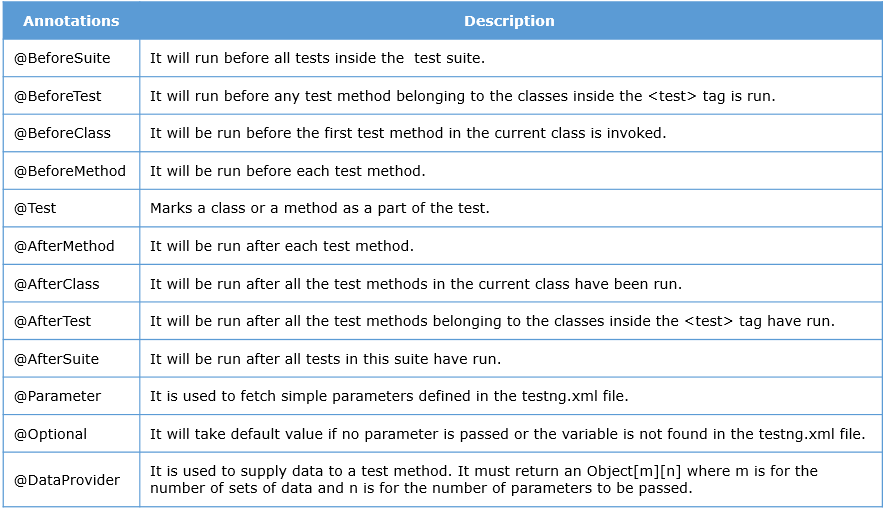
**TestNG** is an open source automated testing framework; where NG means Next Generation. TestNG is similar to JUnit (especially JUnit 4), but it is not a JUnit extension. It is inspired by Junit and NUnit (Junit equivalent in C#). TestNG is designed to cover all categories of tests like unit, functional, and integration.

Major features of TestNG from Selenium perspective are:

* It has the ability to produce HTML Reports after every test execution (Selenium doesn’t have the capability to generate reports)
* Annotations makes the methods easier to differentiate
* Grouping and prioritization of tests can be done to control execution flow of test cases
* Data can be Parameterized using different built in annotations
* Parallel execution of test methods is possible.

**Annotations available in TestNG**.

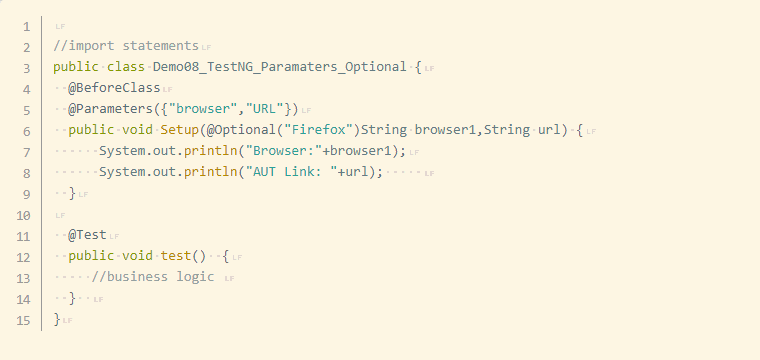


Execution flow will be:

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

**TestNG annotations | Parameters | Optional**

The @parameters in the test method is used to fetch simple parameter values from parameter tag mentioned in the testng.xml file.



From the above code file you can conclude that,

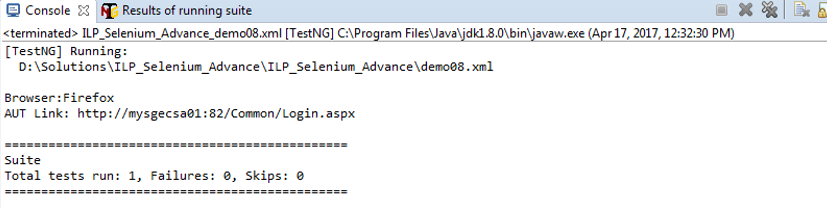
* The TestNG class is expecting two parameters browser and URL which will get fetched from testng.xml file and are Case Sensitive.
* If the parameter browser is not present in testng.xml file or if there is any mismatch in variable name, the test will take the default value mentioned in @Optional during execution.
* The order and number of arguments in @Parameters and the method signature needs to be same as one-one mapping occurs between the value passed through the xml file and the arguments mentioned inside Setup() method.

e.g.: @parameters is having two variables browser and URL and hence the test method should accept two arguments. The value passed through the variable URL will be mapped to the variable url in the Setup() method.



In the above testng.xml file the variable name is Browser but the variable name present in TestNG class is browser and hence the default value Firefox will be taken during the execution.

**Output:**



**DataProvider annotation:-**

The annotation @DataProvider is used to run the test method multiple times with different sets of test data. The method which has the logic to fetch the data and arrange the same will be annotated with @DataProvider annotation and the attribute dataProvider will be used to link the test method and dataprovider method.

The key points about DataProvider annotation:

1. @DataProvider annotation is used to parameterize the test methods with sets of test data.
2. It must return an Object[][] or Iterator<Object[]>
3. The DataProvider attribute in @Test will have the name of the method annotated with @DataProvider as the value.

*@DataProvider*

*public Object[][] getData() {*

*…..*

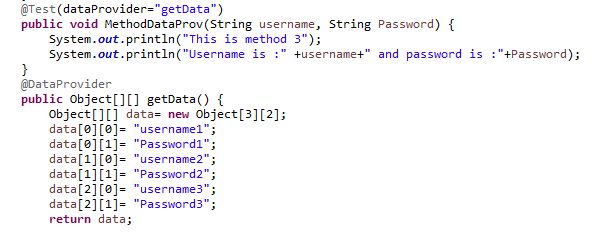
*}*

1. You can map the DataProvider method to the test method using below syntax.

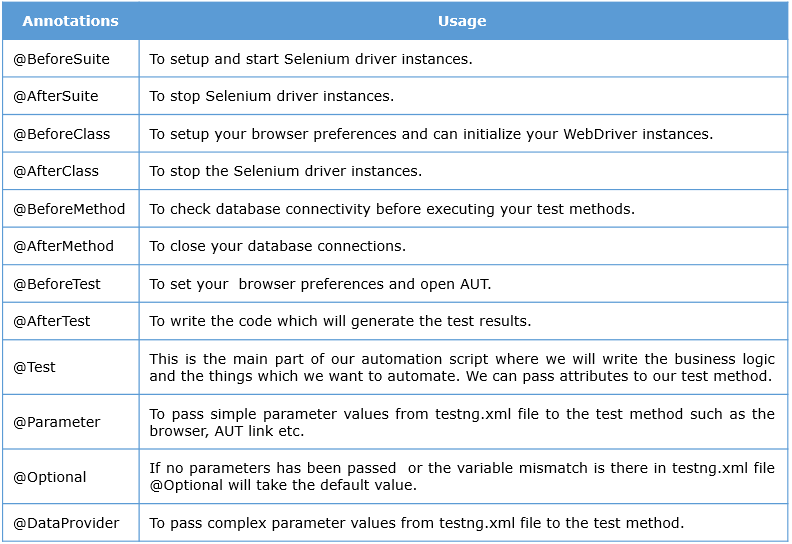
*@Test(dataProvider = "getData")*

You can give a name to the DataProvider method using name attribute.

