NStarX™

Address:

3 Cube Towers, White Field Rd,

Whitefields, HITEC City,

Hyderabad, Telangana 500081

Standard Operating Procedure of Automation Framework - Selenium with Java (*Demo*)

24th February 2023

Created By:

Prakash Srivastava

Vikram Danvale

Ranadheer Durgi

Under Supervision of:

Razia Sultana Syed

Table Of Contents

8. TestNG

9. Test Output

10. XML Files

- 10.1. BatchExecution.xml
- 10.2. GroupExecution.xml
- 10.3. ParallelExecution.xml
- 10.4. pom.xml
- 10.5. testng.xml

11. Drivers

12. Extent Report

13. Screenshot

14. Priority In TestNG

15.TestNG Dependent Test

1. Introduction

The Selenium Framework is basically code structure that makes code maintenance easy and efficient. Without frameworks, users may place the code at some location which is not reusable or readable. Using frameworks leads to increased code reusability, higher portability, reduced cost of script maintenance, better code readability, etc. The Hybrid framework in Selenium offers these advantages by functioning as a combination of both Keyword-Driven-Framework and Data-Driven-Framework.

2. Configuration

Softwares Required

1. JDK(Java Development Kit)
2. Eclipse(Download exe file from Selenium.Dev)
3. Browsers(Chrome, Firefox)
4. GITLab Account
5. URL of Git repository, which is provided by the team lead.

Steps To Follow:-

Step:- 01

Go to the Eclipse and click on file and then click on import option.

Step:- 02

Click on the Git folder on the import window and then click on the project from Git and click on the next button.

Step:- 03

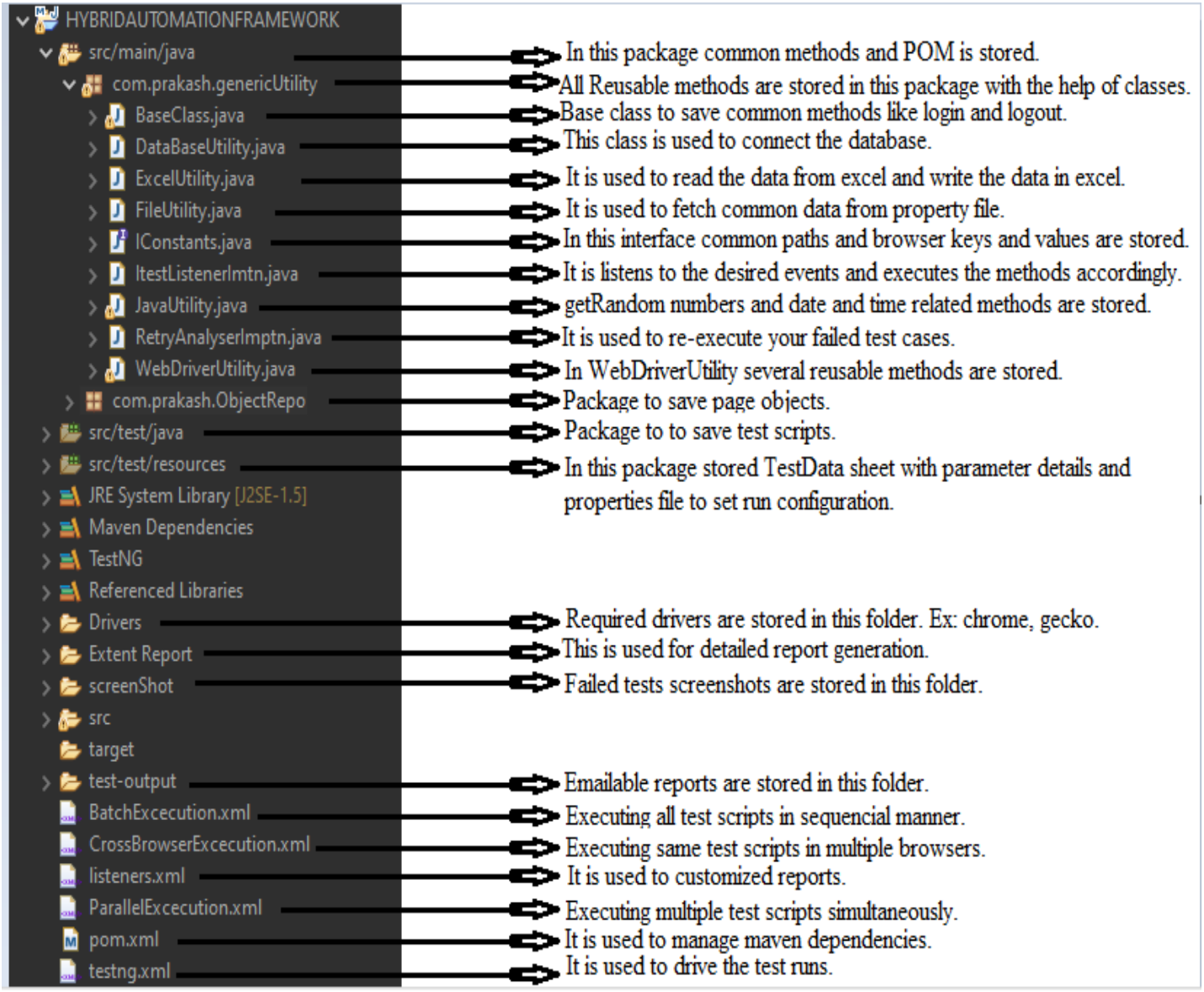
Then the source Git repository window will open and then click on the Clone URI option and click on the next button.

