**Parallelism**

1. Advantages of parallel tests execution

2. Run parallel methods

3. Run parallel tests

4. Run parallel classes

Parallelism or multi-threading in software terms is defined as the ability of the software, operating system, or program to execute multiple parts or sub-components of another program simultaneously.

TestNG allows the tests to run in parallel or multi-threaded mode. This means that based on the test suite configuration, different threads are started simultaneously and the test methods are executed in them.

This gives a user a lot of advantages over normal execution, mainly reduction in execution time and ability to verify a multi-threaded code.

Parallelism or multi-threaded execution can provide a lot of advantages to the users. The following are two:

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

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.

Let’s look at an example

**package** pack5;

**import** org.testng.annotations.AfterMethod;

**import** org.testng.annotations.BeforeMethod;

**import** org.testng.annotations.Test;

**public** **class** ParallelMethodTest {

@BeforeMethod

**public** **void** before\_Method() {

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

System.***out***.println("Inside before\_Method. Thread id is: " + id);

}

@Test

**public** **void** test\_Method\_One() {

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

System.***out***.println("Inside test\_Method\_One. Thread id is: " + id);

}

@Test

**public** **void** test\_Method\_Two() {

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

System.***out***.println("Inside test\_Method\_Two. Thread id is: " + id);

}

@AfterMethod

**public** **void** after\_Method() {

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

System.***out***.println("Inside after\_Method. Thread id is: " + id);

}

}

testing.xml for single thread

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

<test name=*"Test\_Method\_Suite"*>

<classes>

<class name=*"pack5.ParallelMethodTest"*></class>

</classes>

</test> <!-- Test -->

</suite> <!-- Suite -->

Output:

[RemoteTestNG] detected TestNG version 6.14.2

Inside before\_Method. Thread id is: 1

Inside test\_Method\_One. Thread id is: 1

Inside after\_Method. Thread id is: 1

Inside before\_Method. Thread id is: 1

Inside test\_Method\_Two. Thread id is: 1

Inside after\_Method. Thread id is: 1

===============================================

Suite

Total tests run: 2, Failures: 0, Skips: 0

===============================================

Note: Thread ID for all is same above output.

We can run tests in parallel at ‘tests’, ‘classes’ and ‘methods’ level

**package** pack5;

**import** org.testng.annotations.AfterClass;

**import** org.testng.annotations.AfterMethod;

**import** org.testng.annotations.AfterSuite;

**import** org.testng.annotations.AfterTest;

**import** org.testng.annotations.BeforeClass;

**import** org.testng.annotations.BeforeMethod;

**import** org.testng.annotations.BeforeSuite;

**import** org.testng.annotations.BeforeTest;

**import** org.testng.annotations.Test;

**public** **class** ParallelTest\_Cat {

/\* @BeforeSuite

public void before\_Suite\_of\_Cat() {

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

System.out.println();

System.out.println("Inside before\_Suite of Cat. Thread id is: " + id);

}

@BeforeTest

public void before\_Test\_of\_Cat() {

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

System.out.println("Inside before\_Test of Cat. Thread id is: " + id);

}

@BeforeClass

public void before\_Class\_of\_Cat() {

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

System.out.println("Inside before\_Class of Cat. Thread id is: " + id);

System.out.println();

}

@BeforeMethod

public void before\_Method\_of\_Cat() {

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

System.out.println("Inside before\_Method of Cat. Thread id is: " + id);

}

\*/

@Test()

**public** **void** test\_Cat1() {

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

System.***out***.println("Inside test\_Cat1. Thread id is: " + id);

}

@Test()

**public** **void** test\_Cat2() {

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

System.***out***.println("Inside test\_Cat2. Thread id is: " + id);

}

/\* @AfterMethod

public void after\_Method\_of\_Cat() {

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

System.out.println("Inside after\_Method of Cat. Thread id is: " + id);

System.out.println();

}

@AfterClass

public void after\_Class\_of\_Cat() {

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

System.out.println("Inside after\_Class of Cat. Thread id is: " + id);

}

@AfterTest

public void after\_Test\_of\_Cat() {

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

System.out.println("Inside after\_Test of Cat. Thread id is: " + id);

}

@AfterSuite

public void after\_Suite\_of\_Cat() {

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

System.out.println("Inside after\_Suite of Cat. Thread id is: " + id);

}

\*/

}

**package** pack5;

**import** org.testng.annotations.AfterClass;

**import** org.testng.annotations.AfterMethod;

**import** org.testng.annotations.AfterSuite;

**import** org.testng.annotations.AfterTest;

**import** org.testng.annotations.BeforeClass;

**import** org.testng.annotations.BeforeMethod;

**import** org.testng.annotations.BeforeSuite;

**import** org.testng.annotations.BeforeTest;

**import** org.testng.annotations.Test;

**public** **class** ParallelTest\_Mouse {

/\* @BeforeSuite

public void before\_Suite\_of\_Mouse() {

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

System.out.println();