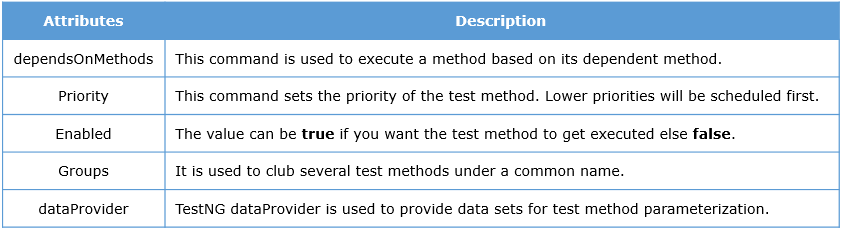
Eg:



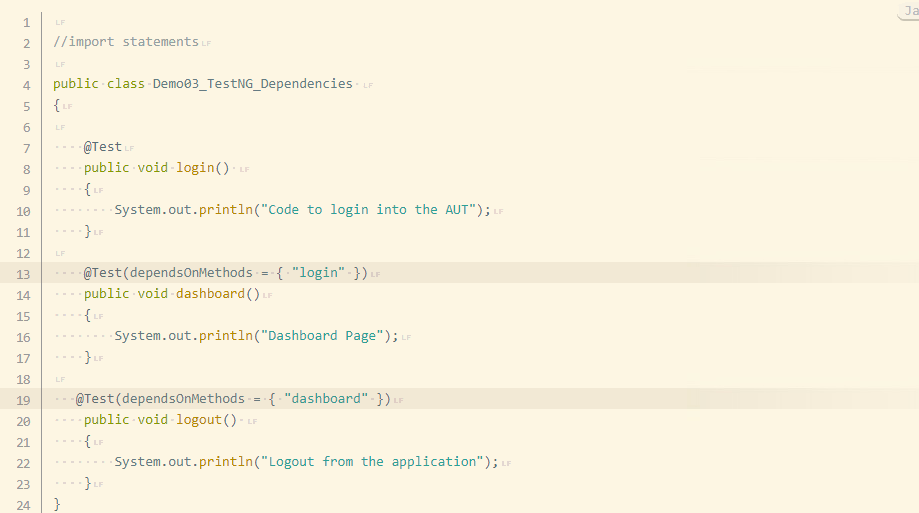
**Usage of TestNG annotations in Selenium**



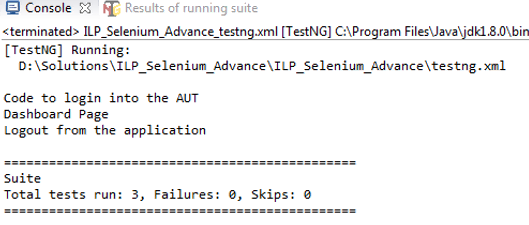
**Attributes of @Test annotation**



The **dependsOnMethods** attribute helps to define the dependency between various test methods. Let’s see it in action in the code below.

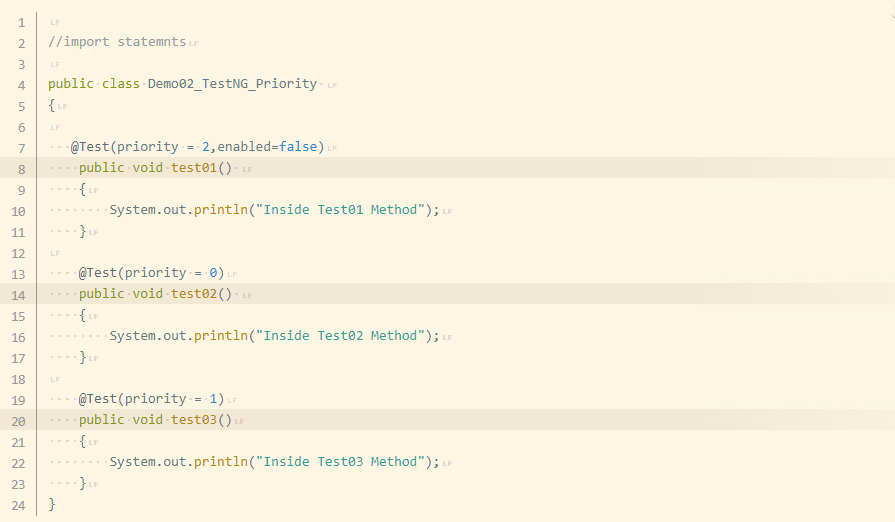


**Output**

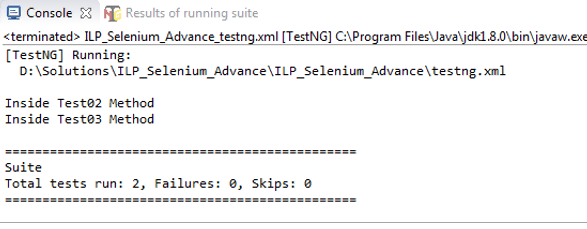


The **priority attribute** helps in prioritizing the test methods. The method accepts numerical values and the method with the lowest numerical value will be executed first.

The **enabled** attribute accepts **true** or **false** as values. The method having the value **false** will not be executed during the execution.



Output:



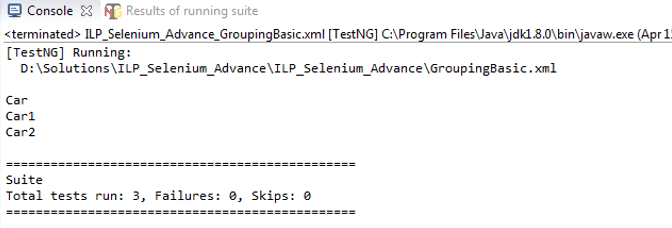
The **groups attribute** is used to add test methods to a group. All the test methods belonging to a group can be accessed at a later point using the group name.



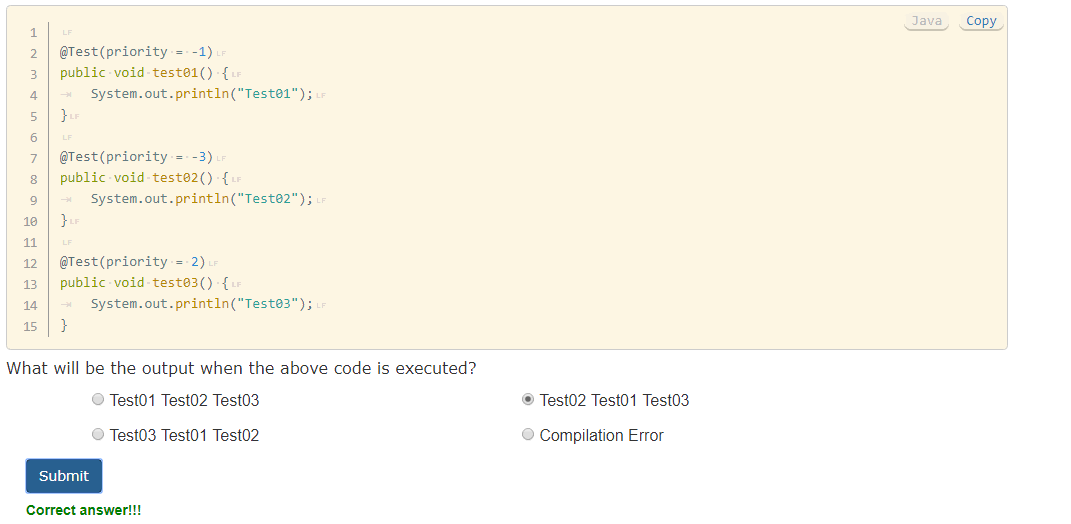
When you have to execute methods belonging to a specific group, you can configure the group name to be executed in the testng.xml file by using the tags <groups>, <run> and <include> . The xml file will look as below.