Then paste Url in the URI field and Enter username and password in appropriate fields and then click on next button and then click on finish button.

Project will be imported successfully and it will be visible in the left side section.

2.1 Structure of Framework

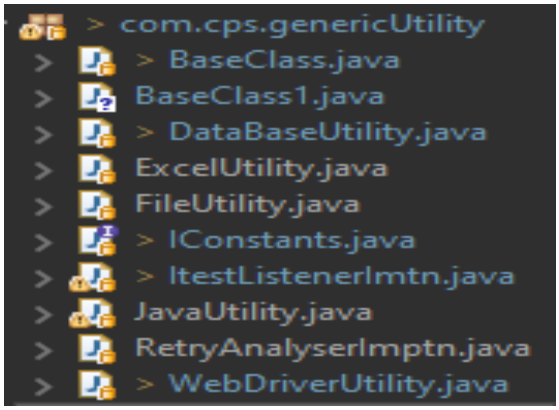
The picture below describes the short information about each and every element of the framework.



3. Directory-src/main/java

3.1. Generic Utility

It is one of the package of the demo framework that created several classes by using reusable methods. And all those classes are broadly explained below one by one.



3.1.1. BaseClass.java

Base class was created under the Generic Utility package and it is performed several operations like.

1. Open the browser and enter the URL.
2. Enter the login credentials and login to the application.
3. Logout the application and then close the browser.

The above operations reusable methods are used so we don't need to write these steps again repeatedly, just need to "extend" base class in executable script.

Screenshot of base class code attached below and in that package name is mentioned where the values stored of particular fields with arrow mark.You need to go to that particular package and change the values as per your requirement.

```java
public class BaseClass1 {

    public static WebDriver sdriver;
    public WebDriver driver;
    public DataBaseUtility dLib=new DataBaseUtility();
    public ExcelUtility eLib=new ExcelUtility();
    public FileUtility fLib=new FileUtility();
    public WebDriverUtility  wLib=new WebDriverUtility();
    public JavaUtility jLib=new JavaUtility();

    @BeforeClass
    public void launchTheBrowser()
    {
        String BROWSER = null;
        try {
            BROWSER = fLib.getPropertKeyValue("browser");
        } catch (Throwable e) {
            e.printStackTrace();
        }
        System.out.println(BROWSER);
        String URL = null;
        try {
            URL = fLib.getPropertKeyValue("url");
        } catch (Throwable e) {
            e.printStackTrace();
        }
        if(BROWSER.equalsIgnoreCase("firefox"))
        {

            driver=new FirefoxDriver();
        }else if(BROWSER.equalsIgnoreCase("chrome"))
        {
            driver=new ChromeDriver();
        }
        else {
            driver=new ChromeDriver();
        }
        System.out.println("Browser successfully launched");
```

