Sample Framework

**Architecture**

* **Project\_Folder**
  + **src**
    - **page\_Objects**
      * **Login\_Page.java**
      * **Home\_Page.java**
      * **Registration\_Page.java**
    - **parallel**
      * **TestNGCreateXML.java**
    - **report**
      * **ExtentManager.java**
      * **ExtentTestManager.java**
    - **utils**
      * **BackupResultFiles.java**
      * **BrowserUpdate.java**
      * **Constants.java**
      * **CreateBatFile.java**
      * **ReadExcel.java**
      * **ScriptDetails.java**
      * **ZipResultFiles.java**
    - **wrappers**
      * **ApplicationWrapper.java**
      * **Components.java**
      * **FunctionLibrary.java**
      * **RobotKeyboard.java**
  + **JRE System Library**
    - * **Java .jar files**
  + **Referenced Libraries**
    - * **External .jar files (like testing.jar, etc.)**
  + **Batch\_Scripts**
    - * **runSuite.bat**
      * **testing.xml**
  + **BrowserDrivers**
    - * **Chromedriver.exe**
      * **Geckodriver.exe**
      * **IEDriverServer.exe**
  + **Config**
    - * **browser.properties**
      * **Env\_AUTO.properties**
      * **Env\_DEV.properties**
      * **Env\_FUNC.properties**
      * **extent-config**
      * **GlobalAddressFinderValues.properties**
      * **log4j.properties**
      * **proguard.pro**
      * **Runtime.properties**
      * **SystemConfig.properties**
  + **Lib**
    - * **Jar files like extentreports-3.0.1 etc.**
  + **Reports** 
    - **Test\_Reports**
      * **CRM\_Regression\_Report.xls**
    - **Test\_Results**
      * **\_Screenshots**
        + **12\_09\_2018\_07\_33\_15\_PM.jpg**
        + **12\_09\_2018\_07\_33\_15\_PM.jpg**
      * **TestReport\_CHROME\_\_14\_09\_2018\_144105\_1.html**
      * **TestReport\_CHROME\_\_14\_09\_2018\_144105\_1.html**
    - **Applog.log**
  + **Resources**
    - **Test\_Batch**
      * **Test\_Batch\_Config.xls**
      * **Test\_Batch\_Parallel\_1.xls**
      * **Test\_Batch\_Serial\_1.xls**
    - **Test\_Data**
      * **FRA\_Registration\_001**
      * **FRA\_Registration\_002**
    - **Browser\_Exec\_Order.xls**
  + **Support\_Files**
    - **10MB file.jpg**
  + **test-output**
    - * **testng-results.xml**
  + **TestNGXMLs**
    - **Test\_Batch\_Parallel\_1\_No1to1 .xml**
    - **Test\_Batch\_Serial\_1\_No1to1.xml**
  + **build.xml**
  + **create\_xml\_bat.xml**

**Types of Test Automation Frameworks**

* Linear Scripting
* Modular
* Data Driven
* Keyword Driven
* Hybrid
* Behavior Driven Development

**What is a framework?**

* It defines a set of rules or best practices to achieve the desired results in systematic way.
* Automation frameworks deal with best practices to achieve the goals of our automation project.

### **Linear Scripting Framework:**

* Record and Playback
* Basic level test automation framework.
* Used to test small sized applications.
* This also known as ‘Record and Playback’ framework
* In this type, creation, and execution of test script are done individually for each test case individually.
* Merits:
  + We can generate test scripts (Record and playback) without planning much or consume much time.
* Demerits:
  + lack of reusability and hard coding the data does not allow to run with multiple data sets

**Modular Testing Framework:**

* Divide the application into multiple modules and create test scripts individually.
* These individual test scripts can be combined to make larger test scripts by using a master script to achieve required scenarios.
* This master script is used to invoke the individual modules to run end to end test scenarios.
* In this framework, testers write function libraries to use it whenever it required.
* This is AKA modularity framework or module-based framework.

**Data-driven Framework:**

* Is focused on separating the test scripts logic and the test data from each other.
* Allows us to create test automation scripts by passing different sets of test data.
* The test data set is kept in the external files or resources such as MS Excel Sheets, MS Access Tables, SQL Database, XML files etc.,
* The test scripts connect to the external resources to get the test data.
* By using this framework we could easily make the test scripts work properly for different sets of test data.
* This framework significantly reduces the number of test scripts compared to module based framework.
* Merits:
  + This framework gives more test coverage with reusable tests and flexibility in execution of tests only when required and by changing only the input test data and reliable in terms of no impact on tests by changing the test data
* Demerits:
  + this framework needs to have hands-on programming knowledge to develop test scripts

**Keyword Driven Testing Framework:**

- It is also known as table-driven testing or action word based testing.

