?
                    scenario=1");
driver.manage().window().maximize();
Thread.sleep(3000);
driver.findElement (By.xpath("//section[.='Web
                    Elements']")).click();
Thread.sleep (3000);
driver.findElement (By.xpath("//section[.='Date
                    and time picker']")).click();
Thread.sleep (3000);
driver.findElement (By.xpath("//section[.='Date
                    Picker']")).click();
Thread.sleep (3000);
driver.findElement (By.xpath ("//input[@place
                    holder=\"Select A Date\"]")).
                    sendKeys (currentDate);
}
}
```

4/3/24

*

*

a)

→

a)

b)

Note →

b)

→

c)

SECTION - 3

4/3/24

papergrid

Date: / /

* Frame Work →

Framework is an empty structure in which we are going to write our codes in an organised manner for better understandability and standardization.
we can give example of class room.

* Types of framework → (means different - different ways to organise manner)

a) Keyword driven framework → This type of framework are very easy to use because each and every functionality based on a keyword so person who don't have technological knowledge can easily use this framework.

→ The drawback of this framework is
a) Very difficult to develop.
b) It is too much time consuming.

Note → That's y now a days no company is using this framework because maintainence is very difficult.

b) Modular driven development framework → Whenever in our application we need to deal with huge no. of modules, in that case we have to go with this particular approach.
eg → Ecommerce applications, Banking applications, Salesforce application
→ It is very easy to use or to develope also.

c) Hybrid framework → Hybrid framework is nothing but combination of multiple frameworks.
eg → DDD + TDD (Test driven development means TestNG)
↓
Data driven development.

if maintainence is high → that means cost high resources
low → less resources

papergrid
Date: / /

* Data driven development framework → Whenever we have to deal with huge number of data in our application. We have to go for data driven development framework.

⟹ Advantages →
- → Easy to develop.
- → Very much user friendly
- → Data seggregation will be easy.
- → It consumes very much less memory.
- → Data will be secured.

* Types of Data in a project →

data

In any application we have

common data (that is also test data)
- ↳ UN
- ↳ PW
- ↳ URL

Storing place → properties file
- ↳ Very less memory
- → Store value in key & value (KV) pair
- → data will be secured

Test data module
- ↳ for Individual Test case data
- ↳ Excel
  - ↳ maintainence of data will be easy.
  - ↳ Excel works very fast
  - ↳ data will be secured

This common data, we ~~will~~ can store in a excel also, some company follow this rule. Only they create one more page in excel.

→ In real time we are having two types of test data -
a) Common test data
b) Individual test case data

a) Common test data → This type of data is common for entire testing team.

eg → Username, Password, URL

→ This type of data we are going to store in properties file.

⇒ Advantages of properties files -

→ It will consume very less memory (It is very much light weighted.)
→ We can store the value in KV pair.
→ Data will be secured.

b) Individual test case data → This type of data will be used by different different test engineers.

→ This type of data we are going to store in excel.

⇒ Advantages of Excel -

→ Data seggregation will be easy which means maintainence will be easy.
→ Data will be secured.
→ Excel works faster.

papergrid
Date: / /

* MAVEN PROJECT CREATION FOR HYBRID FRAMEWORK →

→ We have to create a maven project by following few standards.

a) Create a simple project and click on next.

b) provide group id and artifact id.

groupid → Company name

Artifact id → org/com . Ecommerce . falcon
(Project details)

type — domain name — Project name

→ Once we create a maven project 4 folders have to be displayed.

→ If it is not displayed right click on pom.xml → Maven → Update project → force update

→ pom.xml is the most important part of any maven project.

→ Inside this pom.xml we are going to add all the dependencies.

Note → pom stands for project object model

papergrid

Date: / /

* Store the common data in properties file and use the data in script →

→ To perform this operation, we have to follow few steps -

a) Create a properties file (.properties extension)

b) Create an object of FileInputStream class and provide the path of properties file.

c) Create an object of properties class and load the data with the help of load method. load().

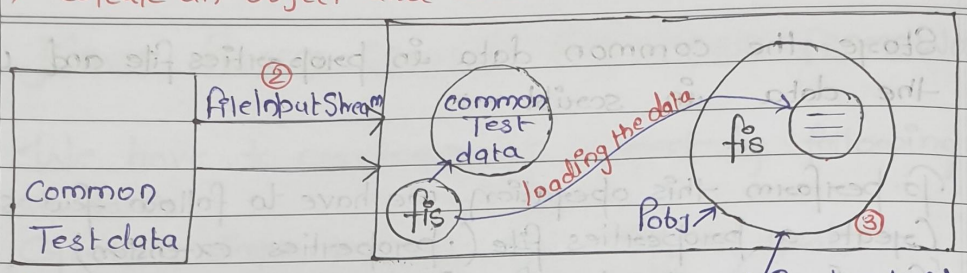
d) Use getProperty() to fetch the encapsulated data.

Note → The key format which we are using in properties file exactly same format we have to use inside getProperty().

```
Package genericUtility;
import java.io.FileInputStream;
import java.util.Properties;
class PropertiesUtility {

public void main(String[] Jargs) throws Exception {
  FileInputStream fis = new FileInputStream
        ("C:\\SeleniumFolder\\OrangeHrmCredentials.
          properties");
  Properties pobj = new Properties();
  pobj.load(fis);
String USERNAME = pobj.getProperty("username");
String PASSWORD = pobj.getProperty("password");
String URL      = pobj.getProperty("url");
  SOP(USERNAME);  // o/p → Admin
  SOP(PASSWORD);  // → admin123
  SOP(URL);       // → url we get
}
}
```

1) Store the data in properties file
2) Provide some path so by which the data come into your stream by FileInputStream.
3) Create an object and load the data inside the object



① properties file

4) Here data is secured inside the object, so how we will fetch the use by the secure data by using getters method, so by getproperty().

```
/**
* This class is for properties utility
* @ author Mamta Choudhary
*/

public class PropertiesUtility {

/**
* This method is a generic method to fetch data from
  properties file
* @ param data
* @ return
* @ throws Exception
*/
public String getValueFromProperties (String data) throws
                                                Exception
{
    FileInputStream fis = new FileInputStream (IconstantUtility.propertiesPath);
    properties pobj = new Properties ();
    pobj.load(fis);
    String value = pobj.getProperty(data);
    return value;
}
```

in class level also

in real time we have to give for standard in method level

argument

method with return type with argument

By doing this my code we will optimize

→ for login, we will not use the logic, only we call the method in our testscripts.

→ If multiple servers are there to check compatibility testing with the diff different data that time same method they can also use.

papergrid

Date

⇒ for Interface →

→ ~~How path will be same for all?~~

```
Package genericUtility;

/**
* this is an interface to store all the constant paths
* @Mamta Choudhary
* @Version 3.5.24
*/

public interface IconstantUtility {

Sting propertiesPath = "C:\\Seleniumfolder\\Orange

("C:\\\\Users\\\mamta\\\OneDrive\\\Desktop\\\Selenium\\\
  OrangeHrmCredentials.properties");
```

* Steps to add dependency →

→ Go to maven repository [search in google]
→ Click on the 1st link
→ Search for the required dependency
→ Select the dependency
→ Select the most stable versions
→ Copy the dependency and paste it into pom.xml
→ Whenever we add any dependencies we have to update the project.
→ Whatever dependency we are going to add we have to add in between dependencies tag.

Date: / /

```
<dependencies>
    <dependency 1> </dependency 1>
    <dependency 2> </ —"— 2>
    <  —"— 3> </ —"— 3>
    <  —"— 4> </ —"— 4>
</dependencies>
```

* Scenario 41 → fetch the data from properties file
  → Use the data to perform login operation of OrangeHrm

```
Public class LoginAttimeByUsingProperties Data {

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

/* fetch data from properties Utility */
```

already we've created class of PropertiesUtility for using the member of this class we've to create object.

