



Test NG

Class 2

Agenda

Priority in TestNG

Enable/ Disable in TestNG

Assertion in TestNG

Hard Assertion vs Soft Assertion

TestNG XML file

Test Case Prioritization in TestNG

In TestNG, we can prioritize the Test Cases. Test case prioritization means, we can assign the priority to Test case.

Test case priority decides test case execution order. Higher prioritize test case execute first. So in TestNG “Priority” is used to schedule the test cases.

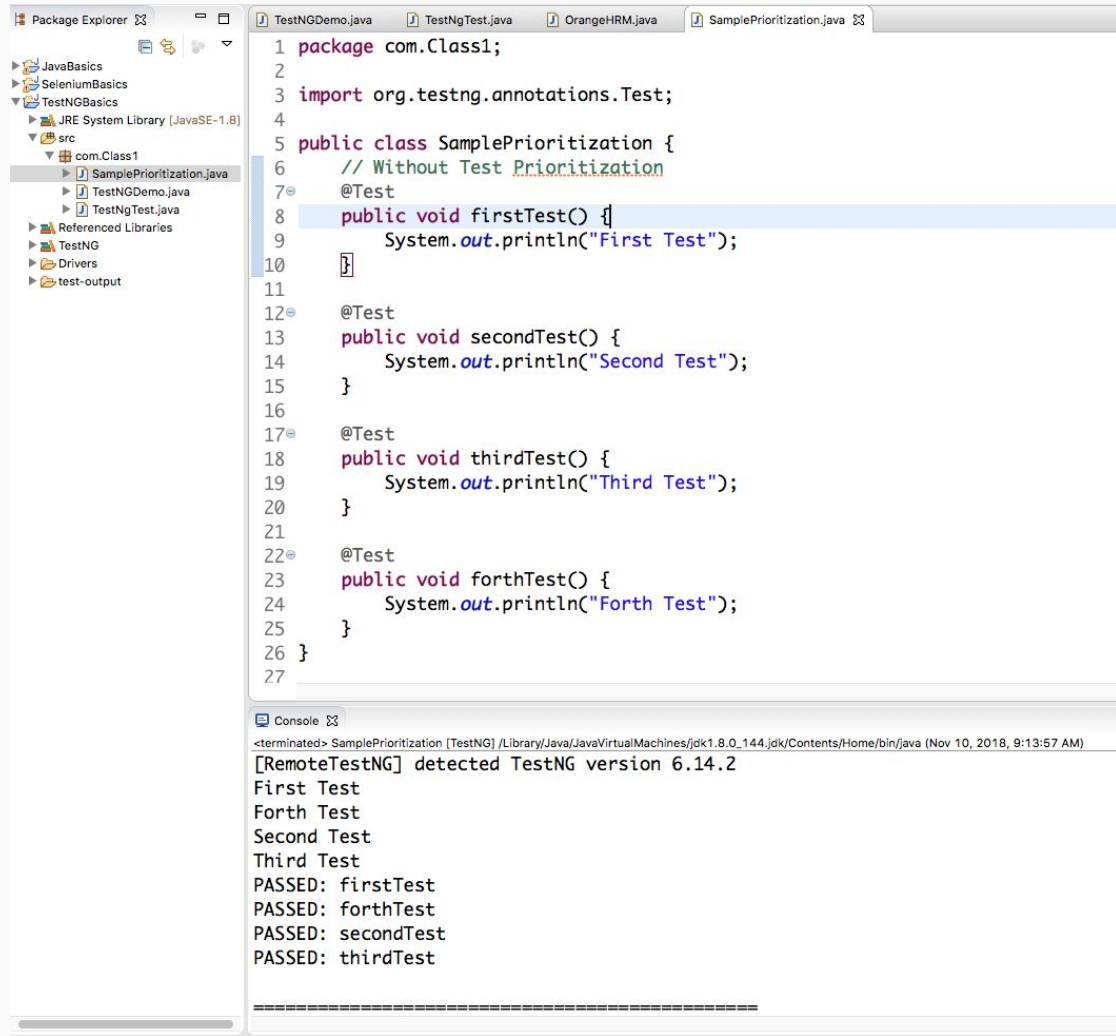
We can use this feature, When there are multiple test cases, we want to execute test cases in order.