- We use a table format to define keywords or action words for each function or method that we would execute.

- It performs automation test scripts based on the keywords specified in the excel sheet.

- By using this Framework, testers can work with keywords to develop any test automation script, testers with less programming knowledge would also be able to work on the test scripts.

- Logic to read keywords and call the required action mentioned in the external excel sheet is placed in the main class.

- Keyword-driven testing is similar to data-driven testing.

- Even though to work on this framework doesn’t require much programming skills but the initial setup (implement the framework) requires more expertise.

**Hybrid Driven Testing Framework:**

* Combination of two or more frameworks.
* Leverage the strengths and benefits of other frameworks for the particular test environment it manages.
* Most of the teams are building this hybrid driven framework in the current market.

**Behavior Driven Development Testing Framework:**

* To create a platform which allows everyone (such as Business Analysts, Developers, Testers etc.,) to participate actively.
* It requires increased collaboration between Development and Test Teams.
* It doesn’t require the users to be acquainted with a programming language.
* We use non-technical, natural language to create test specifications.
* Some of the tools available in the market for Behavior Driven Development is [JBehave](http://jbehave.org/), [Cucumber](https://cucumber.io/), etc.,

**Our Project framework:**

* We are following Hybrid Testing Framework.
* It is combination of Modular Testing Framework & Keyword Driven Testing Framework.
* Simply say in modular testing framework we implemented the Keyword driven testing framework.

Notes-

* Testcasename and testdata name are same(ex- DO\_ONLINE\_TOPUP\_001 )

Beforetest: To Create Test Node in the Report and to kill the browser

Test\_Batch\_Serial\_1.xls

(Test Suits)

BeforeSuit: To Create Report at the Start of any TestNG Suite and kill the browser drivers and Opened browsers

Aftersuit: End the Test Node in the Report.

This Create/finish the report for the currently executed TestNG XML Suite

Once execution is completed that the report is generated in Test\_Reports folder, html result files are generated in Test\_Results folder and screenshots are saved in \_Screenshots folder.

In FunctionLibrary.java it contains low level functions. With the help of properties files, pageobjects, testdata(excel sheet contains test data for each step) it will execute the corresponding function.

In TestNGCreateXML.java by using DOM parser that the testNG.xml files are created in TestNGXMLs folder. (Ex- Test\_Batch\_Serial\_1\_No1to1.xml)

Based upon the step it directly navigate to Components.java and call that particular method.

In Components.java it contains high level components like Login(), Registration() etc. It calls the particular function directly to FunctionLibrary.java

In excel (GBR\_SUBS\_FULLVIEW\_4393.xls) it is clearly listed the steps (Execution\_Order) in first sheet and the remaining sheets contain inputs for that steps.

In testing.xml file that the testcase name and its location are mentioned.

(<parameter name="testcaseName" value="GBR\_FULLVIEW\_4393"/>

<parameter name="testdatasheet" value="./Resources/Test\_Data/GBR\_SUBS\_FULLVIEW\_4393.xls"/>)

)

Aftertest: To End the Test Node in the Report.

* also monitors the Report file size and when it crosses the configred size then New Report will be created

Also for Parallel Suite type Thread Catch concept is included which is handles by Countdown latch and thread.await concept