System.out.println("Inside before\_Suite of Mouse. Thread id is: " + id);

}

@BeforeTest

public void before\_Test\_of\_Mouse() {

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

System.out.println("Inside before\_Test of Mouse. Thread id is: " + id);

}

@BeforeClass

public void before\_Class\_of\_Mouse() {

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

System.out.println("Inside before\_Class of Mouse. Thread id is: " + id);

System.out.println();

}

@BeforeMethod

public void before\_Method\_of\_Mouse() {

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

System.out.println("Inside before\_Method of Mouse. Thread id is: " + id);

}

\*/

@Test()

**public** **void** test\_Mouse1() {

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

System.***out***.println("Inside test\_Mouse1. Thread id is: " + id);

}

@Test()

**public** **void** test\_Mouse2() {

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

System.***out***.println("Inside test\_Mouse2. Thread id is: " + id);

}

/\* @Test()

public void test\_Mouse3() {

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

System.out.println("Inside test\_Mouse3. Thread id is: " + id);

}

\*/

/\* @AfterMethod

public void after\_Method\_of\_Mouse() {

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

System.out.println("Inside after\_Method of Mouse. Thread id is: " + id);

System.out.println();

}

@AfterClass

public void after\_Class\_of\_Mouse() {

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

System.out.println("Inside after\_Class of Mouse. Thread id is: " + id);

}

@AfterTest

public void after\_Test\_of\_Mouse() {

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

System.out.println("Inside after\_Test of Mouse. Thread id is: " + id);

}

@AfterSuite

public void after\_Suite\_of\_Mouse() {

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

System.out.println("Inside after\_Suite of Mouse. Thread id is: " + id);

}

\*/}

**package** pack5;

**import** org.testng.annotations.AfterClass;

**import** org.testng.annotations.AfterMethod;

**import** org.testng.annotations.AfterSuite;

**import** org.testng.annotations.AfterTest;

**import** org.testng.annotations.BeforeClass;

**import** org.testng.annotations.BeforeMethod;

**import** org.testng.annotations.BeforeSuite;

**import** org.testng.annotations.BeforeTest;

**import** org.testng.annotations.Test;

**public** **class** ParallelTest\_Dog {

/\* @BeforeSuite

public void before\_Suite\_of\_Dog() {

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

System.out.println();

System.out.println("Inside before\_Suite of Dog. Thread id is: " + id);

}

@BeforeTest

public void before\_Test\_of\_Dog() {

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

System.out.println("Inside before\_Test of Dog. Thread id is: " + id);

}

@BeforeClass

public void before\_Class\_of\_Dog() {

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

System.out.println("Inside before\_Class of Dog. Thread id is: " + id);

System.out.println();

}

@BeforeMethod

public void before\_Method\_of\_Dog() {

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

System.out.println("Inside before\_Method of Dog. Thread id is: " + id);

}

\*/

@Test()

**public** **void** test\_Dog1() {

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

System.***out***.println("Inside test\_Dog1. Thread id is: " + id);

}

@Test()

**public** **void** test\_Dog2() {

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

System.***out***.println("Inside test\_Dog2. Thread id is: " + id);

}

/\* @AfterMethod

public void after\_Method\_of\_Dog() {

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

System.out.println("Inside after\_Method of Dog. Thread id is: " + id);

System.out.println();

}

@AfterClass

public void after\_Class\_of\_Dog() {

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

System.out.println("Inside after\_Class of Dog. Thread id is: " + id);

}

@AfterTest

public void after\_Test\_of\_Dog() {

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

System.out.println("Inside after\_Test of Dog. Thread id is: " + id);

}

@AfterSuite

public void after\_Suite\_of\_Dog() {

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

System.out.println("Inside after\_Suite of Dog. Thread id is: " + id);

}

\*/}

There are three class files:

ParallelTest\_Cat

test\_Cat1

test\_Cat2

ParallelTest\_Mouse

test\_Mouse1

test\_Mouse2

ParallelTest\_Dog

test\_Dog1

test\_Dog2

Each class file has

@BeforeSuite

@BeforeTest

@BeforeClass

@BeforeMethod

Testing.xml

The parallel attribute on the <suite> tag can take one of following values:

<suite name="My suite" parallel="methods">

<suite name="My suite" parallel="tests">

<suite name="My suite" parallel="classes">

<suite name="My suite" parallel="instances">

1. parallel="methods": TestNG will run all your test methods in separate threads. Dependent methods will also run in separate threads but they will respect the order that you specified.
2. parallel="tests": TestNG will run all the methods in the same <test> tag in the same thread, but each <test> tag will be in a separate thread. This allows you to group all your classes that are not thread safe in the same <test> and guarantee they will all run in the same thread while taking advantage of TestNG using as many threads as possible to run your tests.
3. parallel="classes": TestNG will run all the methods in the same class in the same thread, but each class will be run in a separate thread.
4. parallel="instances": TestNG will run all the methods in the same instance in the same thread, but two methods on two different instances will be running in different threads.