We can declare the Priority at Method level as well as Class level. We use need to add annotation as `@Test(priority=)`.

The default value will be zero for priority.

If you don't mention the priority, it will take all the test cases as “priority=0” and execute.

Test Case Prioritization in TestNG



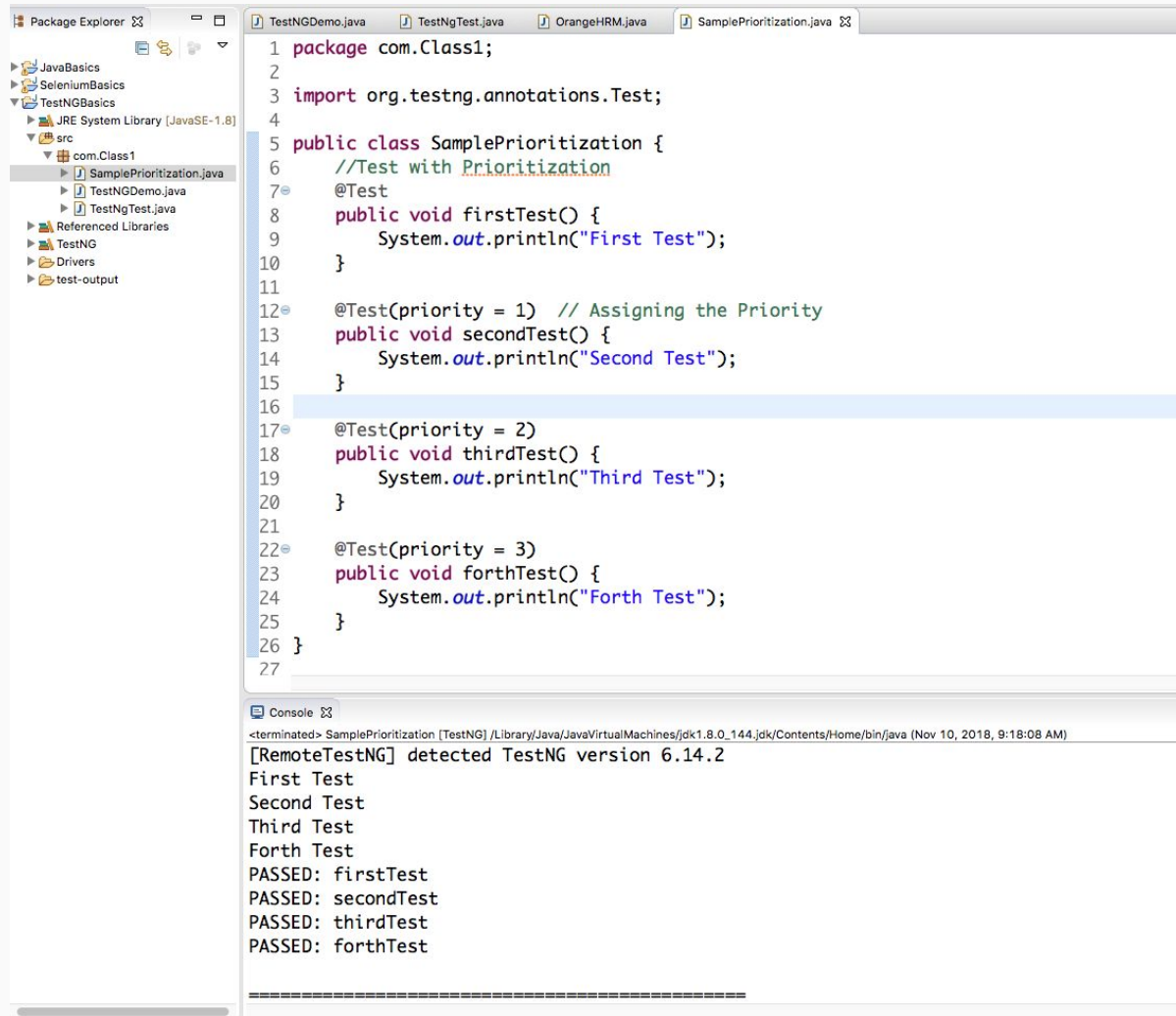
The screenshot displays an IDE with a Package Explorer on the left and a code editor on the right. The Package Explorer shows a project structure with folders like JavaBasics, SeleniumBasics, and TestNGBasics, and a file named SamplePrioritization.java. The code editor shows the following Java code:

```
1 package com.Class1;
2
3 import org.testng.annotations.Test;
4
5 public class SamplePrioritization {
6     // Without Test Prioritization
7     @Test
8     public void firstTest() {
9         System.out.println("First Test");
10    }
11
12    @Test
13    public void secondTest() {
14        System.out.println("Second Test");
15    }
16
17    @Test
18    public void thirdTest() {
19        System.out.println("Third Test");
20    }
21
22    @Test
23    public void forthTest() {
24        System.out.println("Forth Test");
25    }
26 }
27
```