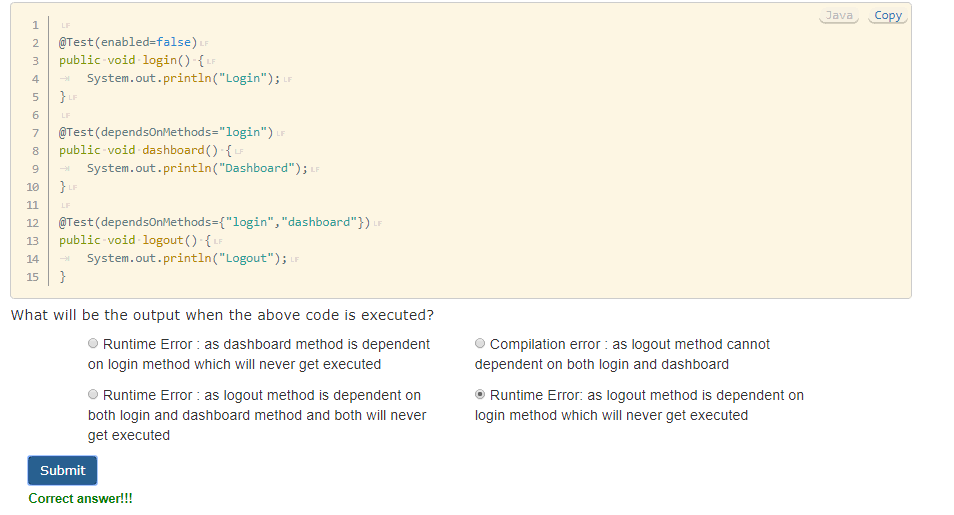
Output



**Question:**



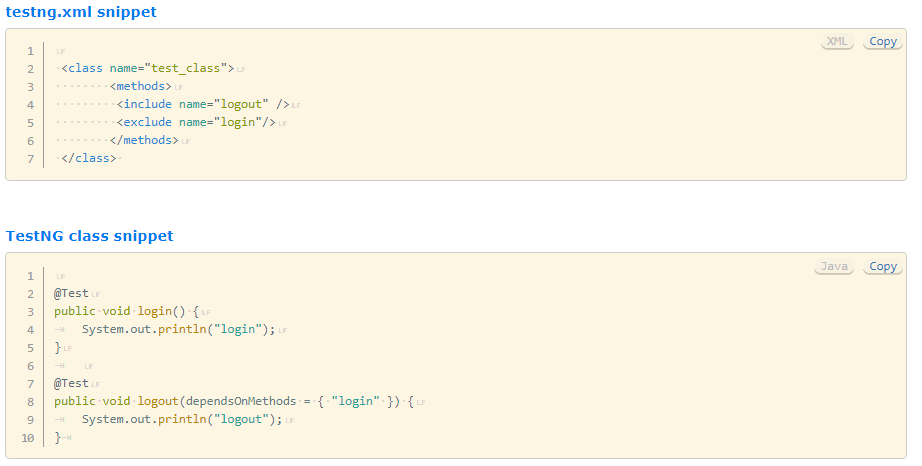
**Question:**

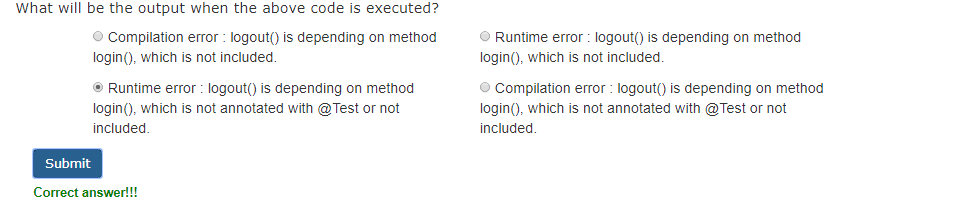


**Question:**



**Question:**





**TestNG assertions**

Verification is required to ensure that a functionality in the AUT is working fine. The test cases created needs to display the status as either pass or fail after execution which can be achieved by using the assertions provided in TestNG framework.

TestNG has two types of assertions available:

1. Hard assert (normal assertion)
2. Soft assert

**Hard assert**

This assertion puts a strict restriction on the test script. The test script will stop executing when the assertion fails and the test will be shown as failure in the TestNG report.

If the requirement being asserted is crucial and the execution should stop as soon as the assertion fails then you should opt for hard assert.

**Soft assert**

This assertion ensures that the test execution continues even if an assertion statement fails in the test script. The failure of a test can be reported in the TestNG report even if multiple soft assert statements has failed.

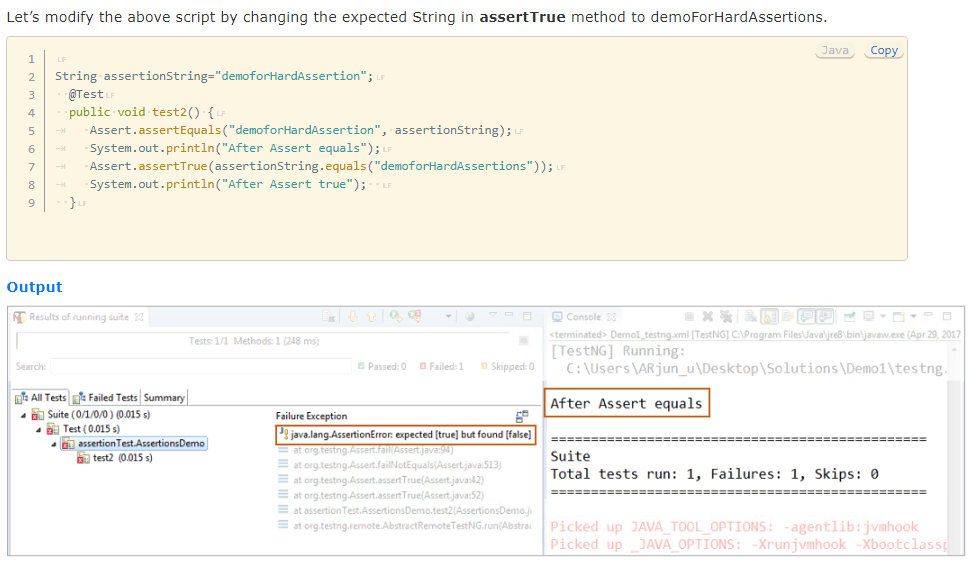
If the requirement being asserted is not crucial and the execution can still continue even if the assertion fails then you can opt for soft assert.