3.1.2. DataBaseUtility.java

It is one of the class of Generic Utility and it mentioned the code to connect the database and to disconnect from the database.

```java
package com.cps.genericUtility;

public class DataBaseUtility {
    /**
     * connection to DB
     */
    public void connectToDB()
    {
        System.out.println("database connection is successful");
    }
    /**
     * Closing DB
     */
    public void closeDB()
    {
        System.out.println("close DB connection");
    }
}
```

3.1.3. ExcelUtility.java

It is one of the class of generic utility package. This class is mentioned used to performed the operations like,

1. To read the data from excel
2. To write the data In excel.

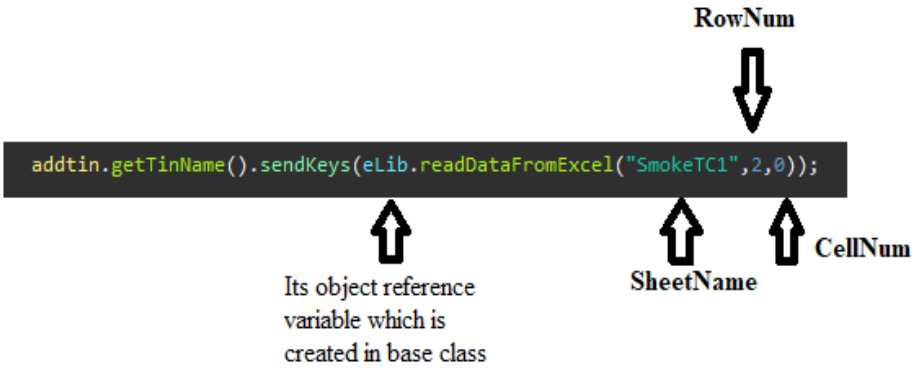
Because in the executable script we don't hardcode the values, instead of that we fetched the values from the excel.

Steps To Follow:-

1. Go to the src/test/resources package and open the TestData.xls excel sheet and create excel sheet and give the sheet name.

2. When you called that method in the executable script then at that time mention the sheet name which you created and also enter the appropriate row number and cell number.

For Ex.



```java
package com.cps.genericUtility;

import java.io.FileInputStream;


public class ExcelUtility {
    /**
     *its used to read the data from excel file
     */
    public String readDataFromExcel(String sheetName,int rowNum,int cellNum){
        FileInputStream fileInputStream = null;
        try {
            fileInputStream = new FileInputStream(IConstants.excelPath);
        } catch (FileNotFoundException e1) {
            e1.printStackTrace();
        }
        HSSFWorkbook workbook = null;
        try {
            workbook = (HSSFWorkbook) WorkbookFactory.create(fileInputStream);
        } catch (EncryptedDocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        HSSFSheet sheet = workbook.getSheet(sheetName);
        HSSFRow row = sheet.getRow(rowNum);
        HSSFCell cell = row.getCell(cellNum);
        String data = cell.toString();
        return data;
    }
```

```java
/**
 * its used to write data into excel file
 */
public void writeDataIntoExcel(String sheetName,int rowNum,int cellNum,String data) {
    FileInputStream fileInputStream = null;
    try {
        fileInputStream = new FileInputStream(IConstants.excelPath);
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
    Workbook workbook = null;
    try {
        workbook = WorkbookFactory.create(fileInputStream);
    } catch (EncryptedDocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    Sheet sheet = workbook.getSheet(sheetName);
    Row row = sheet.getRow(rowNum);
    Cell cell = row.createCell(cellNum);
    cell.setCellValue(data);
    FileOutputStream fileOutputStream = null;
    try {
        fileOutputStream = new FileOutputStream(IConstants.excelPath);
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
    try {
        workbook.write(fileOutputStream);
    } catch (IOException e) {
        e.printStackTrace();
    }
```

3.1.4. FileUtility.java

It's one of the class of Generic Utility and it's used to get common data from property file based on the key which is specified as an argument in the base class.

For example:- "url", "username", "password".