**Detailed in Framework**

* **Project\_Folder**
  + **src**
    - **page\_Objects** 
      * + Split wise pages and list out all the elements of the page using **HashTable**
      * **Login\_Page.java**
      * **Home\_Page.java**
      * **Registration\_Page.java**
    - **parallel**
      * **TestNGCreateXML.java** 
        + Used to Create Every TestNG XML files and Bat files to Start the **Batch Execution**
        + To Create TestNG XMLs from the Parallel and Serial Test Batch sheets placed in the TestBatch folder for both Parallel and Serial Spreadsheets
        + **Note:** Multiple TestBatch sheets are supported. Multi TestNG files will be created based on the configured split count.
        + To Create Config TestNG XML file from the Test\_Batch\_Config.xls placed in the TestBatch folder
        + **Note**: No Multiple TestBatch Spreadsheets will supported and No Multi TestNG files will be created. Only single file.
        + Objective : To Create Full Suite TestNG XML with the generated Config, Parallel and Serial testNG xml file names
        + A TestNG File Generated by this method is the full suite to get executed.
        + **Note**: All Multiple TestBatch XMLs and Multi TestNG XML file names will be consolidated and created here in this XML file
        + Objective : To Create Browser Update TestNG XML based on the BrowserName and Description
        + A TestNG File Generated by this method will Execute the Browser Update Script in utils package based on the given Input
        + Objective : To Create Final TestNG XML with the BrowserUpdate Script and Full Suite one after the another
        + A .bat File Generated by this method is the full suite to get executed with multiple browser.
        + Objective : To Create Final TestNG XML with the BrowserUpdate Script and instead Full Suite xml
        + we specify all the TestNG xmls in TestNG file seperately one after the another
        + A .bat File Generated by this method is also the full suite to get executed with multiple browser.
        + Executing this .bat file will reduce the long continuous execution of same single TestNG thread for all Batch
        + Objective : To Create Backup Result Files TestNG XML
        + A TestNG File Generated by this method will Execute the Backup Result File class in utils package
        + Objective : To Create Zip Result Files TestNG XML
        + A TestNG File Generated by this method will Execute the Zip Result File class in utils package
    - **report**
      * **ExtentManager.java**
        + Main Class to Create Extent Reports for Parallel Execution
      * **ExtentTestManager.java**
        + Class has all the functions to write and create a Report for Parallel Run
    - **utils**
      * **BackupResultFiles.java**
        + Class will copy only the HTML files from Results folder and store them in a temporary folder HTML\_Files
        + All the HTML files which were copied in HTML\_Files will be zipped and stored in Results\_Zipped folder as Results.zip
      * **BrowserUpdate.java**
        + Objective : To Update the Browser name in Browser Properties files based on the TestNG file and its Parameters executing this class
        + Objective : Update the Name of the Browser in browser.properties file
      * **Constants.java**
        + All Variables which remains constant in Excel E.g. Header Rows and SheetName
      * **CreateBatFile.java**
        + Used to Create .bat files based on the TestNG Creation class
        + To only Create .bat file with the given filename
        + To write into the .bat file with the given filename
      * **ReadExcel.java**
        + All Functions to read and write in a Excel
      * **ScriptDetails.java**
        + This class is used to Create HTML Overview Excel Report and also updated the Script details tracker which has to be placed in the desired path.

Following 2 actions will be generated by this Script.

1. HTMLOverView Report Generation

2. Script details Tracker updation.

* + - * + HTMLOverView Report Generation

This is an Excel Report generated based on the HTML Report available in the Given Path

* + - * + Script details Tracker updation

Script details should be placed in the Path configured in the System Config Property file, if placed then the tracker will be updated

If not placed only HTML Overview report will only get generated.

* + - * + On completion of each testNG.xml file, this script will be triggered.
      * **ZipResultFiles.java**
        + Class will copy only the HTML files from Results folder and store them in a temporary folder HTML\_Files
        + All the HTML files which were copied in HTML\_Files will be zipped and stored in Results\_Zipped folder as Results.zip
    - **wrappers**
      * **ApplicationWrapper.java**
        + Class holds all the TestNG Annotation for Creating Report
        + To Kill the browsers based on the Browser Name in Browser.Properties file
        + Note: Based on the killBrowser property value the Browser will be killed
        + To Kill the Plugin container displayed during the Firefox browser instantiate
        + To Kill the browsers drivers based on the Browser Name in Browser.Properties file
        + Note: Browser drivers will be mandatorily killed when this function is triggered
        + To Reduce the HTML Report file size by removing the New lines, Tabs and continuous spaces
        + To Create Report with the Given Report file name and to Initiate the Latchs based on Suite Type
        + Note: Browser drivers will be mandatorily killed when this function is triggered
        + **beforeSuite**- To Create Report at the Start of any TestNG Suite and kill the browser drivers and Opened browsers
        + **beforeTest**- To Create Test Node in the Report and to kill the browser/browserdrivers
        + **afterTest**- To End the Test Node in the Report

This method also monitors the Report file size and when it crosses the configured size then New Report will be created

Also for Parallel Suite type Thread Catch concept is included which is handles by Countdown latch and thread.await concept