Most commonly used assert statements:-

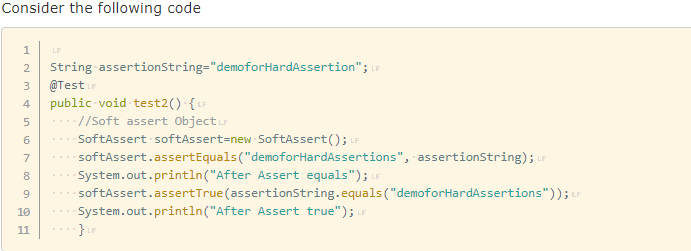
1. **assertEquals(String actual, String expected):** This method compares whether the expected string value and actual string value are the same. If the values are different an AssertionError is thrown.
2. **assertTrue(boolean condition):** This method asserts that a condition is true. If the condition is not true an AssertionError is thrown.
3. **assertFalse(boolean condition):** This method asserts that a condition is false. If the condition is not false an AssertionError is thrown.

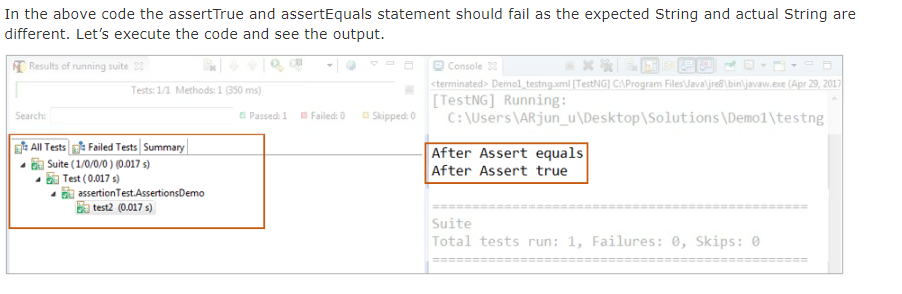
Eg. TestNG assertion: Hard assertion





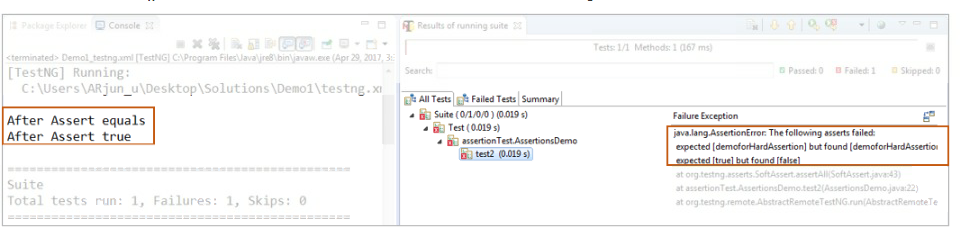
Eg. TestNG assertion: Soft assertion





The failed assertion statement had no effect on the TestNG result. For the failure to get reflected in the TestNG report you have to ensure that the method assertAll() is called after all the assertions are done. This method collates all the failures and decides whether to fail the test or not at the end. So instead of writing a custom logic, the TestNG library itself offers the facility to perform Soft Assertions in your test.

Invoke the assertAll() method inside the test method and execute the testcase again.

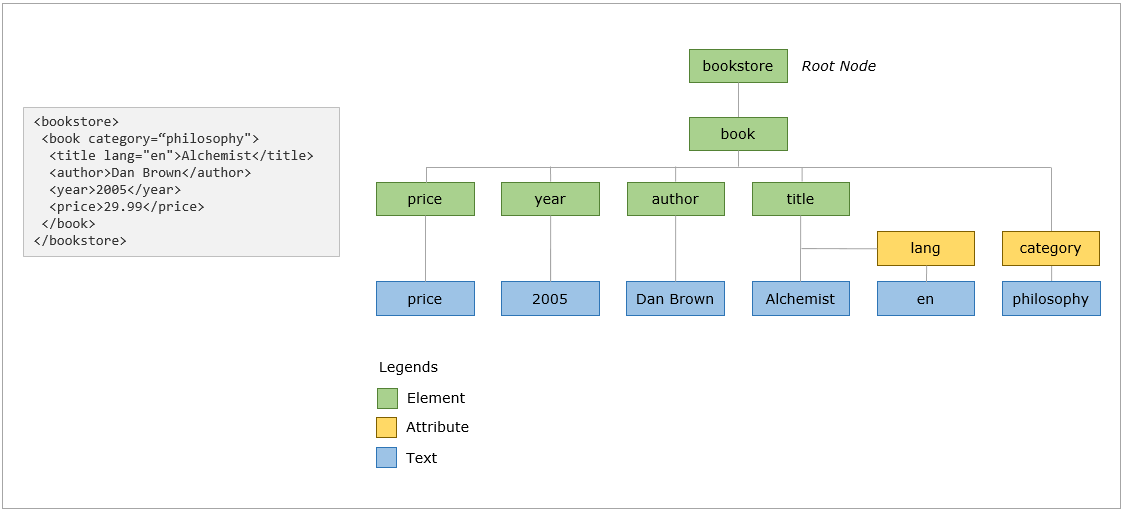


**Question**

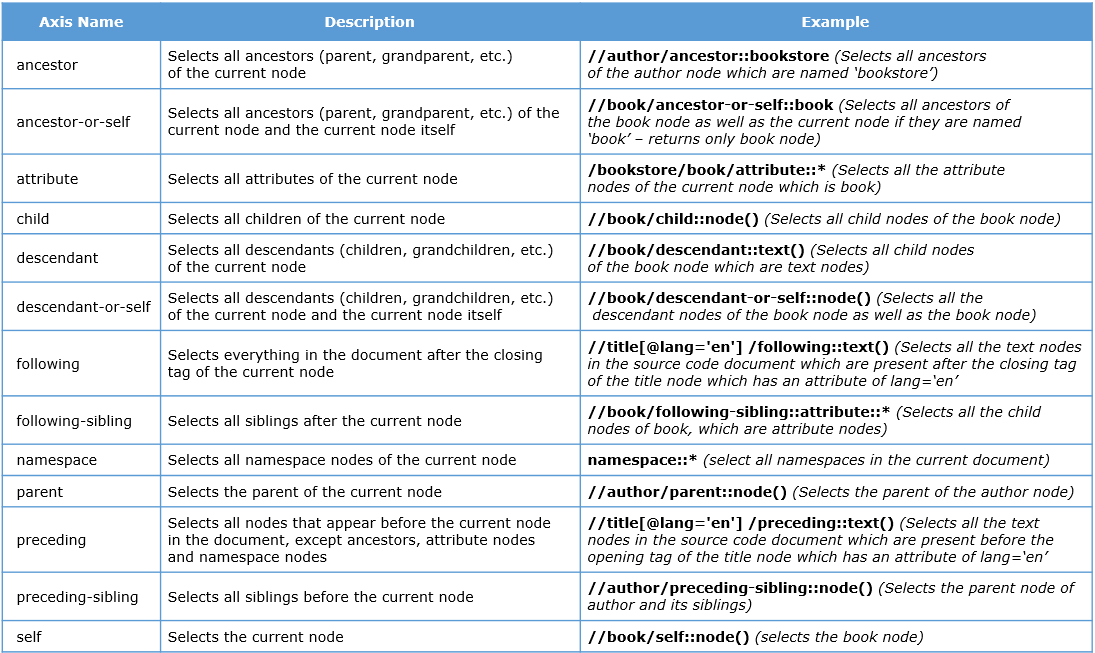


**XPath Axes terminologies**

* **Parent:** Node immediately above.
* **Child:** Node immediately below.
* **Ancestors:** Parent, parent’s parent and so on till the root node.
* **Descendants:** Child, child’s child and so on till any of the atomic nodes.
* **Siblings:** Nodes sharing the same parent node.
* **Namespace:** Node which contains the definition of other nodes. Corresponds to namespace tag in the document.
* **Attribute:** Characteristics defined for a specific node type. Corresponds to attribute labels for tags in the document.
* **Context Node:** The node with respect to which, other nodes are referred to.