```
PropertiesUtility PUTIL = new PropertiesUtility ();
```

calling method which has given in PropertiesUtility class.

```
String USERNAME = PUTIL.getValueFromProperties ("username");
String PASSWORD = PUTIL.getValueFromProperties ("password")
String URL      = PUTIL.getValueFromProperties ("url");
```

these thing already in PropertiesUtility only we've to use it

```
/* launch the browser */

WebDriver driver = new ChromeDriver ();
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds
                                          (10));

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

If this scenario is working that means the data which I've stored in properties Utility. I am able to access and I can able to use.

/* login to the application by using the properties file data */

```
driver.findElement(By.xpath("//input[@name='username']"))
    .sendKeys(USERNAME, Keys.TAB, PASSWORD, Keys.ENTER);
```

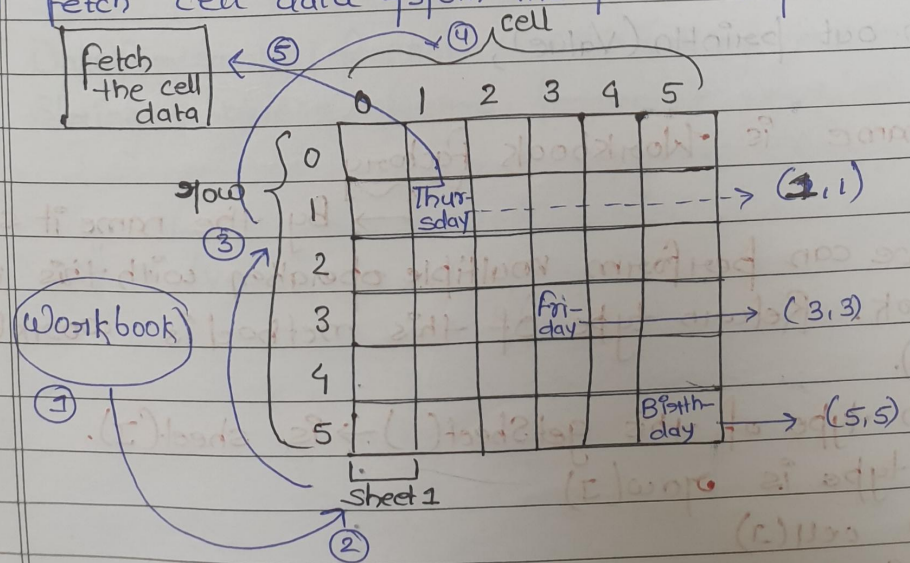
* Fetch Data from Excel →

→ To perform this operation we have ~~two~~ ~~ap~~ to add two dependencies -

a) apache Poi Common (universal apache) (sometime it wont work)
b) apache Poi ooxml Schemas (every excel work)

→ Steps to access and fetch data →

1) Access to the workbook
2) Access to the worksheet
3) Access to the row
4) Access to the cell
5) Fetch cell data from the particular cell



```
Package samplepackage

import java.io.FileInputStream;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.usermodel.WorkbookFactory;

public class fetchDataFromExcel {

public static void main(String[] args) throws Exception{
 FileInputStream fis = new
                FileInputStream("C:\\seleniumfolder\\OrangeHrm
                    TestData.xlsx");
 Workbook book = WorkbookFactory.create(fis);
 Sheet sh = book.getSheet("Sheet1");
 Row r = sh.getRow(6);
 Cell c = r.getCell(3);
 String value = c.getStringCellValue();

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

→ class name is WorkbookFactory
→ By the name it suggests that we can perform multiple operation with this multiple workbook. Return type of this method is workbook (Interface).

→ which sheet we want access

→ Return type of this getSheet( ) → is sheet(I).

→ Return type is row(I)

for cell → cell(I)

→ getStringCellValue → Return type is String

papergrid

Date: / /

* Special Case → In real time when we are fetching the data from excel sometimes numbers or other types of data might be present. In that case we will convert the data type into a string to avoid "Illegal State Exception".

→ To solve this issue we have to create an object of ~~data~~ DataFormatter class.

```
import org.apache.poi.ss.usermodel.DataFormatter;

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

    FileInputStream fis = new FileInputStream("C:\\Selenium
                                Folder\\OrangeHrmTestData.xlsx");
    Workbook book = WorkbookFactory.create(fis);
    Sheet sh = book.getSheet("Sheet1");

    // String value = sh.getRow(2).getCell(4).getStringCell
                                Value();  → method chaining

    DataFormatter format = new DataFormatter();
    String value = format.formatCellValue(sh.getRow(2).
                                getCell(4));

    Sopln(value);
```

papergrid

Date: / /

* Generic method to fetch data from excel →

```
/** this class consist all the reading & writing data in a worksheet
Public class ExcelUtility

/** this method is to fetch individual test data from an excel sheet
 * @ param sheetName
 * @ param rowNum
 * @ param cellNum
 * @ throws return
 * @ throws Exception
 */

public String getDataFromExcel (String sheetName, int rowNum, int cellNum)
                                                    throws Exception

FileInputStream fis = new FileInputStream (IconstantUtility.excel path);
Workbook book = WorkbookFactory.create (fis);
Sheet sh = book.getSheet (sheetName);
DataFormatter format = new DataFormatter ();
String value = format.formatCellValue (sh.getRow (rowNum).getCell (cellNum));
return value;
```

papergrid
Date: / /

* Write Data in excel →

→ To write data in an excel we have to create an object of "FileOutputStream class."

→ We have to use a method called as write().

Note→ Whenever we are writing any data in an excel, excel should be in closed condition otherwise we will get "FileNotFoundException".

→ We have to use 2 method is here
1) One ? Create row();
2) Create Cell();

→ We have to use setCellValue() to provide the required data.

```
import Java.io.FileOutputStream;

Public class WriteDataInExcel
{
public static void main(String[] args) throws Exception
{
  FileInputStream fis = new FileInputStream
        ("C:\\Selenium folder\\OrangeHrmTestData.xlsx");

WorkBook book = Workbook factory. create(fis);
sheet sh = book.getSheet("sheet1");
        sh.CreateRow(10).CreateCell(0).setCellValue
                        ("Friday");

FileOutput Stream fos = new FileOutput Stream
  ("C:\\ selenium folder \\ OrangeHrmTestData.xlsx");
```

```
    book.write(fos);
  }
}
```

* Generic method to write data in excel sheet.

```
/**
 * this method is to write data in excel sheet
 * @param sheetName
 * @param rowNum
 * @param cellNum
 * @param newValue
 * @throws Exception
 */
```

why it is void here because in this method we don't want any string value.

```
public void setDataInExcel (String sheetName, int rowNum,
        int cellNum, String newValue) throws Exception
{
  FileInputStream fis = new FileInputStream (IConstantUtility.
                                excelPath);
  Workbook book = WorkbookFactory.create (fis);
  Sheet sh = book.getSheet (sheetName);
  sh.createRow(rowNum).createCell (cellNum).setCellValue(newValue);
                                  ——> this is method chaining.
  FileOutputStream fos = new FileOutputStream (IConstant
                                Utility.excelPath);
  book.write (fos);
}
}
```

{ Data formatter is not required when we read the data on that time it is required not for writing the data }

* Reac

public

public

Fil

w

S

D

because row is starting from 1

}

papergrid

Date: / /

* Read Multiple data from Excel →

```
public class ReadMultipleData {

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

    FileInputStream fis = new FileInputStream("C:\\selenium
                          folder\\OrangeHrmTestData.xlsx");
    Workbook book = Workbook factory.create(fis);
    Sheet sh      = book.getSheet("Sheet 2");
    DataFormatter format = new DataFormatter();
    int lastRowNum = sh.getLastRowNum();
    for (int i=1; i<=lastRowNum; i++)
    {
        Row r = sh.getRow(i);
        return type is row (Interface).
        for (int j=1; j<=r.getlastCellNum-1; j++)
        {
        String data = format.formatCellValue(r.getCell(j));
          System.out.println(data);
        }
    }
  }
}
```

because row is starting from 1

Chaining.

papergrid

Date: / /

* Generic method for (fetching) reading multiple data from excel.

