What is TestNG Framework?

TestNG is an open-source test automation framework for Java. It is developed on the same lines as JUnit and NUnit. A few advanced and useful features provided by TestNG make it a more robust framework than its peers. The NG in TestNG stands for 'Next Generation.'

What is TestNG in Selenium?

TestNG provides advanced features such as annotations, data-driven testing, test sequencing, and parallel testing to help you organize and execute your Selenium tests more efficiently and effectively.

TestNG Annotations

An annotation tag provides information about the method, class, and suite. It helps to define the execution approach of your test cases and the different features associated with it. Below are the major annotations used:

@Test– This is the root of TestNG test cases. To use TestNG, all methods should be annotated with this annotation.

Below is an example:

A few attributes associated with the TestNG annotation are:

Description: You can describe your test case under the description, stating what it does

```
@Test(description="This test validates login functionality")
```

Priority: You can prioritize the order of your test methods in TestNG by defining a priority. Based on the defined priority, the test shall execute in that order.

```
@Test(priority=1)
```

DependsOnMethod: This attribute works miracles if one test case is dependent on another. For example, to view your profile details, you need to login to the application. So, your profile test case is dependent on the login test case

```
@Test(dependsOnMethod="Login")
```

Enabled: Using this attribute, you can choose to execute or skip the execution of this test case. Setting it to true execute it and putting it to false will skip the test from the execution cycle

```
@Test(enabled='true')
```

Groups: Using this attribute, you can club your test cases into a single group and specify the group you wish to execute in your TestNG XML file. The test cases clubbed to that group will only be executed, and the rest will be skipped

```
@Test(groups="Smoke Suite")
```

@BeforeMethod and @AfterMethod – These annotations run before and after each test method

@BeforeClass and @AfterClass – These annotations run once before and after the first *@test* method in a class

@BeforeTest and @AfterTest – The BeforeTest annotation runs before the *@BeforeClass* annotation and the AfterTest annotation runs after the *@AfterClass* annotation

@BeforeSuite and @AfterSuite– These annotations run before and after any test annotated method in a class respectively. These annotations start the beginning of a test and the end of it, for all the classes in a suite

the execution order of these annotations, they execute in the below order:

@BeforeSuite -> @BeforeTest -> @BeforeClass -> @BeforeMethod -> @Test -> @AfterMethod -> @AfterClass -> @AfterCTest -> @AfterSuite

TestNG Assertions

The word **Assert** means to state a fact or belief confidently or forcefully. In Selenium, Asserts are validations or checkpoints for an application.

Types of Assertions

Hard Assertions

Soft Assertions (Verify Method)

Hard Assertions

Hard Assertions are ones in which test execution is aborted if the test does not meet the assertion condition. The test case is marked as failed. In case of an assertion error, it will throw the "***java.lang.AssertionError***" exception.

Methods:

1.assertEquals() is a method that takes a minimum of 2 arguments and compares actual results with expected results. If both match, the assertion is passed, and the test case is marked as passed.

2. **assertNotEquals()** is a method that does the opposite of the **assertEquals()** method. In this case, the method compares the actual and expected result. But if the assertion condition is met if the two are not identical. The test case is marked as passed if actual and expected results are not the same.

3. **assertTrue()**: This Assertion verifies the Boolean value returned by the condition. If the Boolean value is true, then the assertion passes the test case.

4. **assertFalse()**: This method works the opposite of **assertTrue()**. The Assertion verifies the Boolean value returned by the condition. If the Boolean value is false, the assertion passes the test case.

5.assertNull(): This method verifies if the expected output is null. If not, the value returned is false.

6. **assertNotNull()**: This method works opposite to the **assertNull()** method. The assertion condition is met when the method validates the expected output to be not null.

7.The **assertNotNull()** method verifies if the title of the page www.browserstack.com is null or empty.

Soft Assertions

In a hard assertion, when the assertion fails, it terminates or aborts the test. If the tester does not want to terminate the script, they cannot use hard assertions. To overcome this, one can use soft assertions.

Soft assertions are the ones in which the test execution does not stop if the test does not meet the assertion condition. The remaining tests are executed. All the above methods also work with soft assertions.

Syntax:

Assertion softAssert = new SoftAssert();

Factor	Hard Assert	Soft Assert
Definition	The assertion is an expression that is used to check the expected and actual results of a test case. It is used to make sure that the actual result matches the expected result.	Verification is a process of validating or confirming that the expected behavior of the application is happening.
Execution of the next step	Test execution will be aborted if the assert condition is not met.	Test execution will continue till the end of the test case even if the assert condition is not met.
Test Results	No need to invoke any method to view test results.	Tester needs to invoke assertAll(), to view assertions at the end of the test result.
Default Asserts	By default, Assert in Selenium WebDriver are Hard Asserts.	These are not included by default in the TestNG framework.
Package	Import the org.testng.Assert package.	Import the org.testng.asserts.SoftAssert package.
Syntax	Assertion hardAssert = new Assertion();	Assertion softAssert = new SoftAssert();
Usage	These should be used in scenarios where it is required to halt the test execution if a critical test step results in a failure.	These should be used in scenarios where the failure of certain conditions does not hamper the execution of other test steps in that method.