XPath expression



**What is SSL certificate?**

SSL (Secure Sockets Layer) is standard security protocol which is used to establish secured connection between the server on which application is hosted and the client machine from where the user is trying to access the application.

SSL is used to keep sensitive informations encrypted such as username, password, debit/credit card details sent across the Internet.

If the information is not encrypted with an SSL certificate, any computer in between client machine and the destination server can access the private information.

When an SSL certificate is used, the information will be encrypted and it will ensure that you are sending information to the right and trusted server

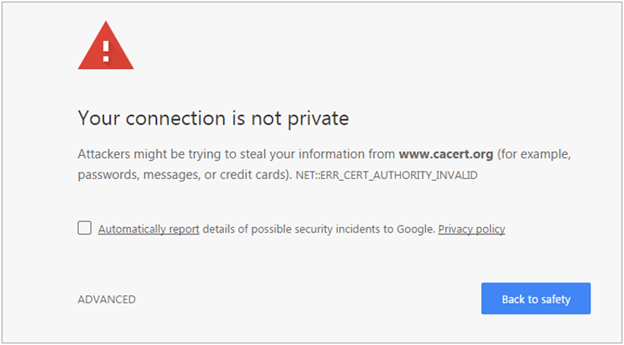


**Different SSL certificate error in different browsers**

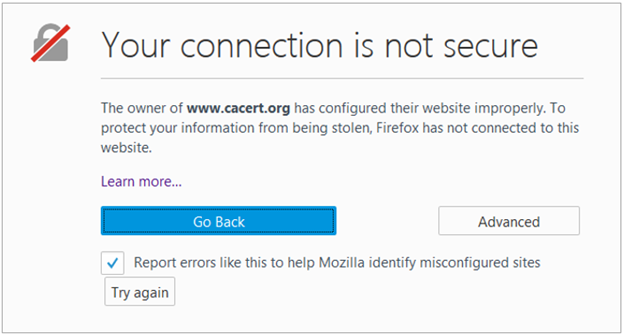
Suppose, you are trying to access a website whose web address starts with https and the SSL certificate for the website got expired or the certificate is not trusted. In this case it will throw an error message as The connection is untrusted or The site's security certificate is not trusted depending upon the browser you are using.

The types of error according to the browser.

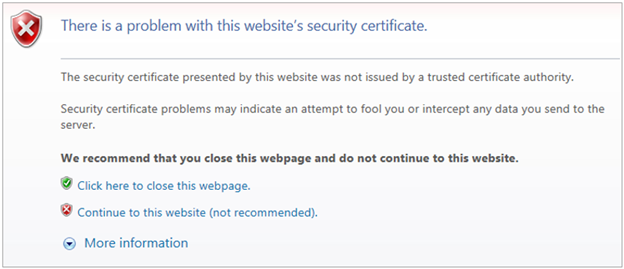
**Google Chrome –** Your connection is not private.



**Mozilla Firefox –** Your connection is not secure.

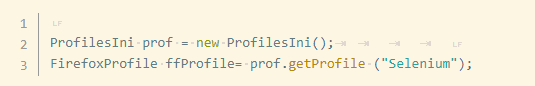


**Internet Explorer –** There is a problem with this website’s security certificate.



**SSL Certificate error handling in Firefox**

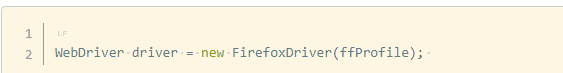
**Step 1:** Access the profile created in the script as below and create the FirefoxProfile object.



**Step 2:** Now we need to set setAcceptUntrustedCertificates and setAssumeUntrustedCertificateIssuer properties in the FireFox profile



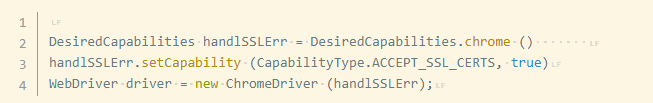
**Step 3:** Now use the FireFox profile in the FireFox driver object.



Note: setAcceptUntrustedCertificates and setAssumeUntrustedCertificateIssuer are capabilities to handle the certificate errors in web browsers.

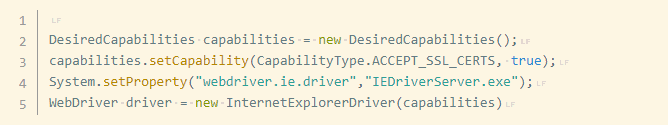
**Expired SSL certificate handling in Chrome**

For handling SSL error in Chrome, you need to use DesiredCapabilities of Selenium WebDriver. The below code will help to accept all the SSL certificates in chrome, and the user will not receive any SSL certificate related error using this code.



**Expired SSL certificate handling in IE**

For handling SSL error in IE, you need to use DesiredCapabilities of Selenium WebDriver. The below code will help to accept all the SSL certificates in IE, and the user will not receive any SSL certificate related error using this code.

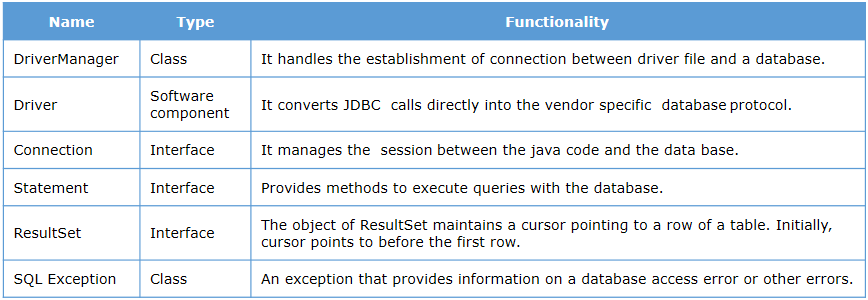


**Database connection**

**What is JDBC?**

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It provides methods to query and update data in a database, and is oriented towards relational databases by acting like a bridge between the Java Programming language and a wide range of databases. JDBC API uses jdbc drivers to connect with a database.

