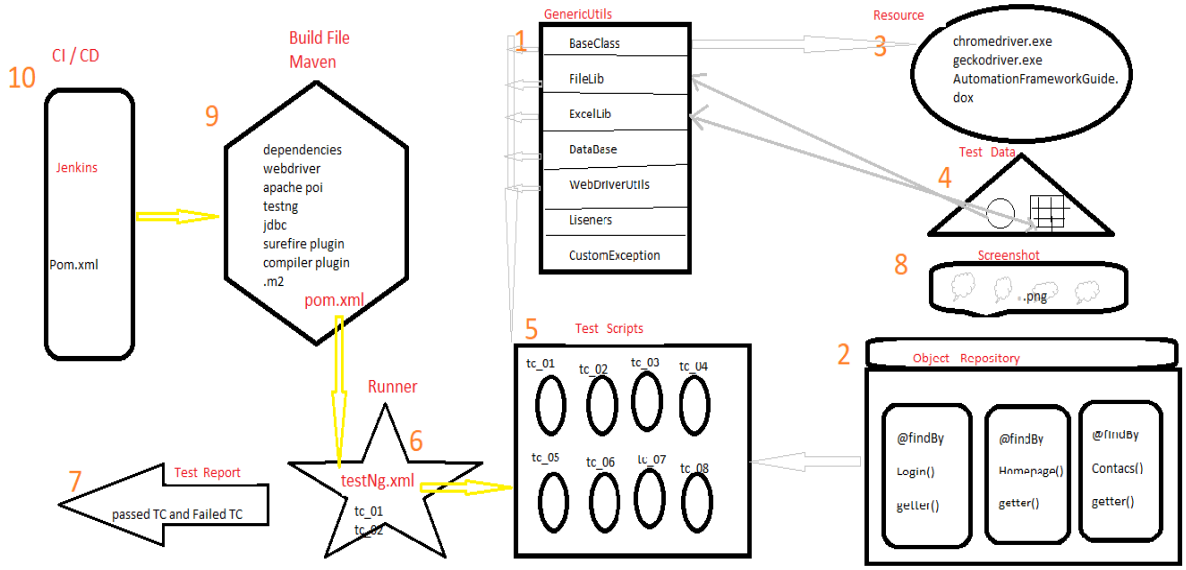

Frame Work Components

FrameWork:

1) Frame work is a well organised structure of components in one place (i.e everything is in separate place like Test Scripts is in one place , Object Repository is in one place, test data is in one place, generic library is in one place) wherein 1 xml file will take care of entire batch execution without any manual interaction is called Frame work.

2) Frame work is a collection of reuseable components that makes Automation development, execution , modification easier and faster.

3) It is a tool created by Frame work developer which contains reuseable components which makes Automation Engineer life easy.

4) Frame work is a process or rules followed by every company to develop test scripts.

My frame work is a Hybrid frame which is a combination of data driver, modular driver, keyword driver.

We have used

- TestNg unit testing frame work tool
- Maven Build testing tool
- Apache poi
- Selenium 3.141.59

My Frame work contains 10 components ,

1. Generic Library
2. Object Repository
3. Resources
4. Test Data
5. Test Scripts
6. Runner / testNg.xlm
7. Reports
8. Screenshots
9. Build File
10. CI / CD

1. **Generic Library**

   It is one of the reusable component in my framework , which contains reuseable classes which can be used for any application.

   Generic Library contains many classes like

   - BaseClass
   - FileLib
   - DataBaseLib
   - ExcelLib
   - WebDriverUtils
   - LisenerImp
   - CustomException

BaseClass

BaseClass contains common testNg annotations which can be used in all the test scripts like

@BeforeSuit

- used to connect with DataBase and to create extend report

@BeforeTest

- used to openBrowser in case of Cross Browser Testing

@BeforeClass

- used to openBrowser in normal execution

@BeforeMethod

- used to Login To Application

@AfterMethod

- used to Logout From Application

@AfterClass

- used to closeBrowser in normal execution

@AfterTest

- used to closeBrowser in case of Cross Browser Testing

@AfterSuit

- used to disconnect from DataBase and to save extend report

Each and every testscript should extends to BaseClass.

FileLib

Basically FileLib is used to read the commonly used data from the properties file. Commonly used data like browser, url, username and password.

If you want to execute 1000 scripts in different browser we will change the browser in properties file.

If you want to run the scripts using different credentials then we are going to change data in properties file.

ExcelLib

ExcelLib is used to read the test data from excel file,

Every test scripts needs test data so we are going to use ExcelLib.

DataBaseLib

InOrder to fetch the data from DataBase we use DataBaseLib.

WebDriverUtils

Here we have general reuseable WebDriver specific methods like

implicitywait
Excplicitlywait
customwait
Alerts
MultipleTab
DropDowns
Frames

ListenerImp

It is implemented using testing, basically used to take screenshot whenever there is fail in test script. So that we can use the screenshot to raise the defect and send for developers.

CustomException

It is used to create some customException assume if there is no data in excel sheet or browser is not opening at that time to display some message we used some customException.

2. Object Repository

Basically Object Repository is also a library which contains

Reuseable WebElements
Business libraries
getters methods

related to specific application only , it is not generic.

Whenever there is change in requirement modification and maintaince of xpath will be easier.

3. Resources

Basically Resources consists of all the resource needed for frame work for example BaseClass needed chromedriver, geckodriver, AutomationFrameworkGuide document so that whenever a new Test Engineer joins he can visit that document to understand the frame work.

BaseClass internally uses resource to launch the browser.

4. **Test Data**

   TestData is one of the component which contains of two types of data. Properties data and ExcelData.

   In Properties data we will keep all the commonly used data for the project. Properties data is lighter and faster compare to ExcelData.

   As per the rule of automation we should not hard code the data so In ExcelData we keep all the test script data -

   For each module we keep separate excel sheet to store test data for that particular module.

   Each row is dedicated for one test script data.

   All the above 4 components are developed by the Frame work developer.

   Once it is ready TestEnginner will create test scripts.

5. **Test Scripts**

   We have to create separate package for all the modules. And inside a specific module package itself we have to create all the test scripts related to that module, this is called as modular driver.

   Using @Test annotation, existing frame work libraries like generic library and object repository library we are going to create test scripts.

   Once test scripts are developed we are going to commit to the git hub.

   Every test script is maintained in modular wise so that we can do regression, smoke testing easily.

6. **Runner / testNg.xlm**

   Basically Runner is a testing.xml file, through which we can do batch execution, group execution, parallel execution.

   Whenever new build comes we use testing.xml to do Smoke testing and all.

7. **Reports**

   When execution is done it will automatically generates the test report, and we will get to know the stability of the application that
   how many testcases are pass ,
   how many testcases are fail ,
   how many testcases are skipped ,
   how long does it takes for the execution
   emailable report.
   If any testcase fails that will be taken screenshot and stored in screenshot folder.
   Sometimes we create customised extend report.

8. **Screenshots**

   Whenever there is fail in test case automatically LisenerImpliment class will take screenshot and store it under Screenshots folder, and we can use it to raise the defect.

9. **BuildFile**

   Entire Project is implement using maven, because maven has a feature called as pom.xml.
   Using maven
   we can handle the dependency jar files,
   we can invoke testing.xml file
   we can execute in command line
   also we can execute in Jenkins

   We used Maven for the creation of the build.

10. **CI / CD**

    Maven can be executed through CI / CD pipeline.