```
/**
* this is a generic method for fetch multiple ~~multipl~~
  data from excel
* @ param sheetName
* @ param startingRow
* @ param startingCell
* @ return
* @ throws Exception
*/

public ~~String~~ void readMultipleDataFromExcel (String sheetName,
                int startingRow, int startingCell) throws
                Exception {

  ~~String data = null~~;
  FileInputStream fis = new FileInputStream (IConstantUtility.
                excelPath);
  Workbook book = WorkbookFactory.create (fis);
    sheet sh = book.getSheet (sheetName);
    DataFormatter format = new DataFormatter ();
    int lastRowNum = sh.getLastRowNum();
    for (int i = startingRow ; i <= lastRowNum ; i++)
      {
        Row r = sh.getRow(i)

    for (int j = startingCell ; j <= r.getLastCellNum()-1; j++
      {
        String data = format.formatCellValue (r.getCell(j));
        System.out.println( data);
      }
  }
}
```

Note:-

* Scenario 42 → Launch chromeBrowser
→ Access OrangeHrm
→ Do the login operation
→ click on admin module
→ provide username and employee name
→ click on search button
→ perform logout operation.

Note:- Read the common test data from properties file, read the individual test data from excel.

```
public class AD001 {

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

/* Read data from properties */

PropertiesUtility PUTIL = new PropertiesUtility();
String USERNAME = PUTIL.getValueFromProperties("Username");
String PASSWORD = PUTIL.getValueFromProperties("Password");
String URL = PUTIL.getValueFromProperties("Url");

/*Read data from excel */

ExcelUtility EUTIL = new ExcelUtility();
String TcUsername = EUTIL.getDataFromExcel("Admin", 1, 1);
String EMPNAME = EUTIL.getDataFromExcel("Admin", 1, 2);
```

papergrid

Date: / /

```
/* launching the browser */

WebDriver driver = new ChromeDriver();
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds
                                        (10));

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

/* login to application */

driver.findElement(By.xpath("//input[@name='username'
                                        ")).
                    sendKeys USERNAME, Keys.TAB,
                    PASSWORD, Keys.ENTER);

/* main test operation */

driver.findElement(By.xpath("//span[.='Admin']")).click();
Thread.sleep(4000);
driver.findElement(By.xpath("//input[contains(@class,'oxd')]
                                        [2]")).
                    sendKeys(TCUsername);
driver.findElement(By.xpath("//input[@placeholder=
                    'Type for hints...']")).
                    sendKeys(EMPNAME).
driver.findElement(By.xpath("//button[.='search']")).click()

/* logout operation */

driver.findElement(By.xpath("//p[@class='oxd-userd-
                    dropdown-name']")).click();
```

```
/* launching the browser */

WebDriver driver = new ChromeDriver();
driver.manage().timeouts().implicitlyWait(Duration.ofSecond
                                        (10));

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

/* login to application */

driver.findElement(By.xpath("//input[@name='username'
                                        ")).
                    sendKeys USERNAME, Keys.TAB,
                        PASSWORD, Keys.ENTER);

/* main test operation */

driver.findElement(By.xpath("//span[.='Admin']")).click();
Thread.Sleep(4000);
driver.findElement(By.xpath("//input[contains(@class,'oxd')]
                                        [2]")).
                    sendKeys(TCUsername);
driver.findElement(By.xpath("//input[@placeholder=
                                        'Type for hints...']")).
                    sendKeys(EMPNAME).
driver.findElement(By.xpath("//button[.='search']")).click()

/* logout operation */

driver.findElement(By.xpath("//p[@class='oxd-userd-
                                        dropdown-name']")).click();
```

```
driver.findElement(By.xpath("//a[.='Logout']")).click();

/* close browser */
driver.quit();
```

* How to add dependency in pom.xml →

→ Google → Search 'maven repository' → click on 1st link → Maven repository: Search/Browse/Explore → Search for which dependency you want → choose the 1st one → which mostly used version u want to click on that → then copy → dependency and paste into the pom.xml → update your project.

→ Developers also used TestNG for unit testing tool.
→ @Test annotatn acts as a main method.

papergrid
Date: / /

11/3/24

* Test Engine → NG

→ Test NG stands for test next Generation.
→ It is only an unique testing tool used by automation test engineers as well as developers also.
→ In case of test NG we don't use main method, instead of that we use @Test annotation.
→ Test NG supports special execution like batch executn, ~~loop~~ or group executn, parallel executn and cross browser executn.
→ Test NG provides ~~readymade~~ ~~predimate~~ reports. by default 1 report will be generated) (while executing automatically
→ Test NG supports listeners implimentation.
→ Test NG supports assertions also.

* @Test declaration Position (place) in a Test case →

```
import org.testing.annotations.Tests;

public class Program {
    @Test
    public void createUser()
    {
      sop("user created successfully");
    }
    @Test
    public void modifyUser()
    {
      sop("user modified successfully");
    }
```

→ In group executn → we divide TC scripts into multiple groups. So which group is required to execute only that particular script will execute.

→ cross browser executn → Here same script will be executed in different browsers like compatibility testing.

```
@Test
public void deleteUser()
{
    sop("user deleted successfully");
}
```

→ for this selenium and there bcoz selenium web driver tool is not the proper tool for compatibility testing.

Conclusion →

→ @Test is mandatory for all the test cases otherwise it will not consider as test case.

→ We have to provide @Test in method level only for expected output.

→ We can declare @Test in class level but it not a recommended choice because some generic information or function will be there in the test case, which will consider as a test case execution part and we will not get the exact output.

→ TestNG support listeners implementatn for capturing screen-shot.

1) → Batch executn → suppose we've written 10 TC scripts. We executed all of this in single script level (to check it is running or not)

→ We merge all TC scripts into [suit] → then we click on run button and 1 by 1 it will execute.

2) Parallel executn → Suppose we write 5 TC scripts at a time 5 scripts will be running in different-different browsers

max^m time	Batch executn	Parallel executn
1TC → 2min	4TC – 1min 15sec	But in parallel executn → all 5 TC scripts max to max it will take 2 min time bcoz all TC scripts will be run at a time. Here we can save the time
2TC → 10sec	5TC – 50sec.	
3TC → 1min	total 5 mins 15sec	

* How to add TestNG or install →
Help → Eclipse Marketplace → search TestNG → Enter → Install → Agree all Trust → Finish

papergrid
Date: / /

* priority based Test Cases →

```
public class Program 1 {

@Test(priority=1)
  public void createUser()
  {
   sop("user created successfully");
  }

@Test(priority=2)
  public void modifyUser()
  {
   sop("user modified successfully");
  }
@Test(priority=3)
public void deleteUser()
  {
   sop("user deleted successfully");
  }
```

Conclusion →

→ If we want to execute based on requirement we have to use an attribute called as priority. eg →
@Test(priority = priority number)

→ The lowest number will get the first priority {1, 2, 3}

→ If we don't provide priority test cases will be executing in alphabetical order.

* How to add TESTNG from libraries.
com.Ecommerce.falcom → Build path → configure build path → Add library → Testng → Next → Finish.

papergrid

Date: / /

→ Priority can be negative also. @Test(priority = -1)

→ The default priority is '0'.

→ If multiple priorities are same, then those priorities will be executed based on alphabetical order.

* By giving attribute "invocationCount" →

→ If we want to execute a test case multiple times we have to use an attribute called as invocationCount.

→ We can provide multiple attributes inside @Test annotation.

