Name- Sanjeeb Kumar Pati(SDET-21)

Email-srisanjeebpati@gmail.com

Phone- 8249445503

---------- FRAMEWORK ----------

My Framework is a Hybrid Framework which consists of Data driven, Page Object Model and TestNG which reduce the time to automate a testcase.

We know as per the rule of automation Data cannot be hard coded. So we have to take the data from external files. In my framework I was taking the common Data from JSON file and the test Data from Excel Sheets.

I use Page Object Model in my framework which is a concept provided by Google to maintain elements in well Organised and in an encapsulated way so that no one cannot change our web element address and by using POM we can handle the stale element exception.

In my framework i use TestNG to execute the test cases, it has some annotations which help us to execute the test cases in a well organised way and the benefits i get like parallel execution, group execution and cross browsing testing.

My framework consists of 4 parts src/main/java, src/main/resources, src/test/java and src/test/resources.

In src/main/java i created two packages one is Generic Utility and one more is object repository(POM pages)

Generic Utility

In this package i created different classes which have some generic methods for code optimization. The classes are

PathConstants (Interface)

Inside this interface we store paths of all directories. As it is an interface all the variable present inside this are by default public, static, and final so that we can use it inside any class and no one cannot change it as it is final.

JSON_File Utility

Inside this class we write a generic method to read the common data such as URL, USERNAME ,PASSWORD, Etc. Here the data stores in the form of key and value pairs.

ExcelFileUtility

Inside this class we write a generic method to read the test data such as First name, last name, org Name, etc.

DataBaseUtility

Inside this class we write two generic methods to read and write our data to the data base.

JavaUtility

In this class we have to write to two different generic methods one is for random numbers and one is for date and time. In some cases we have to run the test cases for multiple times so if duplicity happens in name at the time of execution we concatenate a random number with the test data. Like that at the time of report generation we have to produce multiple reports so every time the reports generate we concatenate the current date and time with the report file name to avoid confusion.

WebDriverUtility

Inside this class we write some generic methods to perform web driver actions like maximize, implicitlywait, mouse hover, context click, double click, drag and drop, switch windows , switch to frames , etc.

BaseClass

This contains many annotations methods of testNG to control the execution flow of scripts. There are many annotations like:

@BeforeSuits- Inside this annotation we have written a method to create the connection between database.

@BeforeClass- Inside this annotation we have to write a method to lunch the browser.

@BeforeTest- Inside this annotation we have to write a method to open url and login to the application because it is common for all tests.

@AfterTest- Inside this annotation we have to write a method to logout.

@AfterClass- Inside this annotation we have to write a method to close the browser.

@AfterSuits- Inside this annotation we have to write a method to close the database.

Listeners

This class implements Itestlistners interface and we have to override all the abstract methods of this class and give them implementations to capture screenshots of failed test cases and generate reports for every failed, skipped, pass test cases.

RetryListners

This class implements IretryAnalyzer interface and we have to override all the abstract methods of this class and give them implementations to re-execute our failed test cases.

POM Repository (package)

In this package we made separate class for every single WebPage of our application.

Every class have a constructor to initialize the driver and to the driver access of the current class.

We locate the elements useing @FindBy, @FindBys ,@FindAll annotations.

All the webelements are store in a private variable to achieve encapsulation and we generate getters and setters of the variables so that we can give read only access to others of our web element, so that no other can change the address of locators of web elements it can interrupt execution of our scripts.

We write all the business methods inside this class whatever action we want to perform in this page for a single test scenario are written in one business method.

Src/Test/Resources(folder)

This folder contains our json file containing all the common data and excel sheets containing our test datas.

Execute flow of test Scripts

We write our test scripts inside src/test/java folder using @Test Method.

Every test case extends baseclass for execution of @annotation for opening database, opening, browser, login to application, sign out, closing browser , close database.

We drive test data from excel sheets by making object of ExcelFileUtility class.

Then whenever our test scenario is as per that we create object of the required pom page and using that object reference we can execute our business methods in our test scripts.