Here the file path of common.properties is already saved in IConstant Interface.

```
public class FileUtility {
	public String getPropertKeyValue(String key){
		FileInputStream fileInputStream = null;
		try {
			fileInputStream = new FileInputStream(IConstants.filePath);
		} catch (FileNotFoundException e) {
			e.printStackTrace();
		}
		Properties pres=new Properties();
		try {
			pres.load(fileInputStream);
		} catch (IOException e) {
			e.printStackTrace();
		}
		String value = pres.getProperty(key);
		return value;
	}
```

3.1.5. IConstants.java

Its Interface of Generic Utility where all the common paths, browser key and browser values are stored.

```
package com.cps.genericUtility;
public interface IConstants
{
	String filePath="./src/test/resources/common.properties.txt";
	String excelPath="./src/test/resources/TestData.xls";
	String chromeValue="C:\\Users\\VikramDanvale\\.m2\\repository\\org\\testng\\testng
				\\7.7.0\\qa-automation\\Drivers\\chromedriver.exe";
	[Note:- You need to change this chrome value according to your driver location.]

	String chromeKey="webdriver.chrome.driver";
	String DbUrl="jdbc:mysql://localhost:3306/";
	String REPORT_PATH="./Extent Reports";
	String DBUsername="root";
	String DBPassword="root";
	int implicitlyWaitDuration=10;
	int explicitWaitDuration=10;
	int scriptTimeOutDuration=10;
```

3.1.6. ItestListener.java

This is the most frequently used TestNG listener. ITestListener is an interface implemented in the class, and that class overrides the ITestListener defined methods. The ITestListeners listens to the desired events and executes the methods accordingly. It contains the following methods:

1. onStart(): Invoked after test class is instantiated and before execution of any testNG method.

2. onTestSuccess(): Invoked on the success of a test

3. onTestFailure(): Invoked on the failure of a test

4. onTestSkipped(): Invoked when a test is skipped

5. onTestFailedButWithinSuccessPercentage(): Invoked whenever a method fails but within the defined success percentage

6. onFinish(): Invoked after all tests of a class are executedThe above-mentioned methods use the parameters ITestContext and ITestResult.

The ITestContext is a class that contains information about the test run. The ITestResult is an interface that defines the result of the test.

```
public class ItestListenerImtn implements ITestListener{
	ExtentSparkReporter spark;
	ExtentReports report;
	ExtentTest test;
	public void onTestStart(ITestResult result) {
		test=report.createTest(result.getMethod().getMethodName());
	}
	public void onTestSuccess(ITestResult result) {
		test=report.createTest(result.getName());
		test.pass(MarkupHelper.createLabel(result.getName()+ " method PASSED",
ExtentColor.GREEN));
	}
	 //To take screenshot of failed test scripts
	public void onTestFailure(ITestResult result) {
		test.fail(result.getThrowable());
		String filePath=null;
		try {
filePath=WebDriverUtility.takeScreenShot(BaseClass.sdriver,result.getMethod().
                                                    getMethodName());
		} catch (Exception e) {
			e.printStackTrace();
		}
		test.addScreenCaptureFromBase64String(filePath,
result.getMethod().getMethodName());
	}
	public void onTestSkipped(ITestResult result) {
		test=report.createTest(result.getName());
		test.skip(MarkupHelper.createLabel(result.getName()+"
			method SKIPPED", ExtentColor.YELLOW));
	}
	public void onTestFailedButWithinSuccessPercentage(ITestResult result) {
```

```
	}
	public void onStart(ITestContext context) {
		spark = new ExtentSparkReporter(IConstants.REPORT_PATH);
		spark.config().setTheme(Theme.DARK);
		spark.config().setDocumentTitle("CredentialMyDocs Reports");
		spark.config().setReportName("Prakash Reports");
		report = new ExtentReports();
		report.attachReporter(spark);
		report.setSystemInfo("PLATFORM", "Windown 11");
		report.setSystemInfo("Created By", "Prakash");
	}
	public void onFinish(ITestContext context) {
		report.flush();
	}
}
```