```
@Test(priority=1 , invocationCount =5)
public void createUser()
{
    sop ("user created successfully");
}
```

bydefault invocationCount = 1, less than 1 it won't run like '0' or '-1'.

papergrid

Date: / /

12/3/24

13/3/24

* How we can stop a test case to participate in execut^n;

→ This operation we can perform in 3 ways -

a) we can use an attribute called as "enabled = false".

eg:- @Test (enabled= false)

b) we can use an annotation called as "Ignore".

eg:- @Ignore
@Test

c) we can provide "invocation count as 0".

eg:- @Test (invocationCount =0)

* If we want to create dependency in b/w multiple methods or multiple test case there we have to use an attribute called as "dependsOnMethods".

eg:- @Test (dependsOnMethods = "createUser")

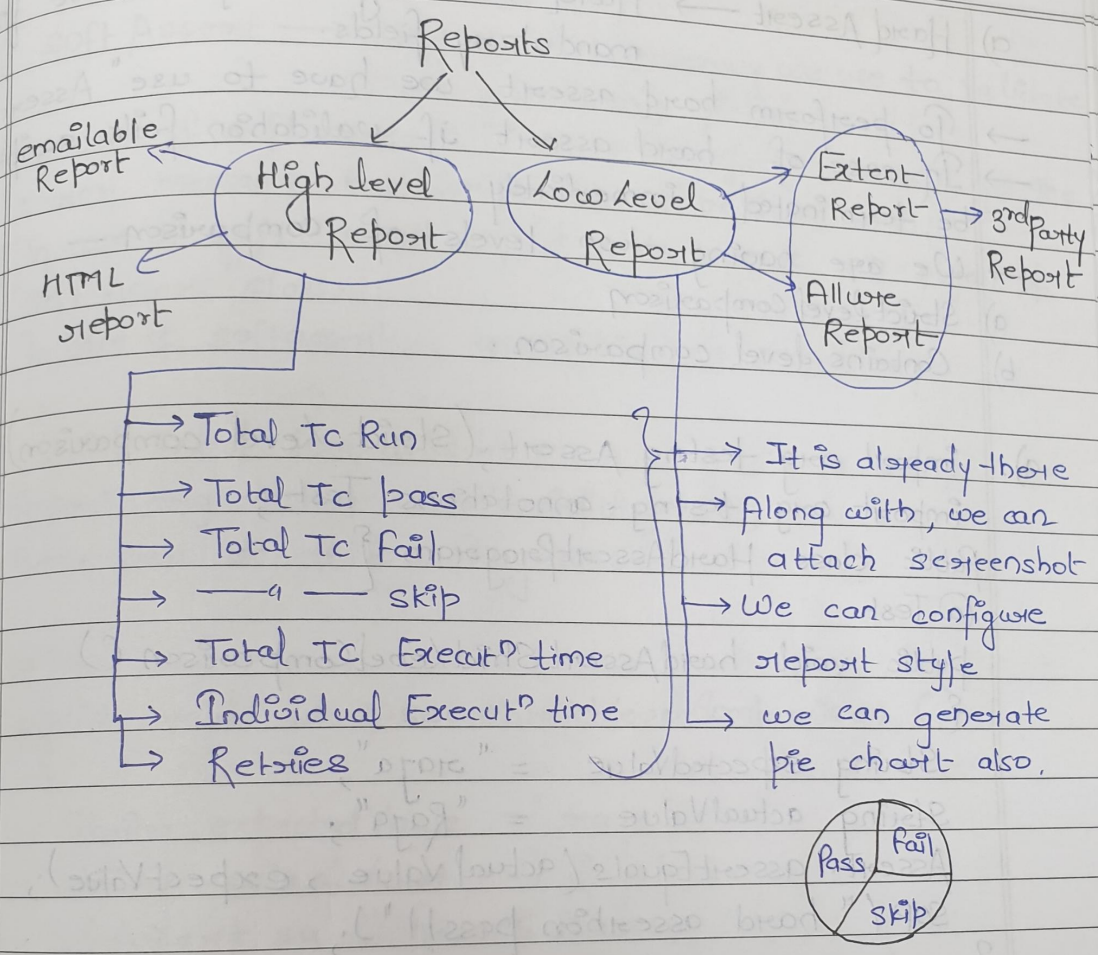
* If we want to reflect an important message in the report what we have to do?

→ To acheive this particular functionality we have to use a method called as log(), which is a static method present in "Reporter class".

```
eg:- @Test
     public void modifyUser()
     {
        sop("user modified successfully");
        Reporter.log("this method depends on createUser");
```

{for checking the report → 1st refresh our project
go to test o/p → emailable.report → open with → web browser}

TestNG Reports



* Assertions →

→ In real time assertions means Validation.

→ Instead of if & else we go for assertions because if & else does not have that much capability to fail a testcase. That's y we should go for assertions.

→ We are having two types of assertions –

a) Hard assert

b) Soft assert

papergrid

Date: / /

a) Hard Assert → Hard assert generally we use for mandatory fields.

→ To perform hard assert we have to use "Assert" class.

→ In case of hard assert if validation fails script will be terminated immediately.

→ We are having two levels of comparison —

a) Strict level Comparison

b) Contains level comparison.

a). import org.testng.Assert; (Strict Level comparison)

```
import org.testng.annotations.Test;
Public class HardAssertProgram {
@Test
public void hardAssertStrictLevelComparison ()
{
  String expectedValue = "raja";
  String actualValue = "Raja";
  Assert.assertEquals(actualValue, expectValue);
  SOP("hard assertion pass!!");
}
```

b) Contains Level comparison

```
@Test
public void hardAssertContainsLevelComparison ()
{
  String expectedValue = "Maharaja";
  String actualValue = "raja";
Assert.assertTrue(expectedValue.contains(actualValue));
SOP("hard assert contains level pass!!");
}
```

b) Soft Assert → Soft assert generally we use to validate non mandatory fields.

→ In case of soft assert if assertion fails we will get the error message but script will not be terminated.

→ In case of soft assert we have to create an object of "Soft Assert" class.

→ In case of softassert, assertAll() is mandatory.

a) Strict Level Comparison

```
import org.testng.asserts.SoftAsserts;
Public class soft Assert Program {

@Test
public void softAssertStrictLevelComparison ()
{
    String expectedValue = "raja";
    String actualValue = "Raja";
    SoftAssert sa = new SoftAssert ();
    sa.assertEquals (expectedValue, actualValue);
    Sop(" soft assertion pass!! ");
    sa.assertAll ();
```

b) Contain Level Comparison