The Console window at the bottom shows the following output:

```
<terminated> SamplePrioritization [TestNG] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Nov 10, 2018, 9:13:57 AM)
[RemoteTestNG] detected TestNG version 6.14.2
First Test
Forth Test
Second Test
Third Test
PASSED: firstTest
PASSED: forthTest
PASSED: secondTest
PASSED: thirdTest
=====
```

Test Case Prioritization in TestNG



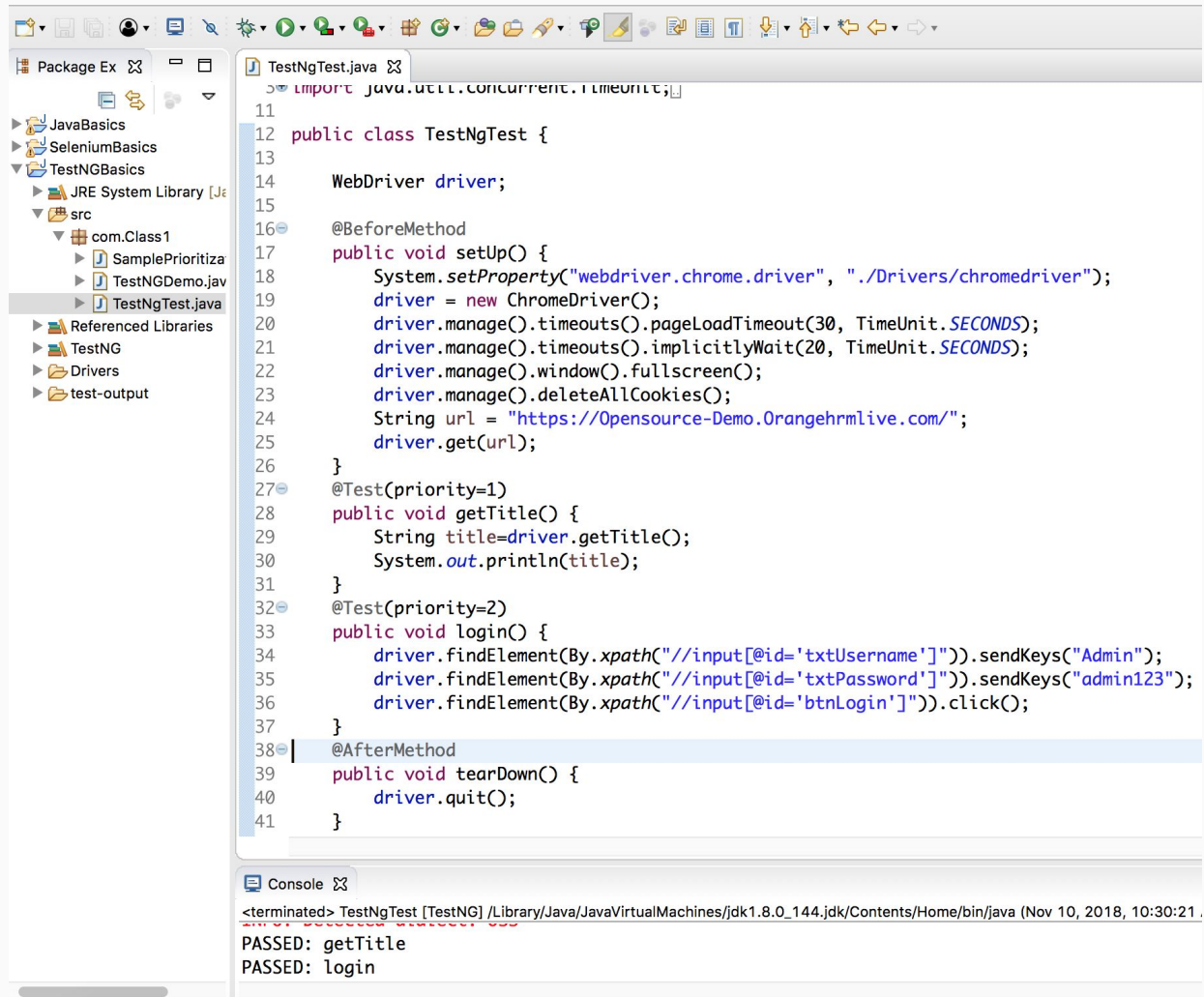
The screenshot displays an IDE with a Package Explorer on the left and a code editor on the right. The Package Explorer shows a project structure with 'TestNGBasics' containing 'src' and 'com.Class1'. The code editor shows the 'SamplePrioritization.java' file with the following code:

```
1 package com.Class1;
2
3 import org.testng.annotations.Test;
4
5 public class SamplePrioritization {
6     //Test with Prioritization
7     @Test
8     public void firstTest() {
9         System.out.println("First Test");
10    }
11
12    @Test(priority = 1) // Assigning the Priority
13    public void secondTest() {
14        System.out.println("Second Test");
15    }
16
17    @Test(priority = 2)
18    public void thirdTest() {
19        System.out.println("Third Test");
20    }
21
22    @Test(priority = 3)
23    public void forthTest() {
24        System.out.println("Forth Test");
25    }
26 }
27
```