3.1.7. JavaUtility.java

In java utility class some methods are created like getRandomNumber(), getSystemDateAndTimeInISTformat(), getSystemDateAndTimeInFormat().

```java
package com.cps.genericUtility;

import java.util.Date;

public class JavaUtility {
    /**
     * its used to generate a random number
     * @return
     */
    public int getRandomNumber() {
        Random random=new Random();
        int randNum = random.nextInt(1000);
        return randNum;
    }
    /**
     * its used to get systemDateAndTime in IST Format
     * @return
     */
    public String getSystemDateAndTimeInISTformat() {
        Date date=new Date();
        return date.toString();
    }
    /**
     * its used to get system date and Time in required format
     * @return
     */
    public String getSystemDateAndTimeInFormat() {
        Date date=new Date();
        String dateAndTime = date.toString();

        String YYYY = dateAndTime.split(" ")[5];
        String DD = dateAndTime.split(" ")[2];
        int MM = date.getMonth()+1;

        String finalFormat = YYYY+" "+DD+" "+MM;
        return finalFormat;
    }
}
```

3.1.8. RetryAnalyser.java

It is an interface to implement to be able to have a chance to retry a failed test.

This method implementation returns true if you want to re-execute your failed test and false if you don't want to re-execute your test.

When you bind a retry analyzer to a test, TestNG automatically invokes the retry analyzer to determine if TestNG can retry a test case again in an attempt to see if the test that just fails now passes.

```
public class RetryAnalyserImptn implements IRetryAnalyzer{
	int count=0;
	int retrylimit=3;
	public boolean retry(ITestResult result) {
		if(count<retrylimit)
		{
			count++;
			return true;
		}
		return false;
	}
}
```

This code shows that the failed test case will run 3 times till it passes. In case it fails the third time, test execution will stop and TestNG will mark this case as failed. We can change the number of tries by changing the value of retrylimit.

3.1.9. WebDriverUtility.java

It's one of the class of Generic Utility package where several reusable methods are stored.

Whenever we require any of the method of them we need to called that particular method in POM or in executable script.

Reusable Methods:-

a) maximizeTheWindow()

This method is used to maximize the browser.

```
public void maximizeTheWindow(WebDriver driver)
	{
		driver.manage().window().maximize();
	}
```

b) minimizeTheBrowser()

This method is used to minimize the browser.

```
public void minimizeTheBrowser(WebDriver driver)
	{
		driver.manage().window().minimize();
	}
```

c) mouseOverAnElement()

This method is used to mouse over an element.

```
public void mouseOverAnElement(WebDriver driver,WebElement element)
{
    Actions action=new Actions(driver);
    action.moveToElement(element).perform();
}
```

d) switchTowindow()

This method is used to switch to a window using the title.

```
public void switchTowindow(WebDriver driver,String actualTitle)
{
    Set<String> set = driver.getWindowHandles();
    for (String string : set)
    {
        driver.switchTo().window(string);
        String title = driver.getTitle();
        if(title.contains(actualTitle))
        {
            break;
        }
    }
}
```

Its a variable which is we have to mention in POM while storing x-path of perticular element.

e) switchToFrame()

This method is used to switch from one frame to another by using path.

```
public void switchtoFrame(WebDriver driver,WebElement element)
    {
        driver.switchTo().frame(element);
    }
```

f) selectDropDownByVtext()

This method is used to select the dropdown option by visible text.

```
public void selectDropDownByVtext(WebElement element,String visible_text)
{
    Select select=new Select(element);
    select.selectByVisibleText(visible_text);
}
```

X-path of dropdown

Text of dropdown option

g) waitForPageToLoad()

It will wait for 10 sec till the page gets loaded. Implicitly wait time duration is stored in IConstant class, So if you want to change so you can change over there.

```
public void waitForPageToLoad(WebDriver driver)
{
    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(IConstants.implicitlyWaitDuration));
}
```

h) waitTillElementToVisible()

This method will wait till the element is visible.

```
public void waitTillElementToVisible(WebDriver driver,WebElement element)
{
    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(IConstants.explicitWaitDuration));
    wait.until(ExpectedConditions.visibilityOf(element));
}
```

i) clickOnEnterButton()

This method is used to click on the enter button using keyboard actions.

```
public void clickOnEnterButton(WebDriver driver)
{
    Actions action = new Actions(driver);
    action.sendKeys(Keys.ENTER).perform();
}
```

j) scrollBarAction()

This method is used to perform scroll bar action.

```
public void scrollBarAction(WebDriver driver)
{
    JavascriptExecutor javaScript = (JavascriptExecutor)driver;
    javaScript.executeScript("window.scrollBy(0,500)");
}
```

3.2. Object Repository

A repository means a central location in which data is stored and managed. An object repository is a common storage location where we store a collection of all required application objects and their properties.