```
@Test
public void softAssertContainsLevelComparison ()
{
    String expectedValue = "Maharaja";
    String actualValue = "raja";
    SoftAssert sa = new SoftAssert ();
    sa.assertTrue (expectedValue.contains (actualValue));
    SOP (" soft assert contains level pass !! ");
    sa.assertAll ();
}
```

14/3/24

* Difference between Hard Assert & Soft Assert

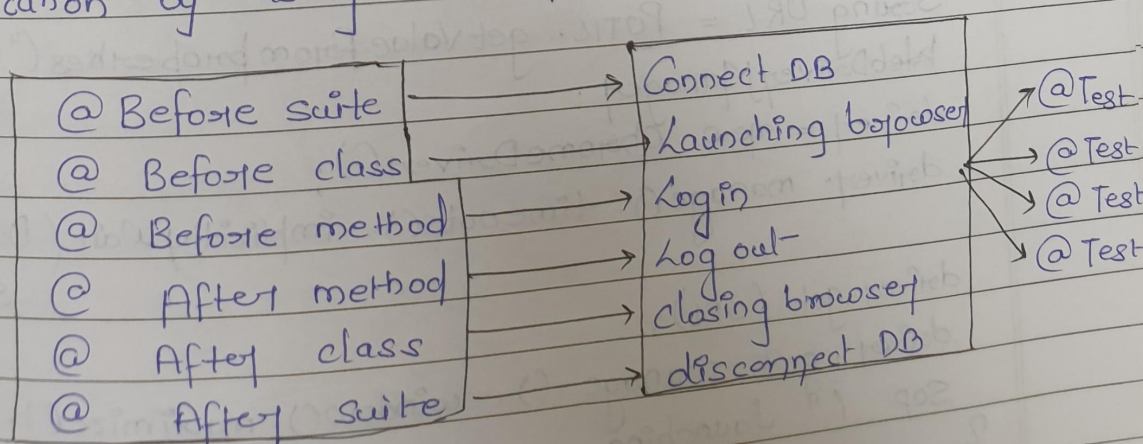
Hard Assert	Soft Assert
→ In case of hard assert if assertion fails, script will be terminated.	In case of soft assert if assertion fails we will get the error message but script will not be terminated.
→ In case of hard assert we have to use "Assert" class.	In case of soft assert we have to create an object of "softAssert" class.
→ In case of hard assert "assertAll()" is not required.	In case of soft assert "assertAll() ~~mett~~ is compulsary.

* TestNG annotations → Test Annotations are nothing but special types of instructions which is stored in the form of "metadata".

List of TestNG annotations →

1) @ Before suite
2) @ Before Class
3) @ Before method
4) @ Test
5) @ After method
6) @ After class
7) @ After suite
8) @ Data provider
9) @ Parameters
10) @ Ignore

* Base Class → Base class is a ~~special~~ specially design ~~inherit~~ class to keep all the common features of an application by using TestNG annotations.



* Base Class in a generic Way →

```
Package genericUtility;

import Java.time.Duration;
import org.openqa.selenium.By;
import org.openqa.Selenium.keys;
import org.openqa.selenium.Webdriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.After suite;
import org.testng.annotations.Before class;
import org.testng.annotations.BeforeMethods;
import org.testng.annotations.Before suite;

public class BaseClass {

public PropertiesUtility PUTIL = new propertiesUtility();
public Static WebDriver driver = null;

@BeforeSuite (alwaysRun = true)
public void CreateDbConnect()
{
SOP("Db connect Successful");
}

@Before Class (alwaysRun=true)
Public void launchingBrowser() throws Exception {

String URL = PUTIL.getValueFromproperties("url");
WebDriverManager.
driver = new ChromeDriver();
driver.manage().timeouts().implicitlyWait(Duration.of
Seconds(10));

driver.get(URL);
driver.get manage().window().maximize();
Sop("launching successfull");
}
```

papergrid

Date: / /

```
@BeforeMethod (alwaysRun = true)
public void loginOperation() throws Exception {

String USERNAME = PUTIL.getValueFromProperties("username");
String PASSWORD = PUTIL.getValueFromProperties("password");
driver.findElement(By.xpath("//input[@name='username']"))
        .sendKeys(USERNAME, Key.TAB, PASSWORD, Key.ENTER);
Sop("login successful");
}

@AfterMethod (alwaysRun = true)
public void logoutOperation()
{
driver.findElement(By.xpath("//p[@class='oxd-userdrop-
                down-name']")).click();
driver.findElement(By.xpath("//a[.='Logout']")).click();
Sop("logout successfull");
}
@AfterClass (alwaysRun = true)
public void closeBrowser()
{
driver.quit();
Sop("browser closed successfully");
}

@AfterSuite (alwaysRun = true)
public void closeDbconnection()
{
Sop("Db connection closed successfully");
}
```

papergrid

Date: / /

Main Execution →

```
import org.openqa.selenium.By;
import org.testng.annotations.Test;

import genericUtility.BaseClass;
import genericUtility.ExcelUtility;

public class AD001 extends BaseClass {

@Test (groups = {"regression"})
public void tc01 () throws Exception {

ExcelUtility EUTIL= new ExcelUtility ();
String TCUsername = EUTIL.getDataFromExcel ("Admin", 1, 1);
String EMPNAME = EUTIL.getDataFrom Excel ("Admin", 1, 2);

driver.findElement(By.xpath("//span[.='Admin']")).click();

driver.findElement (By.xpath("//input [contains (@class, 'oxd'
]] [2]")).
sendKeys (TCUsername);
driver.findElement(By.xpath("//input[@ placeholder =
'type for hints....']")).sendkeys
(EMPNAME);
driver.findElement(By.xpath("//button[.= 'Search']")).
click();
sop ("TC01 executed successfully");

}

}
```

papergrid

Date: / /

Scenario 43 → Launch ChromeBrowser
→ Access OrangHRM
→ Do the login operation
→ click on PIM module
→ provide Empname & Empid
→ click on search button
→ perform logout operation.

```
public class PIM001 extends BaseClass {

@Test (groups = {"smoke"})
public void pim001() throws Exception {

ExcelUtility EUTIL = new ExcelUtility();
String PNAME = EUTIL.getDataFromExcel("PIM", 1, 1);
String PIM_ID = EUTIL.getDataFromExcel("PIM", 1, 2);

driver.findElement(By.xpath("//span[contains(@class,'oxd-
        text oxd-text--span')])[2]")).click();

driver.findElement(By.xpath("(//input[@placeholder='Type
        for hints...'])[1]")).sendKeys(PNAME);

driver.findElement(By.xpath("//input[@class='oxd-
        input oxd-input--active'])[2]")).sendKeys
        (PIM_ID);

SOP("this tc is successfully executed");
}

}
```

papergrid

Date: / /

Scenario 44 → Launch Chrome Browser
→ Access HRM
→ Do the login operation
→ Click on Claim module
→ click on search button
→ perform logout operation.

```
@Listners(genericUtility.ListenersImplementation.class)
public class Cl001 extends BaseClass {

@Test(groups = {"regression", "smoke"})
public void cl001() throws Exception {

ExcelUtility EUTIL = new ExcelUtility();
String PNAME = EUTIL.getDataFromExcel("CLAIM",1,1);
String REF_ID = EUTIL.getDataFromExcel("CLAIM",1,2);

driver.findElement(By.xpath("//span[contains(@class,
                                'ord')])[11]")).click();

driver.findElement(By.xpath("(//input[@placeholder=
                'Type for hints....'])[1]")).sendKeys
                                (PNAME);

driver.findElement(By.xpath("(//input[@placehol-
                der='Type for hints....'])[2]")).
                sendKeys(REF_ID);

Sop("this Cl001 tc is successfully executed");
}
}
```

papergrid
Date: / /

* Special Executions in TestNG —

1) Batch Execution → Executing multiple test cases 1 by 1 is called as batch execution.

2) Parallel Execution → Executing multiple test cases at a same time in multiple browsers to save time is called as parallel execution.

```
<suite name = "Suite" parallel = "methods">
```

3) Group Execution → Dividing the testcases into multiple groups and based on the requirement executing the groups is called as group execution.

⇒ to perform group execution we need do some changes.

a) @Test(groups = {"regression", "smoke"})

return type of groups is "String[]".

b) In testingGroupExecution.xml ⇒

(already there) → <suite> , after suite we have to give groups.