* + - * + **afterSuite**- To End the Test Node in the Report

Note: This Create/finish the report for the currently executed TestNG XML Suite

* + - * + To Create the Report file
        + Note: This method will be executed only by one thread(So one report will be generated and used by other threads also)
        + in case of multiple threads alive
      * **Components.java**
        + Class holds all components
      * **FunctionLibrary.java**
        + Class holds all Low Level Functions
      * **RobotKeyboard.java**
        + For keyboardEvents
  + **JRE System Library**
    - * **Java .jar files**
  + **Referenced Libraries**
    - * **External .jar files (like testing.jar, etc.)**
  + **Batch\_Scripts**
    - * **runSuite.bat**
        + to start execution
      * **testNG.xml**
        + testNG template to create testNG during runtime.
  + **BrowserDrivers**
    - * **Chromedriver.exe**
      * **Geckodriver.exe**
      * **IEDriverServer.exe**
  + **Config**
    - * **browser.properties**
        + Browser details
      * **Env\_AUTO.properties**
        + Environmental details
      * **Env\_DEV.properties**
        + Environmental details
      * **Env\_FUNC.properties**
        + Environmental details
      * **extent-config**
        + Automation report config
      * **GlobalAddressFinderValues.properties**
        + Sample address for GAF
      * **log4j.properties**
        + log4j config
      * **proguard.pro**
      * **Runtime.properties**
        + To reuse the value during runtime
      * **SystemConfig.properties**
        + Configs related to only to framework
  + **Lib**
    - * **Jar files like extentreports-3.0.1 etc.**
  + **Reports** 
    - **Test\_Reports**
      * **CRM\_Regression\_Report.xls**
    - **Test\_Results**
      * **\_Screenshots**
        + **12\_09\_2018\_07\_33\_15\_PM.jpg**
        + **12\_09\_2018\_07\_33\_15\_PM.jpg**
      * **TestReport\_CHROME\_\_14\_09\_2018\_144105\_1.html**
      * **TestReport\_CHROME\_\_14\_09\_2018\_144105\_1.html**
    - **Applog.log**
  + **Resources**
    - **Test\_Batch**
      * **Test\_Batch\_Config.xls**
      * **Test\_Batch\_Parallel\_1.xls**
      * **Test\_Batch\_Serial\_1.xls**
    - **Test\_Data**
      * **FRA\_Registration\_001**
      * **FRA\_Registration\_002**
    - **Browser\_Exec\_Order.xls**
  + **Support\_Files**
    - **10MB file.jpg**
  + **test-output**
    - * **testng-results.xml**
  + **TestNGXMLs**
    - **Test\_Batch\_Parallel\_1\_No1to1 .xml**
    - **Test\_Batch\_Serial\_1\_No1to1.xml**
  + **build.xml**
  + **create\_xml\_bat.xml**

**Framework layout:**

* 1. By using Page Object model, first page wise we store all elements with the help of Hash Table in key value pair.
  2. Or else we use Page Repository by using return type methods.
  3. Config files are stored as Properties files- Constants values are stored in each property file.
     + Like for Browser property, User details, environmental, Log4j, extent report etc
  4. Wrapper class: Component & Function
     + Components: It contains Actual Test case method.
     + Functions: It contains all functions (methods) of Component class. And these methods will call from Component class.
     + Using Modular test framework, it divides the application into multiple modules and create test scripts individually.
  5. Utils: Which contain third party related information
     + Actual Log4j, ReadExcel, AutoIT.
     + This all utils are united by functions and these functions will call by components.
  6. Batch script file:
     + Apart from testing team automation test cases may be run by anyone even Dev team too. So to make it simple we maintain Batch script file which contains all Test case ID’s.
     + What are the test cases we need to run that only marked ‘Yes’ in Batch file for execution.
  7. TestNG:
     + Using create\_xml\_bat.xml that the testNG.xml file is generated in TestNG folder for marked cases during run-time.
     + In Batch Script folder we have runSuite.bat

it will start the execution from already generated testing.xml file.

* + - In generated testNG.xml file contains Parallel or Sequence, Testcase sheet name, TestcaseID, and its path etc.
  1. Test case file:
     + In Test case excel file that the first sheet (Controller Sheet) contains step by step execution using keyword driven framework.
     + Based on the specified keyword it performs automation test scripts.
     + With help of specified keyword, it will navigate to component class for each and every step execution.
     + Logic to read the keywords and call the required action mentioned in the external excel sheet is placed in the component class.
     + Each sheet contains test data for each step.
  2. MAVEN
     + Maintained common structure across all the products.
     + In pom.xml (Project Object Model) we add the below:
       - Jar related info in dependency
       - Added SureFire plugins to execute all our test cases under maven projects.
       - Added TestNG info into plugins section.
     + Use the below cmds in Maven projects:
       - Clean (mvn clean - >to clean the previous build before running execution)
       - Compile (mvn compile -> to know any syntax error)
       - Test (mvn test- > to run a test)
         * By default it will run clean & compile
       - Mvn install / mvn package (to download the dependencies)
     + PROFILES (to run different testNG.xml files for Smoke, Regression)
       - Based upon the profile ID it should trigger the corresponding suiteXMLfile.
         * Cmd (mvn test –PRegression)

Regression suite’s testNG.xml file executed

* 1. Jenkins
     + For CI (Continuous Integration) and CD (Continuous Deployment)
     + Using CI we can easily find out the code merge issues with help of build file.
     + Deploy the latest bundle using CD.
  2. Extent Report
  3. Log4j