In other words, an object repository is a central location where data is stored and managed. In the context of Selenium WebDriver, object repository is mainly used to store element locator values in a central location to avoid hard coding within scripts.

If the locator value of one web element changes, only the object repository needs to be changed rather than making changes in all test cases in which the locator has been used. This will increase the modularity of framework implementation.

3.2.1. LoginPage.java(POM)

```
public class CmdLoginPage extends WebDriverUtility {

        Note:- Here you just need to change the x-path of a particular webelement according to your requirement.

         //declaration

                @FindBy(xpath="//input[@placeholder='Email']")

                private WebElement emailtxtbox;

                @FindBy(xpath = "//input[@placeholder='Password']")

                private WebElement passwordtxtEdt;

                @FindBy(xpath = "//button[text()='Sign in']")

                private WebElement Btn;

                public CmdLoginPage(WebDriver driver) {
```

```
        PageFactory.initElements(driver, this);
    }
    public WebElement getEmailtxtbox() {
        return emailtxtbox;
    }
    public WebElement getPasswordtxtEdt() {
        return passwordtxtEdt;
    }
    public void clickLoginBtn()
    {
        Btn.click();
    }
    public void loginToApplication(String username,String password)
    {
        emailtxtbox.sendKeys(username);
        passwordtxtEdt.sendKeys(password);
        clickLoginBtn();
    }
```

3.2.2. HomePage.java(POM)

```
public class HomePage1 extends WebDriverUtility {

@FindBy(xpath="//p[@class='oxd-userdropdown-name']")private WebElement SignOut;

	@FindBy(xpath="//a[normalize-space()='Logout']")private WebElement LogoutLink;

	public HomePage1(WebDriver driver) {

		PageFactory.initElements(driver, this);

	}

	public WebElement getSignOut() {

		return SignOut;

	}

	public WebElement getLogoutLink() {

		return LogoutLink;

	}

	public void clickSignOutLink(WebDriver driver) {

		mouseOverAnElement(driver,SignOut);

		LogoutLink.click();

	}

}
```

4. Directory name-src/test/java

In this Directory we created an executable script by using the @Test method.

While writing test scripts we need to extend BaseClass is mandatory.

Package:- executableScript

Class:- LoginPage.java

This script is used to perform the operation to login to the application.

Here the homepage object is created because the getSignOut method is stored in Homepage POM, similarly we have to create an object of a particular POM class where your required methods are stored.

```
public class LoginPage extends BaseClass {

	@Test

	public void loginToApplication()

	{

		HomePage hpage=new HomePage(driver);

		hpage.getSignOut().click();

	}

}
```

Class:- HomePage.java

This script is used to perform the operation to get the title of the homepage.

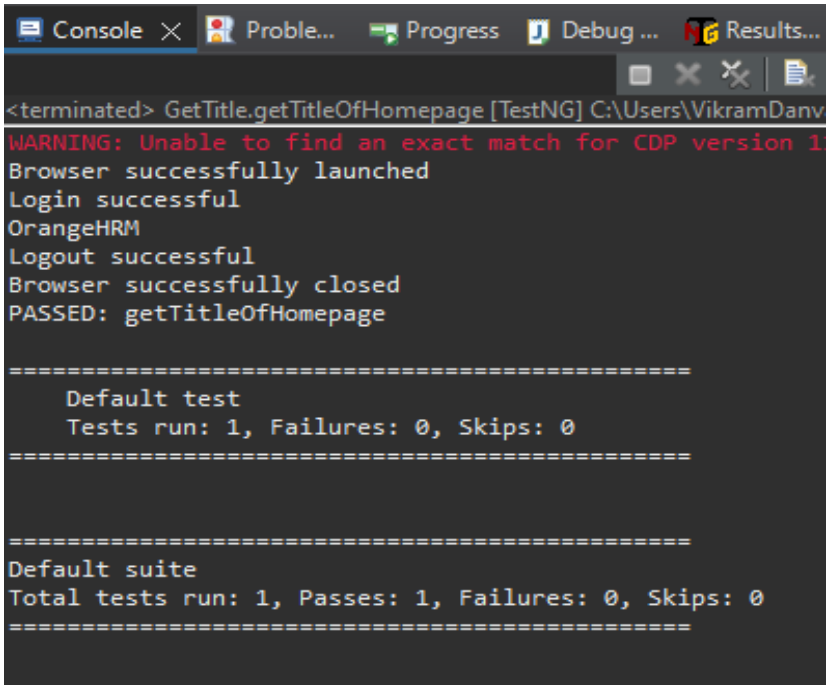
```java
package executableScript;

import org.testng.annotations.Test;

Run All
public class GetTitle extends BaseClass {


    @Test
    Run | Debug
    public void getTitleOfHomepage()
    {
        String title = driver.getTitle();
        System.out.println(title);

        HomePage hpage=new HomePage(driver);
        hpage.getSignOut().click();
    }
}
```



