TestNG

* TestNG is an automation testing framework
* Using TestNG you can generate a proper report, and you can easily come to know how many test cases are passed, failed and skipped.
* You can execute failed test case separately.
* The TestNG provides an option, i.e., testng-failed.xml file in test-output folder. If you want to run only failed test cases means you run this XML file. It will execute only failed test cases.

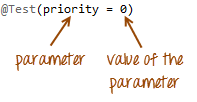
**Why Use TestNG with Selenium?**

Default Selenium tests do not generate a proper format for the test results. Using TestNG we can generate test results.

* Generate the report in a proper format including a number of test cases runs, the number of test cases passed, the number of test cases failed, and the number of test cases skipped.
* Multiple test cases can be grouped more easily by converting them into testng.xml file. In which you can make priorities which test case should be executed first.
* The same test case can be executed multiple times without loops just by using keyword called 'invocation count.'
* Using testng, you can execute multiple test cases on multiple browsers, i.e., cross browser testing.
* The testing framework can be easily integrated with tools like Maven, Jenkins, etc.
* Annotations used in the testing are very easy to understand ex: @BeforeMethod, @AfterMethod, @BeforeTest, @AfterTest
* WebDriver has no native mechanism for generating reports. TestNG can generate the report in a readable format like the one shown below.
* TestNG simplifies the way the tests are coded. There is no more need for a static main method in our tests. The sequence of actions is regulated by easy-to-understand annotations that do not require methods to be static.
* Uncaught exceptions are automatically handled by TestNG without terminating the test prematurely. These exceptions are reported as failed steps in the report.
* There are three major advantages of TestNG over JUnit:

1. Annotations are easier to understand
2. Test cases can be grouped more easily
3. Parallel testing is possible

* Annotations in TestNG are lines of code that can control how the method below them will be executed. They are always preceded by the @ symbol. A very early and quick example is the one shown below.
* TestNG does not require you to have a main() method.
* Methods need not be static.
* We used the @Test annotation. @Test is used to tell that the method under it is a test case.
* Since we use annotations in TestNG, we needed to import the package org.testng.annotations.\*.
* We used the Assert class. The Assert class is used to conduct verification operations in TestNG. To use it, we need to import the org.testng.Assert package.
* We can use multiple @Test annotations in a single TestNG file. By default, methods annotated by @Test are executed alphabetically.
* If you want the methods to be executed in a different order, use the parameter "priority". Parameters are keywords that modify the annotation's function.
* Parameters require you to assign a value to them. You do this by placing a "=" next to them, and then followed by the value.
* Parameters are enclosed in a pair of parentheses which are placed right after the annotation like the code snippet shown below.



* TestNG will execute the @Test annotation with the lowest priority value up to the largest. There is no need for your priority values to be consecutive.
* Aside from "priority," @Test has another parameter called "alwaysRun" which can only be set to either "true" or "false." To use two or more parameters in a single annotation, separate them with a comma such as the one shown below.

@Test(priority = 0, alwaysRun = true)

* Annotations:

@BeforeSuite: The annotated method will be run before all tests in this suite have run.

@AfterSuite: The annotated method will be run after all tests in this suite have run.

@BeforeTest: The annotated method will be run before any test method belonging to the classes inside the tag is run.

@AfterTest: The annotated method will be run after all the test methods belonging to the classes inside the tag have run.

@BeforeGroups: The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked.

@AfterGroups: The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked.

@BeforeClass: The annotated method will be run before the first test method in the current class is invoked.

@AfterClass: The annotated method will be run after all the test methods in the current class have been run.

@BeforeMethod: The annotated method will be run before each test method.

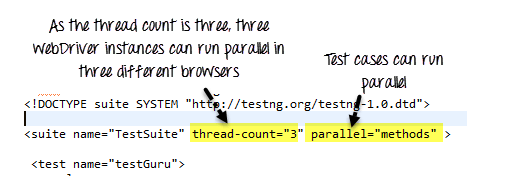
@AfterMethod: The annotated method will be run after each test method.

@Test: The annotated method is a part of a test case

* Grouping can be done by using "include" and "exclude" mechanism supported in testNG.

@Test (groups = { "bonding", "strong\_ties" })

* There are situations where you want to run multiple tests at the same time. In such cases, one can use "parallel" attribute



* The parallel attribute of suite tag can accept four values:

1. Tests: All the test cases inside <test> tag of Testing xml file will run parallel.
2. Classes: All the test cases inside a Java class will run parallel
3. Methods: All the methods with @Test annotation will execute parallel.
4. Instances: Test cases in same instance will execute parallel but two methods of two different instances will run in different thread.

The attribute thread-count allows you to specify how many threads should be allocated for this execution.