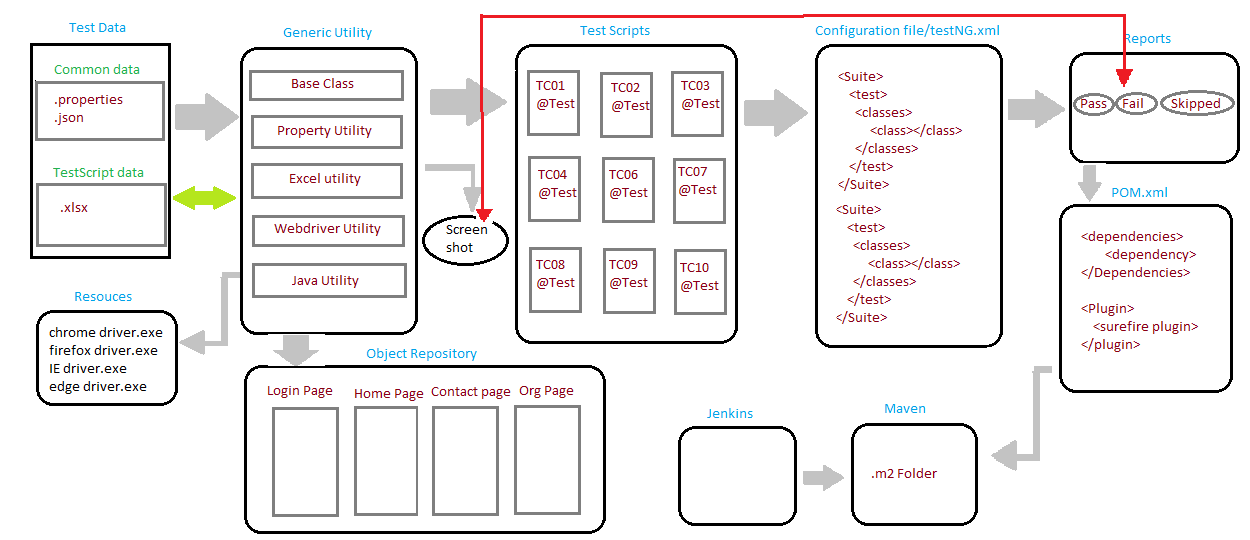
Framework

Framework is a collection of re-usable components and methods that makes automation development, execution and modification faster.

In framework the entire test script execution is taken care with only one driver or configuration file, it is a standard instruction followed by all the organizations.

Architecture of framework



Our framework developers developed my framework using TDD(test driven development). In this method I used several re-usable components to make my work easier.

In the first we developed the test data that to be re-used throughout the test script execution. We categorize the test data into common data and test script data. In common data we used to access the property file or xml file to fetch the data and in general common data is totally common to entire script. e.g. username, password, URL, browser name.

We used to fetch/write the test script data from/to excel file and these data are specific to particular test script. As you se we are using data driven framework concept in our framework.

Then we developed the resources component. In this component we stored all the driver executable file that needs to be used in the entire project like:- chromedriver.exe , firefoxdriver.exe etc.

Then generic utility is developed, which contains all the common classes with reusable method and constant variables those are common to entire framework. Here we implemented method driven framework concept as we developed the methods those are used in the framework.

We developed baseclass here which contains all the configuration annotations to perform all the pre-condition and post-condition.

After that propertyUtility is developed which contains the generic methods to fetch the data from the property file or json file.

excelUtility is then developed which contains reusable method to fetch or write from/to the excel file. We added Apache poi and Apache poi-ooxml dependency to POM.xml in order to work around with excel file.

All the web browser operations related methods are developed and stored inside the webdriverUtility like handling popup, moving mouse cursor, handling drop down box etc.

The methods which need to perform some action using java are stored within javaUtility. For example when I am entering organization name in the organization textbox it can’t take the duplicate value and we need to test for +ve as well as -ve test cases. We can’t always enter the values manually for this instance we need to send the organization name along with some randomly generated number to the textbox in order to avoid duplicate data. So we developed generateRandomNum() within the javaUtility.

To create connection and fetch/write data from/to the database, dbUtility is developed. Through this we can work with the database and this class is built using jdbc.

All the file path/address values that to be used throughout the test script assigned to final variables which are stored within a interface called as constantPath. Using interface we can fetch the values faster then the values stored in the external files.

In case of any failed testcase we designed a class named as listenerImplementation, which provides implementation to the methods of iTestListener interface. To be specific we provided the implementation for the onTestFailure() of the above mentioned interface in our implementation class.

If a testcase failed due to any inconsistent reasons like bad server connection or any synchronization issue, for this case we designed a class called as retryAnalyzerImplementation. Here we can set the no of retry for the test cases by providing the implementation to the retry() method of the iRetryAnalyzer interface.

Then comes objectRepository development part. Here we follow POM(page object model) design pattern and store all the web element locators pagewise/module order, because by the rules of automation we can’t hard code the web element locators to the test scripts.

I also developed some business logics here which contains actions specific to a particular webpage to do some particular actions. As all the elements are stored in a module wise manner, we can call this as a module driven framework.

As you see I have implemented 3 approaches of framework development so I can call my framework as a hybrid framework.

Then the testScript development will takes place as the next component of the framework. Here @test annotated method is used to write each automation test script and as per rule of automation we had extended the base class to each and every test class.

Then in order to do the group, batch or parallel execution of the test scripts we need to convert all the test scripts to .xml file. This component is written using xml language.

After the execution of .xml file we can find the report about the test run inside the test-output folder. Usually we are getting this report in the form of .html file.

There is an important component of my framework is to take the screenshot of the failed test scripts. As I mentioned above this can be done by calling the method listernerImplementation() from the GenericUtility.

That’s conclude about my framework.