**JDBC Classes and Interfaces**



**DriverManager**

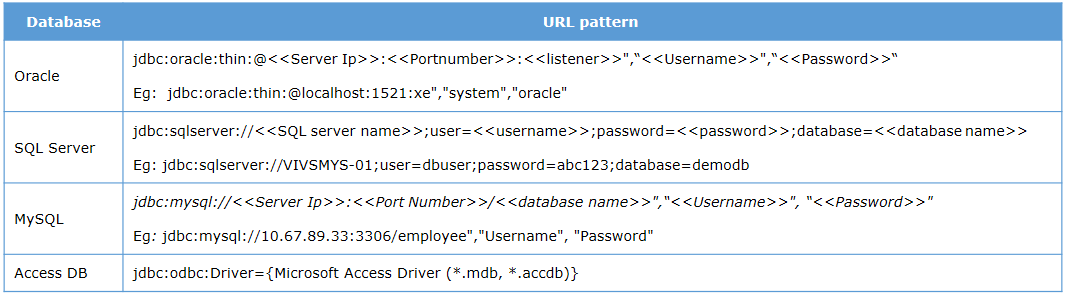
The **DriverManager** class acts as an interface between user and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver.

The most commonly used method to establish connection with the database is getconnection(String URL) method, where the URL is the string to establish connection with DB. Depending upon the database to which we need to connect, the URL changes accordingly.

Format

*jdbc:<dbtype>://ServerIP:portnumber/db\_name,Username,password*

The order of the argument varies from database to database and the connection URL pattern for some databases are given below.



**Driver**

JDBC **Driver** is a software component that enables java application to interact with the database. We will be making use of Thin drivers for establishing connection. The thin driver converts JDBC calls directly into the vendor-specific database protocol and hence known as thin driver. It is fully written in Java language.

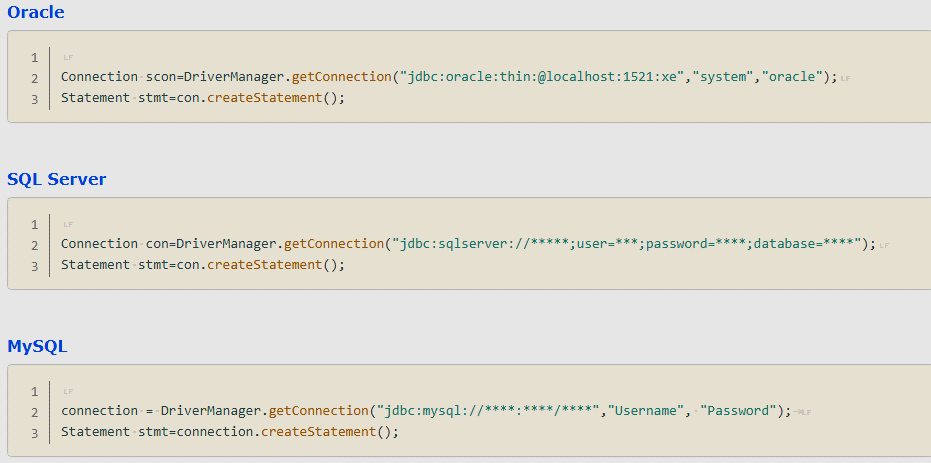
We will be working with Oracle/SQL/MySQL during this session and the following driver files are required to work with the same.

* Oracle: OJDB7 for Oracle 10g and OJDBC14 for Oracle11g and above.
* SQL Server: sqljdbc4-2.0
* MySQL: mysql-connector-java 5.1.31 .

**Connection | Statement | Resultset**

A Connection is the session between java application and database. It provides methods for the creation of statement instances and for managing transactions in database interactions.

The DriverManager getConnection() method returns an instance of Connection. The connection object can be used to create a Statement instance by invoking the method createStatement(). The Statement interface provides methods to execute the SQL queries with the database.



The most important methods in Statement interface are as follows.

* public ResultSet executeQuery(String sql) is used to execute SELECT query. It returns an object of ResultSet type.
* public int executeUpdate(String sql) is used to execute a specified query. Its may be a create, drop, insert, update, delete statements.
* public boolean execute(String sql) is used to execute queries that may return multiple results.
* public int[] executeBsatch() is used to execute batch of SQL commands.

Prominently we will be fetching data from the database, so let’s have a look at the code depicting the execution of query using executeQuery method which returns a Resultset object.

The object of ResultSet maintains a cursor pointing to a row of a table. Initially, cursor points to before the first row. By default, ResultSet object can only be moved forward and is not updatable.

**Overview of Page Object Model**

**Page Object Model(POM)** is a design pattern to create Object Repository for web UI elements.

Under this model, for each web page in the application, there should be a page class. This page class will have variables storing attribute values to identify web elements. The page will also contain methods which perform operations on the Web Elements.

**Why POM?**

Over time the amount of test scripts involved in the test repository will increase. Maintaining huge test scripts of selenium become tedious when the web element attributes gets updated in the AUT. In a similar manner, a lot of preconditions which needs to be met before performing an action will lead to duplication of code in the test scripts.

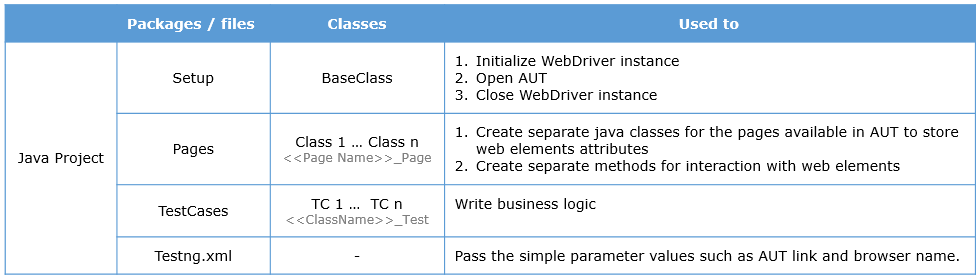
e.g. Before you can perform the check on the Account balance for your account number, you have to successfully login to the application. Hence the code to login to the application will get repeated every time you have to interact with the application.

POM offers the following advantages compared to the traditional way of writing the test scripts

* The Object repository ensures that the attribute value has to be changed only in the page class, for the change to be reflected in all the test scripts.
* Clear separation between the attribute values, interactions and verification.
* Code becomes less and optimized because of the reusable page methods in the POM classes.

Methods get more realistic names which can be easily mapped with the operations happening in the GUI. E.g: if after clicking on a button we navigate to the home page, the method name will be like 'gotoHomePage()'.

**Structure of POM**



From the above images the following can be inferred.

* The Java project will contain the following packages setup, pages and testcases.
* The setup package will contain the BaseClass
* The Pages package will have object repository for different pages in the AUT
* The TestCases package will have the business logic for different modules

**The key points about Page Object Model:**

1. The page object class contains the following:

* Variables storing the web element attribute type and value.
* Methods for various interaction with the web page.

1. Base class which contains the following:

* Method to initialize WebDriver instances
* Methods to terminate WebDriver instances

1. Testcase class contains test methods to invoke the different methods in page objects class. The following points need to be kept in mind:

* Objects for different page object class is created and the methods for various interactions are invoked.
* Whenever there is a method to click on a button or a link, you should return the object to the next page from the method.