5. Directory-src/test/resource

In this Directory two files are stored

1. common.properties.txt

2. TestData.xls

This is used to read the data and fetch the data into the runtime of the test script.

5.1. Common.properties.txt

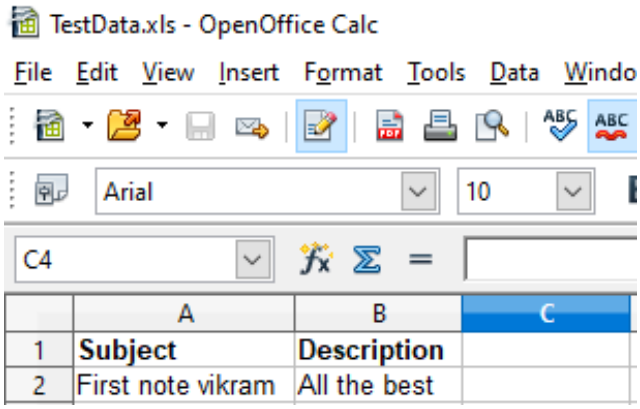
In this file common data is stored as a key-value pair, so we can access any data which is required by using a key.

```
1 url https://opensource-demo.orangehrmlive.com/web/index.php/auth/login
2 username Admin
3 password  admin123
4 browser chrome
```

Key Value

5.2. TestData.xls

This is an excel file where we can store multiple data and whenever we require we can fetch this data by using reusable methods in the executable script. Also we can edit the data any time, double click on the TestData file and edit and save as a current format.



6. JRE System Library

JRE System Library is added by Eclipse IDE automatically on creating Java Projects. JRE System Library implements the Java API in Eclipse IDE. So all the predefined functions of Java, can be accessed by the Java Program written in Eclipse IDE because of this JRE System Library.

7. Maven Dependencies

In the Context of Maven, dependency is simply a JAR file used by a java application. Based on the POM file, Maven will download and add the JAR file to our java path. Java will then be able to find and use the classes in the JAR file. Also in smaller projects with few functionalities or modules, a user can manually download JAR's or dependencies and can add them in the project.

8. TestNG

The folder acts as a repository/artifactory for all the required jar files, libraries etc to successfully build the test environment and to execute the test scripts.

TestNG jar is a Test Framework which allows you to do a lot of things like creating tests, creating configuration methods, running tests, parallel execution, data driven testing, reporting and many more. To use these features you must add a TestNG dependency in maven then this folder will be automatically created.

9. Test Output

TestNG supports default report generation when the user runs testng.xml, from an IDE. By default all reports are generated at the test-output folder. If you want an emailable report then you need to click on emailable-report.html, and if you want an html report then you need to click on index.html.

10. XML Files

10.1 Batch Execution.xml

It's a process of executing all the test scripts or suites in sequential order. There is no need for any manual intervention to run each test suite. When a batch is submitted for execution, a series of commands are sent to the database with respect to the defined component parameters.

10.2 Group Execution.xml

It is a process of grouping different tests together into a straightforward group and running these tests together by just running the group in a single command. And sometimes if you don’t want to execute some test cases then you can group them accordingly. This is done by the include and exclude mechanism that is supported in TestNG.

