**Library**  - is a collection of functions which can be added in our application to serve specific purpose. You call Library.

**Framework** - is skeleton which provides basic structure to develop an application/ a set of programs.Framework drive our code. Framework will call you.



**Types of Test Automation Framework**

**Linear Scripting Framework** - its a basic test automation framework. It uses ‘record and play back’ to generate automatic scripts. Individual test scripts will be created for individual test-cases.

**Modular Testing Framework** - testers 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 the required scenarios. This master script is used to invoke the individual modules to run end to end test scenarios.

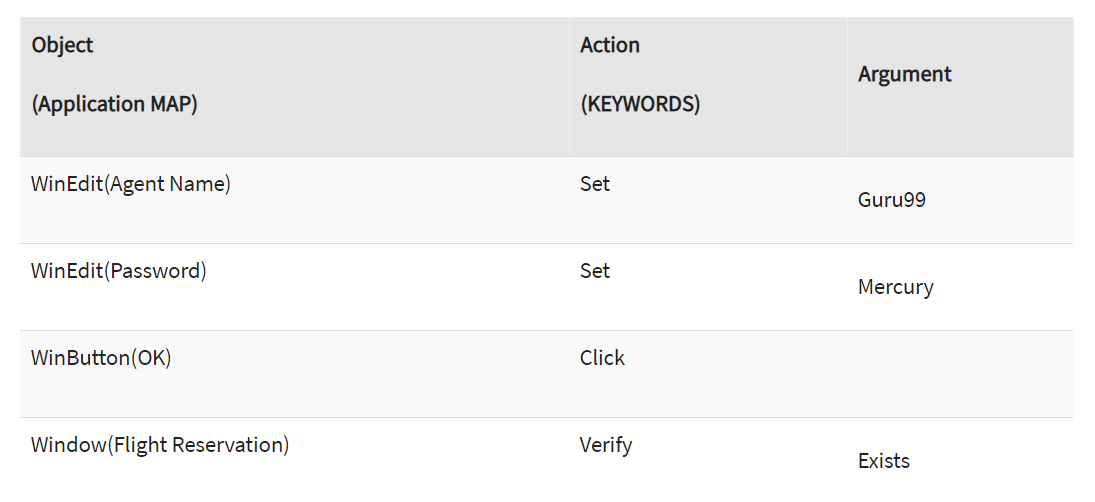
**Library Architecture Testing Framework** - In the modular testing framework, we divide the application under test into modules whereas here we identify the common tasks and grouped them into functions. Once the functions are grouped then these groups will be kept in a library. The test scripts reuse these libraries to create new test cases.

**Data-driven Framework** - The data-driven test automation framework is focused on separating the test scripts logic and the test data from each other. Data for the test scripts will be given dynamically during runtime by fetching data from external source like excel sheet or directly from DB.

**Keyword Driven Testing Framework** - In Keyword-driven 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.

In the keyword approach, each action is driven by three things:

1. Keyword 2. Data (optional) 3. Object



**Behavior Driven Development Testing Framework** - uses non-technical, natural language to create test specifications.

**Hybrid Driven Testing Framework** - Hybrid Test automation framework is the combination of two or more frameworks mentioned above. Mostly the hybrid framework is a combination of both Data-driven and Keyword driven frameworks.

**OOPS in Automation Framework**

**ABSTRACTION -** In Page Object Model design pattern, we write locators (such as id, name, xpath etc.,) in a Page Class. We utilize these locators in tests but we can’t see these locators in the tests. Literally we hide the locators from the tests.

**INTERFACE -** WebDriver driver = new ChromeDriver();

In these webdriver is a interface, we accessing the WebDriver methods using object reference called driver.

**INHERITANCE -** we will be initializing the WebDriver in a Class (Baseclass or BaseTest), and we will use that initialized driver in out test classes by extending the class in which driver is initialized.

**POLYMORPHISM :**

**METHOD OVERLOADING -** driver.switchTo().frame(index or webelement or name) . Depend on the type of argument frame method will be executed.