**The key points about page object model project execution :**

1. Multiple tests can be grouped under a single suite tag.
2. Multiple tests can be executed in parallel by configuring the thread-count and parallel attributes within suite tag.
3. Each test tag inside the suite should have a unique value for name attribute.
4. Every TestNG test execution creates an index.html file inside test-output folder which contains the following:

* The name of the tests which got executed.
* Time taken for the execution of each test.
* Exception details for the failed test method.
* Execution status of each test script.
* Groups which got executed.

**Log4j**

Apache Log4j is a Java based reliable, fast and flexible logging utility which is used for logging execution information.

Why use Log4j?

* It is open source and can be easily configured in Selenium project.
* Using Log4j, we can store the Selenium project execution flow details in a file or databases or console
* Using log statements rather than print statements in the code gives better status of a project execution

**Log4j has three principal components:**

1. Loggers

* It is responsible for logging information’s provided by test script creator
* Primarily there are five kinds of log levels FATAL, ERROR, WARN, INFO and DEBUG

1. Appenders

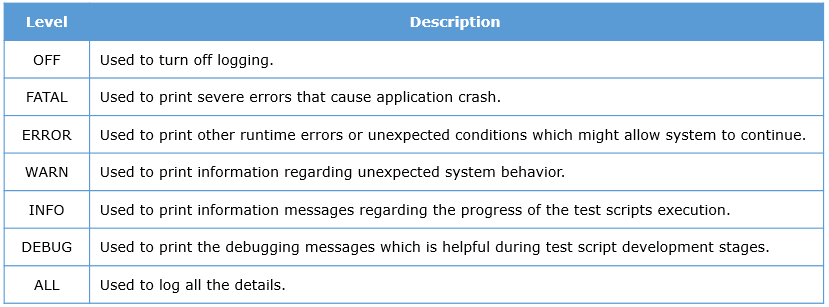
* It is responsible for writing log messages.
* It has several types of Appenders such as
  1. ConsoleAppender used to log standard output in console
  2. FileAppender used to write logs into files
  3. RollingFileAppender extends FileAppender to backup log files when they reach their maximum size

1. Layout

* It is responsible for formatting logging information in different styles.

**Log4j log levels**

The following table defines the built-in log levels and messages in Log4j, in decreasing order of severity.

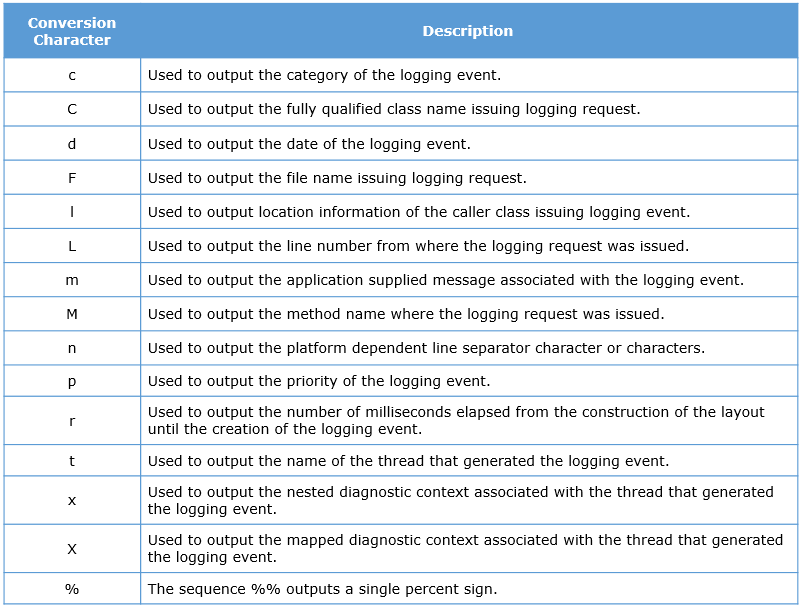


Eg.



**Log4j layout**

To specify a pattern for the log messages you can make use of conversion characters. The conversion characters should be preceded by a % symbol.



e.g:

Please have a look at the pattern below.

[%p] %c %M - %m%n

The sample output will be as follows:

[INFO] MyClass - foo - this is a log message

**Project enhancement using build automation**

To perform build automation, you can make use of the following tools:

1. Apache Maven
2. Apache ANT
3. Gradle

**Overview of Maven**

Maven is a build automation tool used primarily for Java projects. Maven uses convention over configuration which means developers are not required to create build process themselves. Maven provides sensible default behavior for projects. When a Maven project is created, it creates default project structure and developer is only required to place files accordingly.

There are many problems that you may face during the project development which are discussed below:

1. Adding set of Jars in each project: In case of selenium projects, we need to add multiple jar files in each project.
2. Dependencies and Versions: Ensuring that the jar files and the required dependencies are added to the project for developing, compiling and executing the same.

Maven structure

POM in Maven stands for Project Object Model. It is fundamental Unit of Work in Maven. It is an XML file which resides in the base directory of the project as pom.xml. The POM contains information about the project and various configuration detail used by Maven to build the projects.

POM also contains the goals and plugins. While executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, and then executes the goal.

**Maven repository**

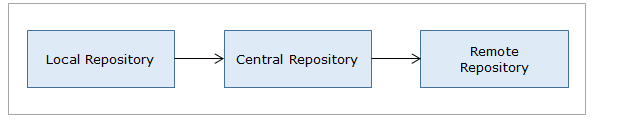
A maven repository is a directory of packaged JAR file with pom.xml file. Maven searches for dependencies in the repositories. There are 3 types of maven repository:

Local Repository

Central Repository

Remote Repository

Maven searches for the dependencies in the following order:



* **Maven Local Repository**

Maven local repository is located in your local system. It is created by the maven when you run any maven command.

By default, maven local repository is %USER\_HOME%\.m2 directory. For example: C:\Users\<<Username>>\.m2 .

* **Maven Central Repository**

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

Please access the central repository. The central repository contains a lot of common libraries that can be viewed by using the search page.

* **Maven Remote Repository**

Maven remote repository is located on the web. Some libraries can be missing from the central repository and may be required to be downloaded from the individual repositories by the firm. Eg: JBoss library file needs to be downloaded from the JBoss nexus repository.

**Build trigger using Jenkins**

**Why Jenkins with Selenium and Maven**

Running Selenium tests in Jenkins allows you to schedule execution every time you make changes in your test scripts and deploy the final project to a new environment when the tests pass.

* Jenkins can schedule your tests to run at specific time.
* You can save the execution history and Test Reports.
* Jenkins supports Maven for building and testing a project in continuous integration.

**What is Jenkins?**

Jenkins is an open source continuous integration tool. It is cross-platform and can be used on Windows, Linux, Mac OS and Solaris environments. It can be used to automate all sorts of tasks such as building, testing, and deploying software.

**Important feature of Jenkins**

Jenkins is easy to install either using direct installation through .exe file or through .war file to deploy using application server.

* Jenkins keeps track of all the changes done in repository for future references.
* It is very easy to configure Jenkins in order to send mail for every build status.
* Jenkins can be configure to run the automation test build on TestNG after every project build
* Jenkins can be configured to distribute the build on multiple machines
* 3rd party plugin can be configure with Jenkins in order to use additional features.