10.3 Parallel Execution.xml

Parallel testing is a process where multiple tests are executed simultaneously/in parallel in different thread processes. With respect to selenium, it allows you to execute multiple tests on different browsers, devices, environments in parallel and at the same time, instead of running it sequentially.

10.4 pom.xml

A Project Object Model or POM is the fundamental unit of work in maven. It is an XML file that contains information about the projects and configuration details used by Maven to build the project, such as dependencies, build directory, source directory and plugins etc.

10.5 testng.xml

TestNG xml file contains all the test configuration and this xml file can be used to run and organize our test. TestNG eclipse plugin can be used to automatically generate testng.xml file so no need to write from scratch.

11. Drivers

To automate any web application we need a browser and to launch the browser and execute selenium test scripts we need a driver of that particular browser, So we download and store those drivers in the drivers folder.

12. Extent Report

Extent report is an open source reporting library useful for test automation. With this library you can create beautiful, interactive and detailed reports for your test. These reports are HTML documents that depict results as pie charts. We can add screenshots, tags, events, devices, authors in the report.

13. Screenshot

We already created the method for taking the screenshots in our framework, so when at the time of execution any test fails then automatically a screenshot will be taken and then it is stored in this screenshot folder.

14. Priority in TestNG

Priority is an attribute that tells TestNG which order the test needs to follow. When we have multiple test cases and want to execute them in particular order, the TestNG priority attribute helps in executing the test cases in that order.

1. The test cases get executed in ascending order of the priority list. Thus, test cases with lower priority get executed first.

2. One test method is allowed to have only one test priority in TestNG.

3. If test priority is not defined explicitly while running multiple cases, TestNG assigns all cases with a default test priority i.e., zero(0). And Executes test cases by alphabetical order.

The syntax for test priority is @Test(priority = x), where x can be any integer (-), (0), (+).

```java
public class Demo3AddUser {
    static {
        System.setProperty(IConstants.chromeKey, IConstants.chromeValue);
    }

    @Test(priority=-1)
    Run | Debug
    public void addUser() {
        Reporter.log("Successfully added  User----priority=-1",true);
    }
    @Test
    Run | Debug
    public void editUser()
    {
        Reporter.log("Successfully edited  User----priority=non",true);
    }
    @Test(priority=0)
    Run | Debug
    public void deletUser1()
    {
        Reporter.log("Successfully deleted  User----priority=0",true);

    }
    @Test(priority=1)
    Run | Debug
    public void saveUser1()
    {
        Reporter.log("Successfully save  User----priority=1",true);

    }
    @Test(priority=1)
    Run | Debug
    public void checkUser1()
    {
        Reporter.log("Successfully check  User----priority=1",true);
    }

}
```

```
<terminated> Demo3AddUser [TestNG] C:\Users\PrakashSrivastava\.p2\
[RemoteTestNG] detected TestNG version 7.4.0
Successfully added  User----priority=-1
Successfully deleted  User----priority=0
Successfully edited  User----priority=non
Successfully check  User----priority=1
Successfully save  User----priority=1
PASSED: saveUser1
PASSED: editUser
PASSED: checkUser1
PASSED: deletUser1
PASSED: addUser

===============================================
    Default test
    Tests run: 5, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 5, Passes: 5, Failures: 0, Skips: 0
```

15. TestNG Dependent Test

If we want to run our test cases in a particular order in TestNG, then we may use the priority parameter for that, but priority will run all the cases without looking for the relationship between them.

The dependent tests in testNG determine the dependency of a test on a single or group tests. In that case, we say that a test is dependent on another test. For ex. B test is dependent on A, then B test is going to execute only when A test is executed and passed.

```
public class Dependency {

	@Test (dependsOnMethods = { "OpenBrowser" })
	public void SignIn() {
		System.out.println("User has signed in successfully");
	}
	@Test
	public void OpenBrowser() {
		System.out.println("The browser is opened");
	}
	@Test (dependsOnMethods = { "SignIn" })
	public void LogOut() {
		System.out.println("The user logged out successfully");}
	}
```

In the above example OpenBrowser() does not depend on any method so it will execute first then signIn() will execute because it depends on OpenBrowser() then at last LogOut() will execute because it depends on SignIn(). Here if OpenBrowser() is not executed or failed then remaining methods will not be executed because it depends on OpenBrowser.