**METHOD OVERRIDING -** in Extent report, ITestListener interface will be extended and its methods (onTestStart, onTestPass) will be overridden for our own customized implementation of those methods.

**ENCAPSULATION -** Implementation of Page object model (here private variables will be created and will be accessed using public methods)

**Explaining Framework**

**Language :** For scripting we are using JAVA **(java 8 and IDE oxygen 3.7.4)**

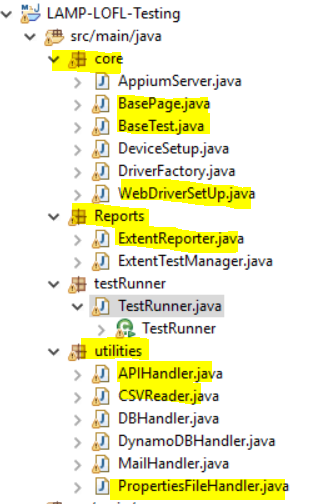
**Type of Testing Framework :** Data driven Framework - we using CVS file for data input.

**Design Pattern :** Page Object Model. Will be maintaining separate class for each web page. The class will contain Locators of that page and methods related to that page.

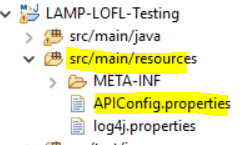
**Unit Testing Framework :** testNG **(7.3.0)** , it helps us to create and run over tests in an organized manner.

**Build Tool:** Maven, best way to build and organize a Project.

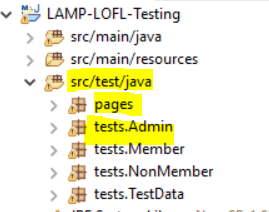
**src/main/java - packages**



**src/main/resources**



**src/test/java**



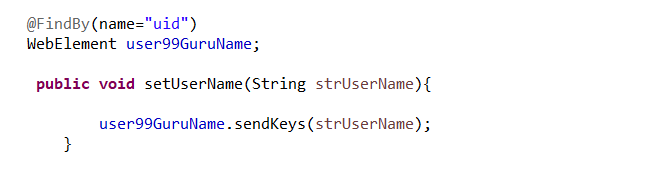
**Report:** Extent report.(2.41.2)

**Version control:** We use Git as a repository to store our test scripts.

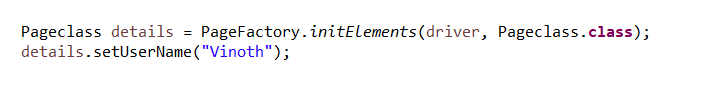
**Runner:** we use Jenkins for CICD. Scripts will be executed for every deployment on lower environments and test results will be shared across team.

**Page Factory -** is one way of implementing the Page Object Model and is an inbuilt Page Object Model framework concept for Selenium WebDriver.

Page Class -



Test class -



**Logger - log4j**

**Logs:** capturing info/activities at the time of program execution.

Types of Logs: info, warn, error, fatal.

We can generate logs using **Apache log4j framework**. It generates log based on the configuration mention in the log4j.properties file

**log4j has 3 major components:**

**Loggers** – It is used for logging information on different level like info, warn, error, fatal.

**Appenders** – In log4j, an output destination is called an appender.

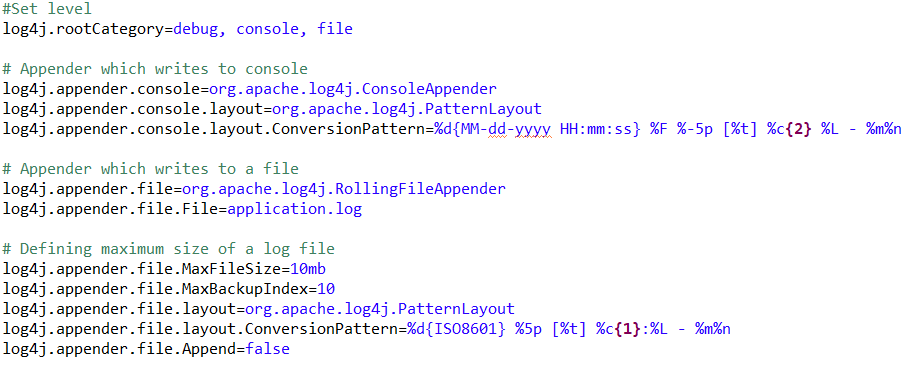
**ConsoleAppender** – It logs to some standard output.