The Console window at the bottom shows the following output:

```
<terminated> SamplePrioritization [TestNG] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Nov 10, 2018, 9:18:08 AM)
[RemoteTestNG] detected TestNG version 6.14.2
First Test
Second Test
Third Test
Forth Test
PASSED: firstTest
PASSED: secondTest
PASSED: thirdTest
PASSED: forthTest
=====
```

Test Case Prioritization in TestNG



The screenshot displays an IDE with a project structure on the left and a code editor on the right. The project structure includes:

- Package Ex
- JavaBasics
- SeleniumBasics
- TestNGBasics
 - JRE System Library [J]
 - src
 - com.Class1
 - SamplePrioritiza
 - TestNGDemo.jav
 - TestNgTest.java
- Referenced Libraries
- TestNG
- Drivers
- test-output

The code editor shows the following Java code in `TestNgTest.java`:

```
11 import java.util.concurrent.TimeUnit;
12 public class TestNgTest {
13
14     WebDriver driver;
15
16     @BeforeMethod
17     public void setUp() {
18         System.setProperty("webdriver.chrome.driver", "./Drivers/chromedriver");
19         driver = new ChromeDriver();
20         driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS);
21         driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
22         driver.manage().window().fullscreen();
23         driver.manage().deleteAllCookies();
24         String url = "https://Opensource-Demo.Orangehrmlive.com/";
25         driver.get(url);
26     }
27     @Test(priority=1)
28     public void getTitle() {
29         String title=driver.getTitle();
30         System.out.println(title);
31     }
32     @Test(priority=2)
33     public void login() {
34         driver.findElement(By.xpath("//input[@id='txtUsername']")).sendKeys("Admin");
35         driver.findElement(By.xpath("//input[@id='txtPassword']")).sendKeys("admin123");
36         driver.findElement(By.xpath("//input[@id='btnLogin']")).click();
37     }
38     @AfterMethod
39     public void tearDown() {
40         driver.quit();
41     }
42 }
```

The console output at the bottom shows the execution results:

```
<terminated> TestNgTest [TestNG] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Nov 10, 2018, 10:30:21 )
PASSED: getTitle
PASSED: login
```

Enable / Disable in TestNG

In some situations, we may need to avoid the execution of some of the tests or set of tests present in a specific test class.

