**TestNG Framework: -** It is an open source automation testing framework where NG means Next Generation. It is designed to cover all categories of test like Unit, Functional, Integrational etc. It is similar to Junit but not a Junit extension.

1. It has ability to produce HTML reports.
2. Annotations makes the method easier.
3. Grouping & Prioritization of tests can be done.
4. Parallel execution of test is possible.

**@BeforeSuite: -** It will run before all tests inside test suite.

**@BeforeTest:** - It will run before any test method belonging to classes inside <test> tag.

**@BeforeClass:** - It will run before 1st test method in current class.

**@BeforeMethod:** - It will run before each test method.

**@Test:** - It is an actual test method.

**@AfterMethod:** - It will run after each test method.

**@AfterClass:** - It will run after all test method in current class has been run.

**@AfterTest:** - It will run after all test methods belonging to the class inside <test> tag.

**@AfterSuite:** - It will run after all test in suite have run.

**Execution Flow Is: -**

**BeforeSuite -> BeforeTest -> BeforeClass -> BeforeMethod -> Test -> AfterMethod -> AfterClass -> AfterTest -> AfterSuite**

**@Parameter: -** To fetch parameters from testing.xml file.

**@Optional:** - Default value if no parameter has been passed from testing.xml file.

**@DataProvider:** - It will supply data to a test method. It must return an Object [m][n] where ‘m’ stands for the number of set of data and ‘n’ stands for the no. of. Parameters to be passed.

**Testing.xml file will be as: -**

**<suite name = “Test Methods”>**

**<Test name = “Testing Fundamentals”>**

**<classes>**

**<class name = “org.qa.test.LoginTest”/>**

**</classes>**

**</Test>**

**</Suite>**

**Test File will be: -**

**@BeforeClass**

**@Parameters ({“browser”,”url”})**

**Public void setData (@Optional (“Firefox”) String browser, String url)**

**{**

**System.out.println (browser +” “+url);**

**}**

**<suite name = “Test Methods”>**

**<parameter name = “browser” value=”chrome>/>**

**<parameter name = “url” value=”**[**https://www.google.com**](https://www.google.com)**”/>**

**<Test name = “Testing Fundamentals”>**

**<classes>**

**<class name = “org.qa.test.LoginTest”/>**

**</classes>**

**</Test>**

**</Suite>**

**@DataProvider** is used to run the test method multiple times with different sets of data.

@Test (dataProvider = “credentials”)

Public void setData (String username, String password)

{

System.out.println (username +” “+password);

}

@DataProvider (name=”credentials”)

Public Object [][] getData ()

{

Object[] [] data = new Object [1][2] ; // [1] stands for Set of Data and [2] for No. Of. Parameters

Data [0][0] = “Akshay”

Data [0][1] = “Shete”

Return data;

}

**Attributes of TestNG: -**

1. **dependsOnMethods: -** It executes a method based on its dependant methods.
2. **Prority:** - To set priority to a test method. Lower priorities executed 1st.
3. **Enables:** - To run test method value must be true.
4. **Groups: -** It clubs several test methods under a common name.

**Assertions** are checkpoints placed in test script to verify whether an expected condition occurred or not.

1. **Hard Assert: -** It stops execution when assertion fails.

**Assert.assertEquals(“a”,”b”);**

**Assert.assertTrue(<Boolean>);**

1. **Soft Assert: -** It will continue execution even if assertion fails.

**SoftAssert s = new SoftAssert ();**

**s.assertTrue (<Boolean>);**

For soft assert called **assertAll ()** after all the assertions are done. This method collects all the failures and decides whether to fail the test or not at the end. Invoke **assertAll ()** inside every test method.

**Run testing xml from Command Prompt: -**

1. Goto project root directory and create a folder **lib** and place the **testNG.jar** file inside it.
2. Open command prompt and navigate to project root directory using **CD.** For Ex: -

**CD D:\solution\test\src**

1. Use command **javac –cp .\lib\\*; D:\solution\test\src\demoes\demo.java**
2. **Java –cp .\lib\\*; org.testng.TestNG D:\solution\src\testng.xml**

Observe the **test-output** folder created and the folder will contains index.html file as reports.

**Run testng.xml file using .bat file:** -

**Consider filename as:** - dataprovider.bat

Write below code in either Notepad or Notepad++

Set projectLocation = D:\solution\test\src\

Cd %projectLocation%

Set classPath = %projectLocation%\lib\\*

Java org.testng.TestNG %projectLocation%\testng.xml

Pause

**Save the file and double-click it for execution**

**Handling SSL Certificate: -** SSL stands for Secure Socket Layer. It is standard security protocol which is used to establish secured connection between Server & Client. It keeps sensitive information encrypted such as Username, Password, Card information etc. If the information is not encrypted, then anyone can access the private information.

1. Browser request for Secure Socket Layer.
2. Server responds with SSL certificate.
3. Session key is encrypted with SSL public key.
4. Server indicated all the transmission are encrypted.

**Page Object Model (POM): -** It is a design pattern to create object repository for web UI elements. Overtime the amount of test scripts involved in the test repository will increase. Then maintaining huge test script of selenium become tedious when element is updated in the AUT. So, the object repository ensures that the attribute value has to be changed only in Page Class & all the changes will be reflected in test scripts. Code becomes less and optimized because of the reusable page methods in POM.