```
<groups>
<run>
<include name = "regression"></include>
</run>
</groups>
```

papergrid

Date: / /

4) Cross browser Execution → Executing the testcases in different-different types of browsers is called as cross browser Execution.

→ Cross browser execution generally we do to perform compatibility testing.

⇒ Steps to convert the testcases into xml suite file.

a) Select the classes
b) Right Click on it
c) Scroll down till TestNG & select convert to TestNG.
d) click on finish.

* For execution in suite level →

→ We have to follow few steps –
a) Select all test cases and move into tc_repository package in src/test/java.

b) After that select all test cases and right click on it → Go in TestNG → Convert to TestNG → "in location you have to provide which type of execution you want like Batch execution, parallel execution or Group execution.

c) And you have to provide name in Location after testng.
eg → Location: /com.Ecommerce.Falcome/testng Batch Execution.xml

d) Click on finish.

e) Now one BatchExecution.xml is generated down to pom.xml.

f) you click on BatchExecution.xml and click on run button.

g) One pop up will be generated then you have to select "TestNG suit" and then click on 'ok' button.

h) your program will run 1 by 1 in Batch Execution.

16/3/24

* Special Interfaces of TestNG →

→ In testNG 3 special interfaces are present –

1) I test listner
2) I test Result
3) I Retry Analyzer

1) I test listner → The functionality of this interfaces is to monitor each and every testcase.

2) I test Result → The functionality of this interfaces is to capture the run time status of a test case which has been monitored by I test listner.

papergrid
Date: / /

```
Package genericUtility
import java.io.File;
import java.io.IOException;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.testng.ITestContext;
import org.testng.ITestListener;
import org.testng.ITestResult;
import org.testng.Reporter;

import com.google.common.io.Files;

public class ListenersImplementation implements ITestListe

@Override
public void onTestStart(ITestResult result) {

String methodName = result.getMethod().getMethodName()
Reporter.log(methodName + "execution successful!!");

}

@Override
public void onTestSuccess(ITestResult result) {

String methodName = result.getMethod().getMethodName
Reporter.log(methodName + "execution successfull");

}
```

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

String methodName = result.getMethod().getMethodName();

TakesScreenshot ts = (TakesScreenshot)Baseclass.driver;
File SRC = ts.getScreenShotAs(OutputType.FILE);
File DEST = new File(".\\Screenshot folder\\" + methodName+
                                        ".png");
try {
    Files.copy(SRC, DEST);
    }
catch (IOException e)
    {
      e.printStackTrace();
    }
}

@Override
public void onTestSkipped(ITestResult result) {

String methodName = result.getMethod().getMethodName();
Reporter.log(methodName + " execution skipped");

}

}
```

Note → Listeners implementation we can pass in suit level also.

→ We have to pass that in b/w suite & test

papergrid
Date: / /

```
<suite name = "suite">
< listeners >
    < listener class name = "genericUtility.ListenersImplementation" />
</listeners>
```

3) I Retry Analyzer → In realtime if testcases failed because of some synchronisation issue, server issue, browser issue or some miscellaneous issues, in this cases we have to go for retry analyzer.

→ Retry Analyzer we should always provide in script level otherwise it will affect the execution time.

```
package - genericUtility

import org.testng.IRetryAnalyzer;
import org.testng.ITestResult;

public class RetryAnalyzerImplementationClass implements
                                    IRetryAnalyzer {
  int count = 1;
  int retryCount = 3;
  @ Override
  public boolean retry (ITestResult result) {
    while (count<= retryCount) {
        count++;
        return true;
    }
    return false;
}
```

```
public class Cl001 extends BaseClass {

@Test(retryAnalyzer = genericUtility.RetryAnalyzerImplementation Class.class)
```

17/3/24 Stale Element Refrence Exception →

→ Stale element refrence Exception means old element refrence.

→ If we are performing some operations in a webpage and suddenly the web page get refresh or reload, we will get this exception.

→ Whenever any page loaded or reloaded new "Id" will be generated, and in this case driver will try to perform in the old "Id" or refrence, that's y this exception will be generated.

→ To handle this stale element refrence exception we have to go for pom class.

* POM CLASS →

→ Pom is a specially designed by goo class by google to handle stale element refrence exception.

→ Pom stands for page object ~~order~~ model.

→ Pom also acts as a repository (elements repository).

→ All the POM classes we have to stored in a object repository.

papergrid

Date: / /

* Advantages of POM →

→ We can handle stale element refrence exception.
→ Maintainence of elements are very easy.
→ POM is perfectly fit for "Agile process".
→ All the elements will be encapsulated.
→ Because of the buisness libraries code will be optimized.
→ Because of findAll we can acheive "auto healing" process [upto a certain extent]

* Steps to design POM class →

→ Number of web pages = no. of POM classes.
→ In case of POM classes all the element should be private and it comes under findBy, findBys and findAll.
→ We need a public constructor for initialization.
→ We need getters to access the private members.
→ We should create buisness library to perform code optimization.

* for login of OHRM →

```
Package ObjectRepository;

/**
* this is a POM class for login page
* @author Mamta Choudhary
* @ Version 1.3.24
*/

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.findBy;
import org.openqa.selenium.support.PageFactory;

public class LoginPage {

@findBy (xpath = "//input[@placeholder = 'username']")
private WebElement username;

@findBy (xpath = "//input[@placeholder = 'password']")
private WebElement password;

@findBy (xpath = "//button[@type = 'submit']")
private WebElement loginBtn;

public LoginPage (WebDriver driver) {
    PageFactory.initElements (driver, this);
}
```

Date: / /

```
public WebElement getUsername() {
    return username;
}

public WebElement getPassword() {
    return password;
}

public WebElement getLoginBtn() {
    return loginBtn;
}

/**
 * this is the buisness library to perform login
   operation
 * @param UName
 * @ param PASS
 */

public void loginOperation( String UNAME , String
                                        PASS)
{
  username.sendkeys (UNAME);
  password.sendkeys (PASS);
  loginBtn.click ();
}
```

In baseclass we have to change →

for login operation

```
LoginPage LP = new LoginPage (driver);
LP. loginOperation (USERNAME, PASSWORD);
```

for logout operation

```
HomePage HP = new HomePage (driver);
HP. logout ();
```

* for Homepage of OrangeHRM →

```
public class Home Page {

@ findBy (xpath = "//span [text()= 'Admin']")
private WebElement admin;

@ findBy (xpath = "//span[text()= 'PIM']")
private WebElement Pim;

@ find By (xpath = "span[text() = 'claim']")
private WebElement claim;

@ findBy (xpath = "//p[@class = 'oxd-usedropdown-
name']")

private WebElement about;
```

```
@FindBy (xpath = "//a[text() = 'Logout']")
private WebElement logoutBtn;

public HomePage (WebDriver driver) {
pagefactory.initElements (driver, this);
}

public WebElement getAdmin() {
  return admin;
}
public WebElement getPim() {
  return pim;
}

public WebElement getClaim() {
  return claim;
}

public WebElement getAbout() {
  return about;
}

public WebElement getlogoutBtn() {
  return logoutBtn;
}
```

```
/**
* this method is to click on admin and logout
*/

public void adminAndLogout()
{
    admin.Click();
    about.click();
    logoutBtn.click();
}

/**
* this method is to click on pim and logout
*/

public void pimAndLogout()
{
    pim.click();
    about.click();
    logoutBtn.click();
}

/**
* this method is to click on claim and logout
*/

public void claim()
{
    claim.click();
    about.click();
    logoutBtn.click();
}
```

```
/**
* this method is to perform logout operation.
*/

public void clickLogout()
{
  about.click();
  logoutBtn.click();
}

public void clickOnAdmin()
{
  admin.click();
}

public void clickOnPim()
{
  pim.click();
}
public void clickOnClaim()
{
  claim.click();
}
```

papergrid

Date: / /

for Main execution →

```
package tc_Repository;
import org.testng.annotations.Test;

import genericUtility.BaseClass;
import objectRepository.HomePage;

public class PomClassExecution extends Baseclass {
@Test
public void adminModule ()
{
HomePage HP = new Homepage (driver);
HP.clickOnAdmin();
}
}
```

* findBys → In case of findBys we can capture multiple WebElements and it will works as an 'AND' operator which means all the elements should be matched.