For example, there might be situation where feature is already broken so we know it will fail and because of that you might want to ignore those test cases from the suite or from the test class.

When we denote any test method with the property of enabled with false value, then that method will not be part of test execution and TestNG will skip such methods.

By default, TestNG considers enabled=true, if you don't provide one

Enable / Disable in TestNG

The screenshot displays an IDE with a project structure on the left and a Java file named `*TestNgTest.java` in the center. The project structure includes `JavaBasics`, `SeleniumBasics`, and `TestNGBasics`. Under `TestNGBasics`, there is a `JRE System Library [Jre]` and a `src` folder containing `com.Class1`, `SamplePrioritiza`, `TestNGDemo.jav`, and `TestNgTest.java`. The `TestNgTest.java` file is selected, showing the following code:

```
11
12 public class TestNgTest {
13     WebDriver driver;
14     @BeforeMethod
15     public void setUp() {
16         System.setProperty("webdriver.chrome.driver", "./Drivers/chromedriver");
17         driver = new ChromeDriver();
18         driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS);
19         driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
20         driver.manage().window().fullscreen();
21         driver.manage().deleteAllCookies();
22         String url = "https://Opensource-Demo.Orangehrmlive.com/";
23         driver.get(url);
24     }
25     @Test(enabled=false)
26     public void getTitle() {
27         String title=driver.getTitle();
28         System.out.println(title);
29     }
30     @Test
31     public void login() {
32         driver.findElement(By.xpath("//input[id='txtUsername']")).sendKeys("Admin");
33         driver.findElement(By.xpath("//input[id='txtPassword']")).sendKeys("admin123");
34         driver.findElement(By.xpath("//input[id='btnLogin']")).click();
35     }
36     @AfterMethod
37     public void tearDown() {
38         driver.quit();
39     }
}
```

The `Console` tab at the bottom shows the execution results:

```
<terminated> TestNgTest [TestNG] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Nov 10, 2018, 10:40:34
PASSED: login

=====

Default test
Tests run: 1, Failures: 0, Skips: 0
```


Task

TC 1: OrangeHRM Title Validation

TC 2: OrangeHRM Successful Login

<https://opensource-demo.orangehrmlive.com/>

Please make sure to use the following:
annotations:

@BeforeMethod

@AfterMethod

@Test

- What would you do if you do not want to execute any specific test case?
- What would you do if you want to execute any specific test case first?

Assertion in TestNG

The assertion in TestNG means “Trust but Verify” Assertion is performed to compare the Actual result with expected result. In TestNG assertion is performed with the help of Assert Class.

Assert helps us to verify the conditions of the test and decide whether the test has failed or passed. It is most popular and frequently used methods in TestNG while creating Selenium Scripts.

Suppose, In selenium, there will be the situation in the test where you need to check the presence of an element. All you need to do is to put an assert statement on to it to verify its existence.

What is Assertion and types of Assertion in TestNG

Asserts helps us to verify the conditions of the test and decide whether test has failed or passed. A test is considered successful ONLY if it is completed without throwing any exception.

There are two types of assertions present in the TestNG :

- Assert class / Hard Assert
- SoftAssert class / Verify

Hard Assert

Assert class is present in `org.testng.Assert` class in TestNG under Selenium webdriver, this assert also known as Hard assert.

In hard assertion tests immediately fail and stop executing while assertion fails.

Hard assert compares two items and results in pass or fail. But in-case if the assert is failed then the code after the assert statement will not be executed and because of this only we are calling it as Hard Assert.

All the method present in Assert class are static methods so we can access all the methods using class reference, so no need to create object

We may want to use a hard Assert to verify if we have logged in correctly and fail the test if we haven't as there is no point in proceeding further if the test if the precondition itself fails.

Soft Assert

SoftAssert in TestNG is a child of **Assert**, the SoftAssert not only asserts a given conditions, but also gives a chance to user to continue the execution after Assertion failure.