**By userId = By.id(“username”);**

**Driver.findElement(userId).sendKeys (“Test”);**

**The POM contains following:** -

1. Variables storing web elements attribute type and value.
2. Methods for various interaction with web page.

**The Base Class contains following:** -

1. Method to initialize web driver instance.
2. Method to terminate web driver instance.

**All pre-requisites are created here such as WebDriver initialization/termination, maximize, web driver wait, load properties file etc.**

**Base Class - III**

**Base Class**

**.properties file - IV**

**All page layer classes extend base calass**

**co**

**Config.properties file  
where we define url, uname, password, environment variables etc.**

**All test layer classes extend base calass**

**Page Layer- I Test Layer - II**

**Create separate java class for each page**

**Test Data file - V**

**LoginPageTest**

**Login Page**

**Test Data in excel or some other file**

**Home Page**

**HomePageTest**

**Common Utilities - VI**

**Some common utilities like Screenshot, Recording etc.**

**SearchPageTest**

**Search Page**

**ContactPageTest**

**Contact Page**

**TestNg classes, annotations from where we call methods of each page defined in page layer.**

**Each page must return Reporting Section - VII  
another page as it is Page  
Object Model. We define   
WebObjects/WebElements   
for each page as well as actions  
or methods for each page.**

**Reporting Section**

**Log4j: -** Apache log4j is a java based reliable, fast and flexible logging utility which is used for logging information.

1. It is an open source and can be easily configured in selenium.
2. Using log4j we can store selenium project execution flow in a file.
3. It gives better status of project.

**Components of Log4J: -**

1. **Loggers: -** It is responsible for logging information provide by test script creator. There are 5 kinds of log level such as **Fatal/Error/Warn/Info/Debug.**
2. **Appendars: -** It is responsible for writing log messages to a file. We can use ConsoleAppender/FileAppender.
3. **Layout: -** It is responsible for formatting logging information in different style.

The default pattern will be **%r [%t] %-5P %c - %m%n**

The root logger is set to **DEBUG** level.

We can use log4j by 3 different ways: -

1. Using **Logger** class and by calling **BasicConfigurator.configure ();**
2. By using **.properties** file (The root level has been set to **INFO** level)
3. By using **.xml** file.

**Maven Overview: -** It is a build automation tool for java projects. When a Maven project is created, it creates default project structure and developer is only required to place a files accordingly. It resolves following problems: -

1. Adding set of jar files.
2. Dependencies and Version.

**POM** in maven stands for **Project Object Model.** It contains information about the project and various config details used by Maven to build the project. It also contains the goals and plugins.

1. **Local Repository: -** Repository located in user’s local system.
2. **Central Repository:** - It is located on web. It has created by Maven itself.
3. **Remote Repository:** - It is located on web.

**Jenkins Overview: -** Jenkins is an open source continuous integration tool. It is cross-platform and can be used on any OS. It can be used to automate all sort of tasks such as building, testing and deploying etc.

1. Jenkins is easy to install
2. It keeps track of changes done in repository
3. It is very easy to configure to send email / report
4. To distribute build on machines

**Automation Framework: -** It is a combination of various guidelines, coding standards, reporting mechanism to pillar automation testing. It reduces manual efforts to a limit. It will reduced maintenance  
cost. In case of any changes in web application then we need to change test data rather than test script. Test data can be separately kept in excel files, db, properties file. It provides benefits of code re-usability, easy reporting etc.

1. **Data Driven Framework: -** It helps user separate the business logic and test data from each other. It lets user to store data in external sources like file, xml, db etc. The data is stored as KEY-VALUE pair. Hence key is used to access data.
2. **Any change in data would not affect test script.**
3. **Increases flexibility and maintainability.**
4. **Keyword Driven Framework: -** It is a type of Functional Automation Testing framework which is also known as Table Driven Framework. In this certain set of code belonging to test script is kept in an external data file. These set of code are known as Keywords. These keywords can be used multiple times. **Advantage: - Single keyword can be used multiple times. Dis-Advantage: - User must know keyword creation strategy and it becomes complex as it grows.**

**Selenium Grid: -** The choice of selenium grid is depending upon the availability of infrastructure within project. If remote machines are available grid will help us to reduced execution time of regression test suite. It distributes the test across multiple physical or virtual machines. So that test scripts can be executed in parallel. Grid makes use of concept **hub-node** where we run the test only on hub and the execution will be done by different machines called as nodes.

1. Save time in executing test suite.
2. Run your test on different browser etc.

**Setting of HUB & NODES: -**

1. Download **selenium-standalone-server.jar** file.
2. Java –jar selenium-server-standalone-2.53.0.jar –port 4444 –role hub
3. Register nodes to the hub for ex: - <http://10.123.75.31:4444/grid/register> (please change IP accordingly as per your hub IP)
4. **OR ELSE DO** localhost:4444/grid/console
5. **Java –Dwebdriver.ie.driver = D:\driver\IEDriverServer.exe –jar** selenium-server-standalone-2.53.0.jar –role webdriver –hub <url> -browser browserName = “InternetExplorer” –port 4444
6. **OR DesiredCapabilities cap = DesiredCapabilities.internetExplorer ();**

**Cap.setBrowserName (“internet Explorer”);**

**String node = “http://10.138.16.48:4444/wd/hub”;**

**Driver = new RemoteWebDriver (new URL (node), cap);**