**TestNG Parallel Execution**  
**Stage:01**

* TestNG provides multiple ways to execute tests in separate threads. In testng.xml, if we set the 'parallel' attribute on the tag to 'tests', testNG will run all the ‘@Test’ methods in the tag in the same thread, but each tag will be in a separate thread.
* If we want to run methods/classes in separate threads, we need to set the 'parallel' attribute on the tag to 'methods' / 'classes'
* This helps us to run test methods/class tests in parallel. By using parallel execution, we can reduce the 'execution time' as tests are executed simultaneously in different threads**. In TestNG, we can achieve parallel execution in two ways**.  
   One with the testng.xml file and we can configure an independent test method to run in multiple threads.

First, let us look at a basic example for Parallel Execution of Test Methods using the 'parallel' attribute on the tag with 'method'.

**E.g. 1: Creating a class with two test methods and will try to execute in different threads.**

package com.parallel;  
import org.testng.annotations.Test;  
public class TestParallelOne {

@Test  
 public void testCaseOne() {  
 //Printing Id of the thread on using which test method got executed  
 System.out.println("Test Case One with Thread Id:- "

+ Thread.currentThread().getId());

}

@Test  
 public void testCaseTwo() {  
 //Printing Id of the thread on using which test method got executed  
 System.out.println("Test Case two with Thread Id:- "

+ Thread.currentThread().getId());

}  
}

The below is the simple testng.xml file,

<!DOCTYPE suite SYSTEM "<http://testng.org/testng-1.0.dtd>">  
<suite name="Parallel test suite" parallel="methods" thread-count="2">  
 <test name="Regression 1">  
 <classes>  
 <class name="com.parallel.TestParallelOne"/>  
 </classes>  
 </test>  
</suite>

**Explanation**: if you observe, we are defining two attributes 'parallel' and 'thread-count' at the suite level. As we want test methods to be executed in parallel, we have provided 'methods'. And 'thread-count' attribute is used to pass the number of maximum threads to be created.

**Note**:  
The assigning of the thread is taken care of by the processor. So we can't say which thread is going to execute which method.  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 **Stage:02**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
**Advantages of parallel test execution**

**a**. **Reduces execution time** – As tests are executed in parallel, multiple tests get executed simultaneously, hence reducing the overall time taken to execute the tests.

**b. Allows multi-threaded tests** – Using this feature, we can write tests to verify certain multi-threaded code in the applications.

Parallel test execution is vastly used by the QA industry for functional automation testing. This feature helps QA to configure their tests to be executed easily in multiple browsers or operating systems simultaneously.  
  
 **Run parallel TestNG test cases**

With more TestNG annotations(@test, @AfterMethod, @BeforeMethod etc)  
ParallelMethodTest.java  
package com.howtodoinjava.parallelism;   
import org.testng.annotations.AfterMethod;  
import org.testng.annotations.BeforeMethod;  
import org.testng.annotations.Test;  
public class ParallelMethodTest

{  
 @BeforeMethod  
 public void beforeMethod() {  
 long id = Thread.currentThread().getId();  
 System.out.println("Before test-method. Thread id is: " + id);

}

@Test  
 public void testMethodsOne() {  
 long id = Thread.currentThread().getId();  
 System.out.println("Simple test-method One. Thread id is: " + id);

}  
 @Test  
 public void testMethodsTwo() {  
 long id = Thread.currentThread().getId();  
 System.out.println("Simple test-method Two. Thread id is: " + id);

}  
 @AfterMethod

public void afterMethod() {  
 long id = Thread.currentThread().getId();  
 System.out.println("After test-method. Thread id is: " + id);

}  
}  
The preceding test class contains two test methods, which prints a message onto the console when executed. The ID of the thread on which the current method is being executed is evaluated using the Thread.currentThread.getId() code.

It also contains the before and after methods, which also prints the thread ID of the current thread onto the console when executed.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
**test-testng.xml**   
  
<suite name="Test-method Suite" parallel="methods" thread-count="2" >  
 <test name="Test-method test" group-by-instances="true">  
 <classes>  
 <class name="com.howtodoinjava.parallelism.ParallelMethodTest" />  
 </classes>  
 </test>  
</suite>

**E.g. 2:** In this example, we will learn about executing TestNG test classes in parallel**.**  
  
**Note**: each test class that is part of the test execution will be executed in its own thread.  
  
ParallelClassesTestOne.java  
public class ParallelClassesTestOne  
{  
@BeforeClass  
public void beforeClass() {  
long id = Thread.currentThread().getId();  
 System.out.println("Before test-class. Thread id is: " + id);  
 }