In soft assertion tests don't stop running even if an assertion condition fails, but the test itself is marked as a failed.

This is useful if we are doing multiple validations in a form and you may actually want to complete all the validations and fail the test once all validations are complete in case of failures.

But in soft assertion you need to add `assertAll()` method in the last statement to mark the test failure otherwise it will not mark the test fail in Test report.

`assertAll()` method collates all the failures and decides whether to fail the test or not at the end. This should be the last statement in your test, the program will not execute after `assertAll()` if the assertion fails.

Soft Assert

Soft Assert – Soft Assert collects errors during `@Test`. Soft Assert does not throw an exception when an assert fails and would continue with the next step after the assert statement.

If there is any exception and you want to throw it then you need to use `assertAll()` method as a last statement in the `@Test` and test suite again continue with next `@Test` as it is.

Methods of Assertion Class

Assert.assertEquals(actual, expected) assertion

- This assertion is useful to compare expected and actual values. If both values match then it's fine and will continue execution. But if it fails then immediately it will mark that specific test method as fail and exit from that test method.

Assert.assertNotEquals(actual, expected, message) assertion

- Its function is opposite to assertEquals assertion. Means if both sides values will not match then this assertion will pass else it will fail. Here you can write your own message for failure condition. Simplest example of Assert.assertNotEquals in selenium webdriver

Assert.assertTrue(condition) Assertion

- assertTrue assertion is generally used for Boolean condition true. It will pass if condition returns "true". If it will return false then it will fail and skip software test execution from that specific method.

Assert.assertFalse(condition) Assertion

- It will check boolean value returned by condition and will pass if returned value is "False". If returned value is pass then this assertion will fail and skip execution from current test method.

Assertion Demo

The screenshot shows an IDE with a project structure on the left and a code editor on the right. The project structure includes a package 'E' with sub-packages 'JavaBasics', 'SeleniumBasics', and 'TestNGBasics'. Under 'TestNGBasics', there is a 'src' directory containing 'com.Class1', which in turn contains 'SamplePrioritiz', 'TestNGDemo.java', and 'TestNgTest.java'. The 'TestNgTest.java' file is open in the editor, showing a Selenium WebDriver test class. The code includes imports for 'java.util.concurrent.TimeUnit', a 'WebDriver driver' field, and methods for setup, verification, login, and teardown. A comment in the 'getTitle' method indicates a test failure: '//passing wrond title to verify' (note the typo 'wrond'). The console output at the bottom shows the test results: 'PASSED: verifyLogo' and 'FAILED: getTitle' with an 'AssertionError' message stating 'expected [OrangeHRM1] but found [OrangeHRM]'.

```
Package E
├── JavaBasics
├── SeleniumBasics
└── TestNGBasics
    ├── JRE System Library [...]
    └── src
        ├── com.Class1
        │   ├── SamplePrioritiz
        │   ├── TestNGDemo.java
        │   └── TestNgTest.java
        ├── Referenced Libraries
        ├── TestNG
        ├── Drivers
        └── test-output
```

```
TestNgTest.java
14
15 import java.util.concurrent.TimeUnit;
16
17 public class TestNgTest {
18     WebDriver driver;
19     @BeforeMethod
20     public void setUp() {
21         System.setProperty("webdriver.chrome.driver", "./Drivers/chromedriver");
22         driver = new ChromeDriver();
23         driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS);
24         driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
25         driver.manage().window().fullscreen();
26         driver.manage().deleteAllCookies();
27         String url = "https://OpenSource-Demo.Orangehrmlive.com/";
28         driver.get(url);
29     }
30     @Test
31     public void verifyLogo() {
32         boolean displayed=driver.findElement(By.xpath("//img[contains(@src, 'logo')]")).isDisplayed();
33         Assert.assertTrue(displayed);
34     }
35     @Test
36     public void getTitle() {
37         String title=driver.getTitle();
38         Assert.assertEquals(title, "OrangeHRM1");//passing wrond title to verify
39     }
40     @Test(enabled=false)
41     public void login() {
42         driver.findElement(By.xpath("//input[@id='txtUsername']")).sendKeys("Admin");
43         driver.findElement(By.xpath("//input[@id='txtPassword']")).sendKeys("admin123");
44         driver.findElement(By.xpath("//input[@id='btnLogin']")).click();
45     }
46     @AfterMethod
47     public void tearDown() {
48         driver.quit();
49     }
50 }
```