Syntax → @findBys ({@findBy (xpath = " "), @findBy (xpath = " ")})

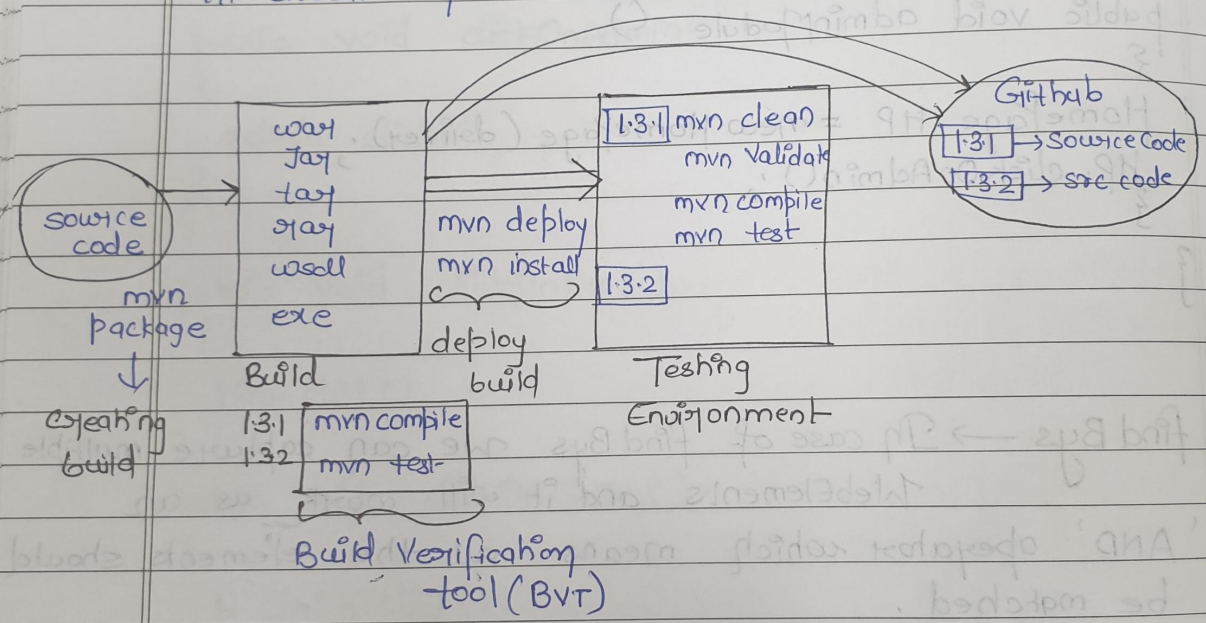
* findAll → In case of findAll we can capture multiple WebElements. It acts as a 'OR' operator which means if one element matches it will works.

Syntax →

@findAll({@findBy(xpath = " "), @findBy(xpath = " ")})

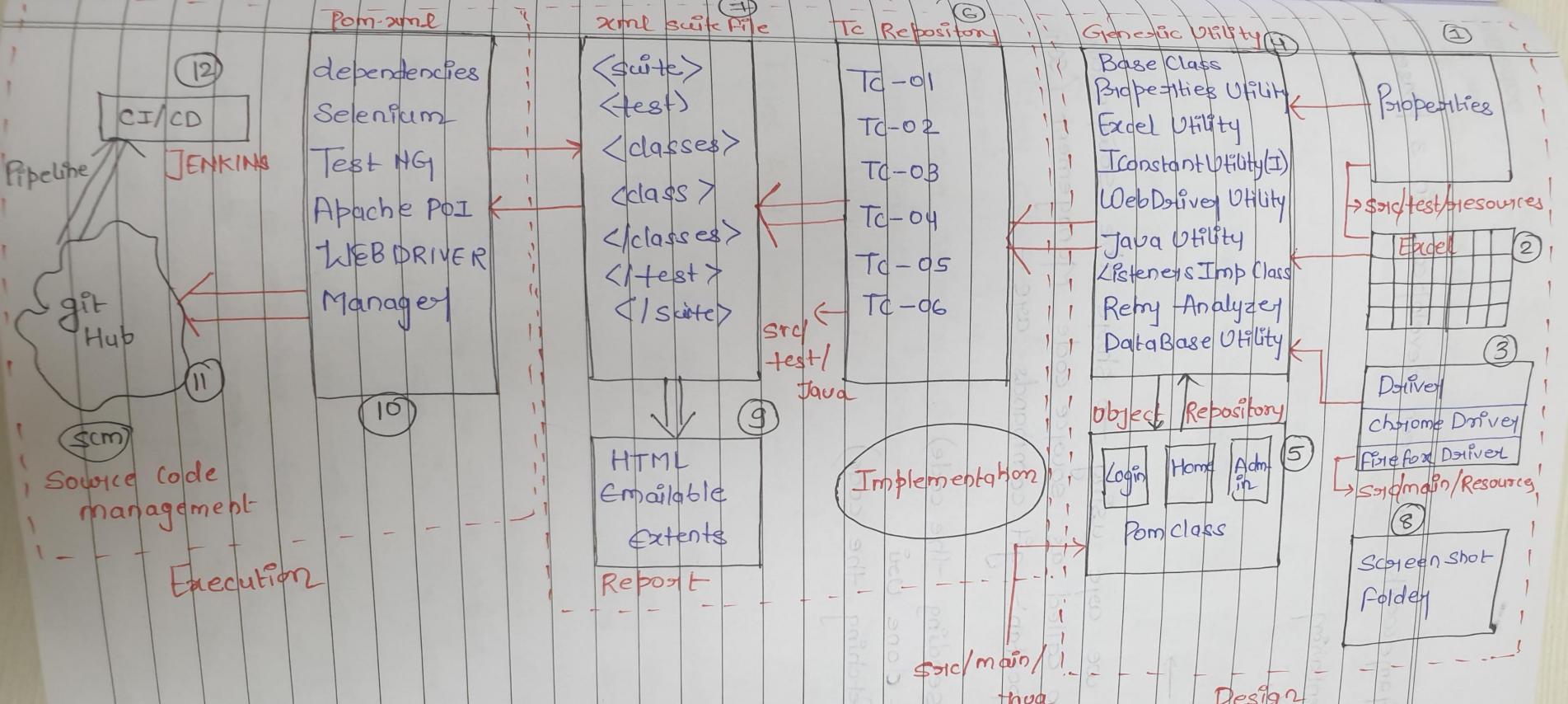
* Maven →

→ Maven is also called as Build management tool. With the help of maven we can acheive the creation of a build till executn of a build.



mvn package mvn Compile mvn test mvn deploy mvn install	→ developers
mvn clean mvn validate mvn compile mvn test	→ Test Engineers maven life cycle