@Test  
public void testMethodOne() {  
 long id = Thread.currentThread().getId();  
 System.out.println("Sample test-method One. Thread id is: " + id);  
 }

@Test  
 public void testMethodTwo() {  
 long id = Thread.currentThread().getId();  
 System.out.println("Sample test-method Two. Thread id is: " + id);  
 }

@AfterClass  
 long id = Thread.currentThread().getId();  
 System.out.println("After test-class. Thread id is: " + id);  
}  
}  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ParallelClassesTestTwo.java  
public class ParallelClassesTestTwo  
{  
@BeforeClass  
public void beforeClass() {  
 long id = Thread.currentThread().getId();  
System.out.println("Before test-class. Thread id is: " + id);  
}

@Test  
 public void testMethodOne() {  
 long id = Thread.currentThread().getId();  
System.out.println("Sample test-method One. Thread id is: " + id);  
 }

@Test

public void testMethodTwo() {  
 long id = Thread.currentThread().getId();  
 System.out.println("Sample test-method Two. Thread id is: " + id);  
 }

@AfterClass  
 public void afterClass() {  
 long id = Thread.currentThread().getId();  
System.out.println("After test-class. Thread id is: " + id);  
}  
}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*t**test-testng.xml**  
<suite name="Test-class Suite" parallel="classes" thread-count="2" >  
 <test name="Test-class test" >  
 <classes>  
 <class name="com.howtodoinjava.parallelism.ParallelClassesTestOne" />  
 <class name="com.howtodoinjava.parallelism.ParallelClassesTestTwo" />  
 </classes>  
</test>  
</suite>

The previous test result clearly shows that each test class and its respective before class and after-class methods are executed in a different thread. This is identified by the id of the thread that is printed on the console.

**TestNG run parallel suites**

Let’s learn about executing each test inside a suite in parallel, that is, each test that is part of the test suite execution will be executed in its own separate respective thread.  
  
Let’s learn about executing each test inside a suite in parallel, that is, each test that is part of the test suite execution will be executed in its own separate respective thread.

ParallelSuiteTest.java

package com.howtodoinjava.parallelism;  
import org.testng.annotations.AfterClass;  
import org.testng.annotations.AfterTest;  
import org.testng.an  
@Parameters({ "test-name" })

public void beforeTest(String testName) {  
 this.testName = testName;  
 long id = Thread.currentThread().getId();  
 System.out.println("Before test " + testName + ". Thread id is: " + id);

}

@BeforeClass  
 public void beforeClass() {  
 long id = Thread.currentThread().getId();  
System.out.println("Before test-class " + testName + ". Thread id is: "

+ id);

}

@Test

public void testMethodOne() {

long id = Thread.currentThread().getId();

System.out.println("Sample test-method " + testName

+ ". Thread id is: " + id)  
 }

@AfterClass

public void afterClass() {

long id = Thread.currentThread().getId();

System.out.println("After test-method " + testName

+ ". Thread id is: " + id);

}

@AfterTest

public void afterTest() {

long id = Thread.currentThread().getId();

System.out.println("After test " + testName + ". Thread id is: " + id);

}

}

**test-testng.xml**

<suite name="Test-class Suite" parallel="tests" thread-count="2">  
 <test name="Test-class test 1">  
 <parameter name="test-name" value="test-method One" />  
 <classes>  
 <class name="com.howtodoinjava.parallelism.ParallelSuiteTest" />  
 </classes>  
 </test>

<test name="Test-class test 2">  
 <parameter name="test-name" value="test-method One" />  
 <classes>  
 <class name="com.howtodoinjava.parallelism.ParallelSuiteTest" />  
 </classes>  
 </test>  
</suite>

The previous test result clearly shows that each test in a suite is executed in its respective thread. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
**Configure a TestNG test to run in multiple threads**  
Earlier we discussed how to run classes, methods, and tests in parallel or in multi-threaded mode. TestNG also provides the flexibility to configure a test method to be run in a multi-threaded environment. This is achieved by configuring it while using the @Test annotation on a method.  
  
IndependentTest.java

public class IndependentTest

{

@Test(threadPoolSize = 3, invocationCount = 6, timeOut = 1000)

public void testMethod()

{

Long id = Thread.currentThread().getId();

System.out.println("Test method executing on thread with id: " + id);

}

}  
  
The method is configured to run in multi-threaded mode by using the threadPoolSize attribute along with the Test annotation. The value of the threadPoolSize is set to 3; this configures the test method to be run in three different threads.

The other two attributes, invocationCount and timeOut, configure the test to be invoked a multiple number of times and fail if the execution takes more time.

-----------------------------------------------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*------------------------------------------

 By Mithilesh Singh