**What is Automation Framework?**

An automation framework is a combination of various guidelines, coding standards, project hierarchies, modularity, reporting mechanism, test data injections etc. to pillar automation testing. A user can follow these guidelines while automating application to take advantages of various productive results.

The advantages in using a framework are as follows.

1. Frameworks reduces the manual efforts to a limit, works can be completed with minimal user intervention.
2. It will reduce maintenance cost with effective project repositories.
3. In case of any changes in the web application, instead of changing test scripts, only the test case file needs to be updated.
4. Test data can be separately kept in excel files, no need to include them inside the test scripts. So in case of any modifications, you can simply manipulate the data in the excel file.
5. Common library files can be reused when required, no need to develop them every time. Thus it saves time of the team and improves the productivity.
6. It also provides us the benefits of code re-usability, role-based access, easy reporting, consistency and maximum test coverage.

The four most popular testing frameworks are:

* Modular
* Data-Driven
* Keyword
* Hybrid

**Modular Framework**

Modular Framework is based on the object oriented programming concept of abstraction. The framework divides the entire “Application Under Test” into a number of logical and isolated modules. For each module, you have to create a separate and independent test script. These test scripts taken together builds a larger test script representing more than one modules.

Advantages

* The framework introduces high level of modularization which leads to easier maintenance.
* If the changes are implemented in one part of the application, only the test script representing that part of the application needs to be fixed leaving all the other parts untouched.

Disadvantages

* While implementing test scripts for each module separately, you embed the test data (Data with which we are supposed to perform testing) into the test scripts. Thus, whenever you are supposed to test with a different set of test data, it requires the manipulations to be made in the test scripts.

**Data-Driven framework**

Data Driven Testing Framework helps the user separate the business logic and the test data from each other. It lets the user store the test data in an external data source. The external data source can be a property file or xml file or excel sheet or text file or ODBC repository or a combination of multiple types. The data is conventionally stored as a “Key-Value” pair and hence the key can be used to access and populate the data within the test scripts.

**Advantages**

* It considerably reduces the total number of test scripts required to cover all the possible combinations of test scenarios and hence lesser amount of code is required to test a complete set of scenarios.
* Any change in the test data would not affect the automation scripts created.
* Increases flexibility and maintainability

Disadvantage

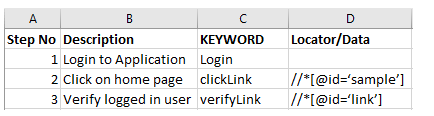
* The process requires good knowledge of the programming language as we need to develop the generalized logic to read data from any data sets.

**Keyword framework**

Keyword Driven Framework is a type of Functional Automation Testing Framework which is also known as Table-Driven testing or Action Word based testing. In this framework certain set of code belonging to the test script is kept in an external data file.

These set of code are known as keywords. These will be generalized functions written in a class to perform certain sets of actions.

e.g.



keyword

In the above example keywords like login, clickLink and verifyLink are defined within the code. All the keywords can be reused multiple times in test cases. Locator column contains the locator value that is used to identify the web elements on the screen or the test data that needs to be supplied.

All the required keywords are designed and placed in base code of the framework.

Advantages

* A single keyword can be used across multiple test scripts.

Disadvantage

* The user should be well versed with the Keyword creation strategy.
* The framework becomes complicated as it grows and the number of keyword increases.

**Hybrid framework**

Hybrid Test framework is a concept with the advantages of both Keyword and Data driven framework.

**Features expected from Hybrid framework**

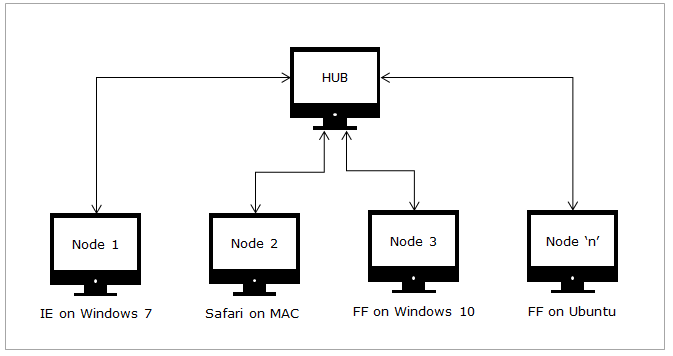
Please note that hybrid framework can be implemented by combining the features from the following:

* Modular and Data-Driven
* Data-Driven and Keyword-Driven
* Combination of all three

**Overview of Selenium Grid**

Selenium grid is part of the selenium suite which distributes the test across multiple physical or virtual machines, so that the test scripts can be executed in parallel (Simultaneously). It helps us in covering the test process across browsers and across different platforms.

Grid makes use of the concept of hub-node where you run the test only on the hub machine, but the execution will be done by different machines called as nodes.



Selenium grid should be made use in case you want to do either one or both of the following:

1. Run your tests against different browsers, operating system and machines all at the same time.
2. Save time in executing the test suite.

**Arhitecture of Selenium Grid**

The HUB

1. There will be only one HUB in a grid.
2. The test scripts are loaded in the HUB.
3. The Tests run in the HUB will be executed in the Node.

The Nodes

1. There can be one or more nodes in a grid.
2. Nodes are selenium instances, that will execute the tests loaded on the HUB.
3. Nodes can be launched on multiple platforms with different browsers.
4. The platform of the node need not be the same as that of HUB.

**Setting up HUB**

To set up the HUB in the local machine, you need the selenium server standalone jar file. Please follow the following steps to start the HUB in the machine.

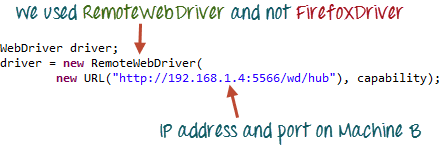
https://www.guru99.com/introduction-to-selenium-grid.html

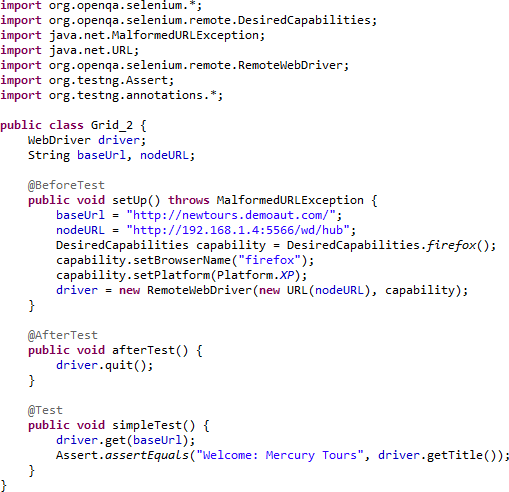
* Start the hub:

java -jar selenium-server-standalone-2.30.0.jar -role hub

* Start the node:

java -Dwebdriver.gecko.driver="C:\geckodriver.exe" -jar selenium-server-standalone-3.4.0.jar -role webdriver -hub [http://192.168.1.3:4444/grid/register -port 5566](http://192.168.1.3:4444/grid/register%20-port%205566)





**Selenium grid configuration using JSON File:**

**Selenium Architecture:**

<http://makeseleniumeasy.com/2017/04/02/hierarchy-of-selenium-classes-and-interfaces/>