**File appender** – It prints logs to some file at a particular destination.

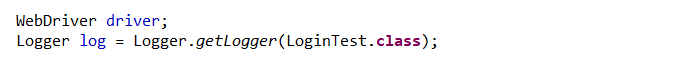
**Rolling file appender** – It is used to for a log file with maximum size.

**Layouts** – It is used to format the logging information in different style.

**Log4j.properties file configuration** (need to create under src/main/resources)



**We can also generate custom logs using below line,**



**Maven -** Maven is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation.

**Maven build cycle :** set of tasks that are executed when maven build is run

clean: handles project cleaning, cleans the previous build

Build: handles project build and deployment

site: handles the creation of project site documentation

**Build Phase:** It represents a stage in Maven build lifecycle

**Validate** - would check if our project is correct and all the required information available

**Compile** - compile the source code.Class file will be created inside the target folder.

**Test** - run unit tests

**Package** - would verify if the package is valid

**Verify** - would verify if the package is valid

**Install** - would locally install the package.It will create executable jar file under target folder

**Deploy** - would be used in integration or release environment by copying the final project into a remote repository where it can be accessed by other projects or developers

**Clean -** cleans up previous build artifacts.Target folder will be deleted.

**Site -** creates site documentation for the project

**Goals :** Maven goals represent a specific task that contributes to the building and managing of a project.

compiler:compile

Compiler:test compile

Jar:jar

Install:Install

Deploy:Deploy

**Major Goals used in Test Automation -**clean,install,test

**Plugins:** A plugin generally provides a set of goals

**failsafe plugin** have integration-test and verify goal

**Surefire plugin** - helps Maven to identify the tests

**Maven Repository:** a directory of packaged jars. Maven searches for dependencies in the repositories.

**Local Repository -** located in our local system. It is created by the maven when you run any maven command.

**Central Repository -** Maven central repository is located on the web. It has been created by the apache maven community itself.

**Remote Repository -** which is developer's own custom repository containing required libraries. It also located on the web.

**POM - Project Object Model -** contains information of project and configuration information for the maven to build the project such as dependencies, build directory, source directory, test source directory, plugin, goals.

**Elements of maven pom.xml file**

Project- it is the root tag of pom.xml

Model version - sub tag of project and it should be 4.0.0

Group ID - it is the Project’s group name

Artifact ID - it is the name of the Project

Version - it is the version of the project.

**Artifact -** is a file usually a JAR type that gets deployed to a maven repository. It has group id, artifact id, version. These 3 uniquely identifies the artifact.

**Elements of Dependencies**

Dependency tag

Group ID - it is the Project’s group name

Artifact ID - it is the name of the Project

Version - it is the version of the project.

**Maven commands,**

**mvn clean** - it will delete the previous build(deletes the target folder)

**mvn package -**  it will build and run the tests (creates jar in target folder)

**mvn install** - it will build and run the tests (creates jar in target folder) additionally it will install the package into the local repository, for use as a dependency in other projects locally

**mvn test** - it will only run the test (but jar file will not be created)

**mvn clean install**

**mvn install -DskipTests : It will skip running the test but build will be generated (jar will be created)**

**To execute the created Jar file with arguments in cmd**

java -jar projectName.jar argument1 argument2

**To do Maven commands in CMD, following things need to be added as variables in user/system**

MAVEN\_HOME - C:\Program Files\apache-maven-3.8.1

M2\_HOME - C:\Program Files\apache-maven-3.8.1

Path - C:\Program Files\apache-maven-3.8.1\**bin**