Example of parallel=*"methods" at Suite level*

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name=*"Suite"* parallel=*"methods"*>

<test name=*"Test\_Mouse"*>

<classes>

<class name=*"pack5.ParallelTest\_Mouse"*></class>

<class name=*"pack5.ParallelTest\_Dog"*></class>

</classes>

</test>

<test name=*"Test\_Cat"*>

<classes>

<class name=*"pack5.ParallelTest\_Cat"*></class>

</classes>

</test>

</suite> <!-- Suite -->

Below output shows the all methods have different ids

[RemoteTestNG] detected TestNG version 6.14.2

Inside test\_Mouse1. Thread id is: 13

Inside test\_Dog1. Thread id is: 14

Inside test\_Mouse2. Thread id is: 15

Inside test\_Dog2. Thread id is: 16

Inside test\_Cat1. Thread id is: 17

Inside test\_Cat2. Thread id is: 18

===============================================

Suite

Total tests run: 6, Failures: 0, Skips: 0

===============================================

Example of parallel=*"tests" at Suite level*

All methods in <test> tag will have same thread Id

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name=*"Suite"* parallel=*"tests"*>

<test name=*"Test\_Mouse"*>

<classes>

<class name=*"pack5.ParallelTest\_Mouse"*></class>

<class name=*"pack5.ParallelTest\_Dog"*></class>

</classes>

</test>

<test name=*"Test\_Cat"*>

<classes>

<class name=*"pack5.ParallelTest\_Cat"*></class>

</classes>

</test>

</suite> <!-- Suite -->

Cat methods have id 14 and Mouse & Dog methods have id 13

[RemoteTestNG] detected TestNG version 6.14.2

Inside test\_Mouse1. Thread id is: 13

Inside test\_Cat1. Thread id is: 14

Inside test\_Cat2. Thread id is: 14

Inside test\_Dog1. Thread id is: 13

Inside test\_Mouse2. Thread id is: 13

Inside test\_Dog2. Thread id is: 13

===============================================

Suite

Total tests run: 6, Failures: 0, Skips: 0

===============================================

Example of parallel=*"classes" at Suite level*

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name=*"Suite"* parallel=*"classes"*>

<test name=*"Test\_Mouse"*>

<classes>

<class name=*"pack5.ParallelTest\_Mouse"*></class>

<class name=*"pack5.ParallelTest\_Dog"*></class>

</classes>

</test>

<test name=*"Test\_Cat"*>

<classes>

<class name=*"pack5.ParallelTest\_Cat"*></class>

</classes>

</test>

</suite> <!-- Suite -->

[RemoteTestNG] detected TestNG version 6.14.2

Inside test\_Mouse1. Thread id is: 14

Inside test\_Dog1. Thread id is: 13

Inside test\_Mouse2. Thread id is: 14

Inside test\_Dog2. Thread id is: 13

Inside test\_Cat1. Thread id is: 15

Inside test\_Cat2. Thread id is: 15

===============================================

Suite

Total tests run: 6, Failures: 0, Skips: 0

===============================================

Configuring an independent test method to run in multiple threads

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.

**package** pack5;

**import** org.testng.annotations.Test;

**public** **class** IndependentTest {

@Test(threadPoolSize = 3, invocationCount = 4)

**public** **void** ThreadTest() {

System.***out***.println("Thread ID = " + Thread.*currentThread*().getId());

}

}

Testing.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

<test name=*"Test\_Mouse"*>

<classes>

<class name=*"pack5.IndependentTest"*></class>

</classes>

</test>

</suite> <!-- Suite -->

Output:

[RemoteTestNG] detected TestNG version 6.14.2

Thread ID = 13

Thread ID = 14

Thread ID = 15

Thread ID = 13

===============================================

Suite

Total tests run: 4, Failures: 0, Skips: 0

===============================================

We have created a test class, which contains a test method that is configured

to run in multithreaded or parallel mode. The test method is executed multiple times based on the invocationCount attribute value. Each execution is done in a separate thread that is clearly visible from the test report output. This feature is useful when you want to run only a fixed number of test methods in multithreaded mode and not the whole test suite.