* FRAME WORK WORK FLOW DIAGRAM

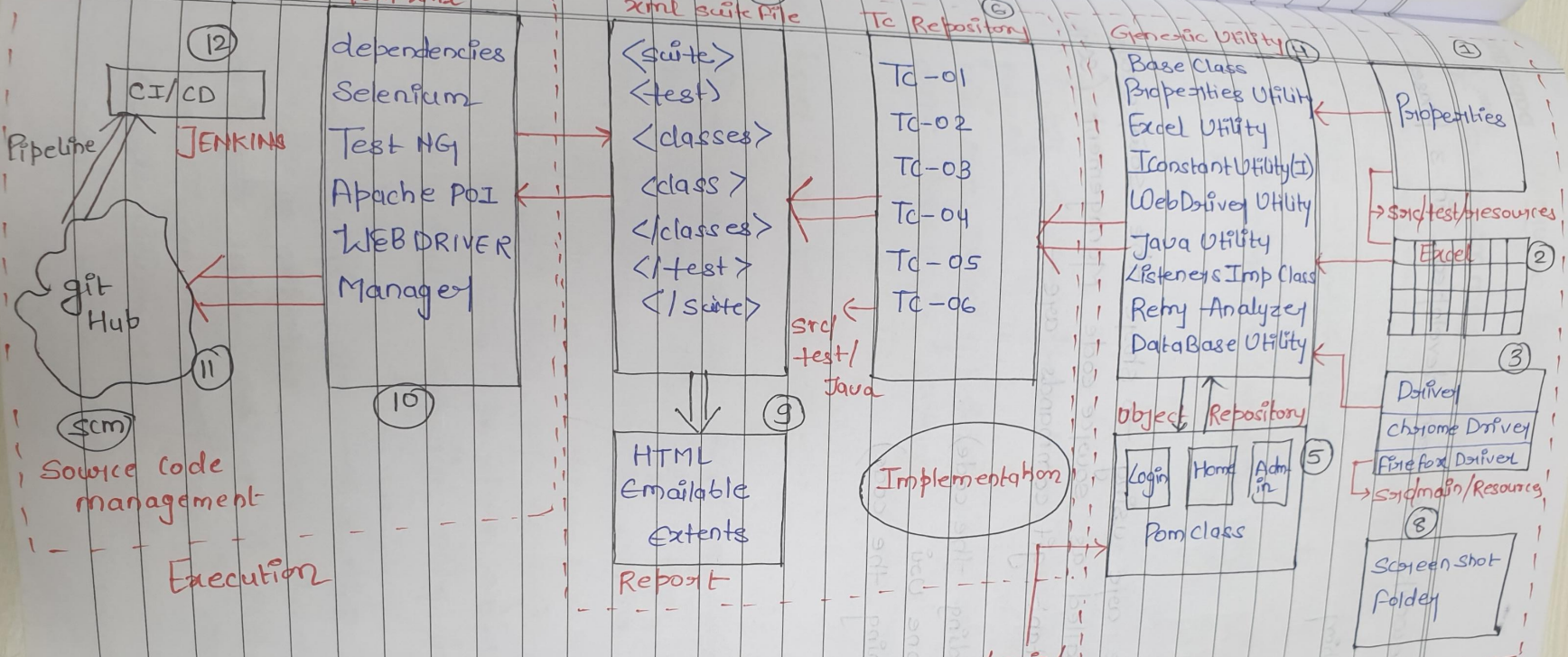


Note → last 15 days what we have learnt apart from using all those things not a single task is for automation test engineers. It's all are developer tasks

→ Only we've to write test cases and converting it to the suite.

→ Framework creation is not our task, We are use that framework to write the test cases. If in this any locators are not working then only you can do the changes.

* FRAME WORK WORK FLOW DIAGRAM →



Note → Last 15 days what we have learnt apart from using all those things not a single task is for automation test engineers. It's all are developer tasks

→ Only we've to write test cases and converting it to the suite.

→ Framework creation is not our task, We are use that framework to write the test cases. If in this any locators are not working then only you can do the changes.

→ Our framework has been developed in 3 phases -
a) design
b) Implementation
c) Execution.

* GitHub →

→ Github we are using to store the source code.
→ It is also called as source code Management (SCM).
→ Some important git commands are -
a) Commit
b) Push (sending the code)
c) import + clone Uri
d) pull (fetching the code)

Interview Points →

* For e-commerce domain →

search
↓
PLP → Product list page
↓
PDP → Product Description page
↓
cart

* Issue →

1) After adding Product in cart, product is not displayed.
2) We are selecting shoes, and when we are adding in cart default is selected.

* Roles And Responsibility →

→ ~~first~~ So the my roles & responsibility is converting test case into test scripts then we do dry run for our test scripts in scripts level.

→ If it is passed then we convert our test scripts into suites according to their cases. And again we perform dry run in suite level.

→ But sometimes some test scripts get failed in suite level on that time we will run only the fail test cases and analyze it and if any bug is there we will report it to the developer through JIRA.

Date: / /

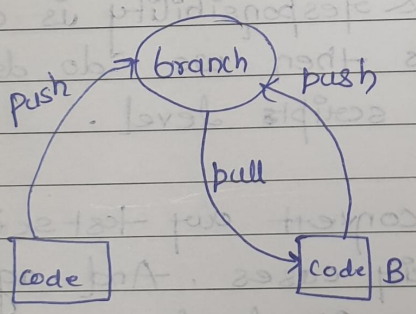
→ When our test scripts will pass in both suite and script level then I create branches and raise a PR to the manager.

→ My manager will review everything & if it is all fine then my ~~team lead~~ manager will merge it to the master class. & if it is not fine then it will revert back to me.

→ ~~Then it will go to Github and from there~~ Once it will merge to the master from there it will go to CI/CD tool for the execution.

→ Under Github we have a master repository or master branch

* Github → Conflict →



Github conflict faced by a second person when he/she is trying to push a code in branch because according to Github, the person is not aware by the changes which has done by the first person then that time Github will show conflict. for dealing with this issue first we have to blindly pull from the gith

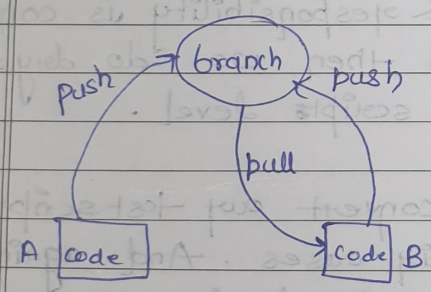
→ When Dev test scripts will pass in both suite and script level then I create branches and raise a PR to the manager.

→ My manager will review everything & if it is all fine then my ~~team lead~~ manager will merge it to the master class. & if it is not fine then it will revert back to me.

→ ~~Then it will go to Github and from there~~ Once it will merge to the master from there it will go to CI/CD tool for the execution.

→ Under Github we have a master repository. or master branch

* GitHub → Conflict →



Github conflict faced by a second person when he/she is trying to push a code in branch because ~~a~~ according to Github, the person is not aware by the changes which has done by the first person then that time ~~it~~ Github will show conflict. for dealing with this issue first we have to blindly pull from the github

then we ve to push it to the branch.

* Scheduling → (Jenkins) — related to

1) On demand → When the demand comes, it will trigger
2) On scheduling → In what time we are scheduling that time this schedule will happen.
3) Poll scm (source code management) → when any changes happen in github automatically it will update you through poll scm.

→ Master Branch is all the combinatn of all branches.

* Frame Work → Hybrid Framework
  - → data driven
  - → TestNG

→ developed in 3 phases

a) design
- object class → (Pom classes)
- Generic Utility
  - ↳ web driver
  - ↳ Base class → (for common features like login, logout operatn)
  - ↳ Properties Utility
  - ↳ Excel Utility
  - ↳ I constant Utility
  - ↳ Listners Implementation

b) Implementatn
↓
Test Scripts (dry run)

c) Execution
↓ suite
Test Scripts (dry run)
→ SS (Extent)
↓
Report
↓
Pom xml (dependencies)

Jenkins ← Github

* Maven →

mvn clean → clean All old reports

mvn validate
It will validate the dependencys

mvn compile
↓
It will perform Compilation

mvn test
↓
It will execute.

* TestNG →

1) Priority
2) Invocation Count
3) @Ignore
   enabled = false
   Invocation Count = 0 or -ve
4) depends on Method
5) Annotations
   ↳ Base Class
6) Assertions → Hard Assert or Soft Assert
7) Listeners → Extent Report
8) Excel sheet Connections or properties file.