Console

```
<terminated> TestNgTest [TestNG] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Nov 10, 2018, 1:53:15)
PASSED: verifyLogo
FAILED: getTitle
java.lang.AssertionError: expected [OrangeHRM1] but found [OrangeHRM]
```


testng.xml

Now supposing we have two/multiple classes to test software web application how will we run them?

To run multiple classes together we can run with help of **testng.xml** file.

testng.xml is an XML file that describes configuration of TestNG test suite. Using Testng.xml file we can run test available in one or more class file

testng.xml file is a configuration file in TestNG. It is used to define test suites and tests.

testng.xml file used to pass Parameters to the test methods.

testng.xml file provides different options to include packages, classes and independent test methods in our test suite.

testng.xml file allows us to configure multiple tests in a single test suite and run them in multi threaded environment.

Can we run test without using testng.xml file?

Yes but once our test become larger and requires configuration of multiple test at one place, the best option is **testng.xml**.

testng.xml

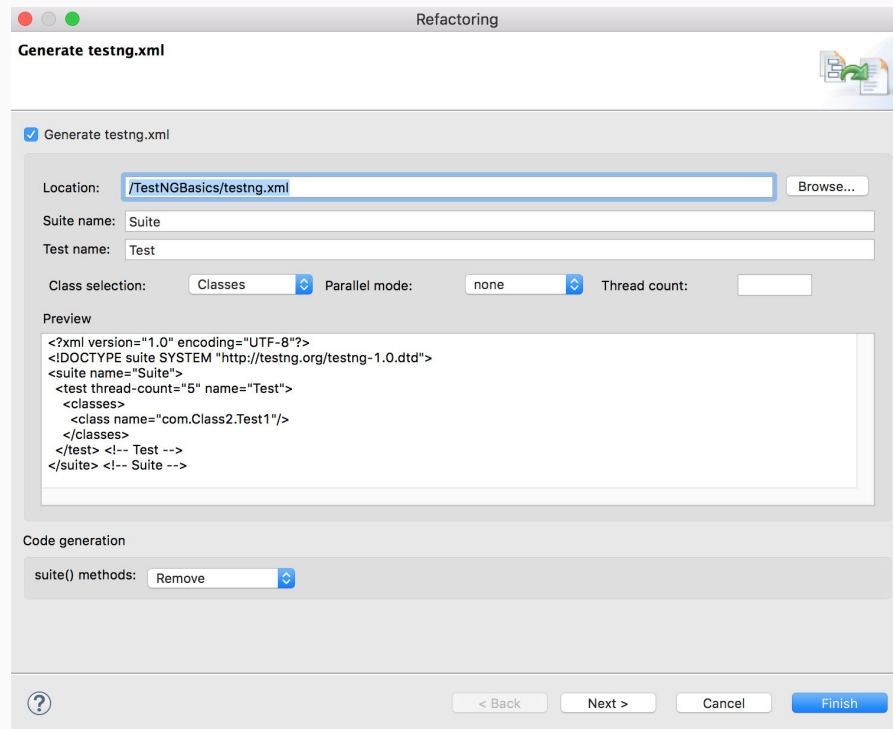
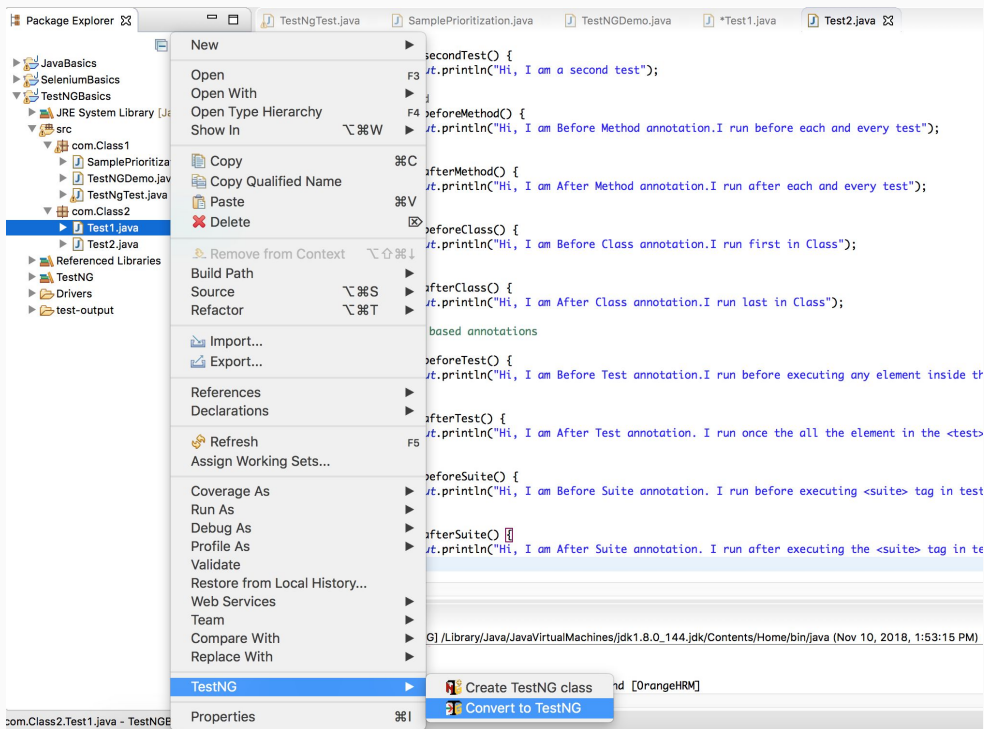
TestNG allows:

- Create tests with packages
- Create tests using classes
- Create tests using test methods
- Include/exclude a particular package, class, or test method
- Use of regular expression while using the include/exclude feature
- Store parameter values for passing to test methods at runtime
- Configure multi threaded execution options

Test Suite in TestNG

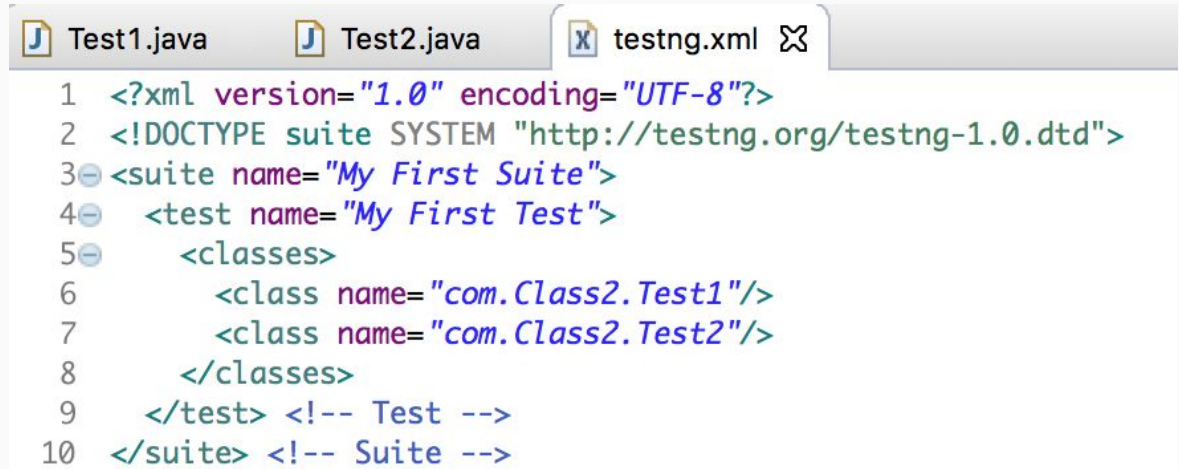
- Test Suite is something which is used to execute the test cases in a batch. We include many test cases in a single suite and run these test cases as the batch.
- So running a set of test cases together is call executing a Test Suite.
- In TestNG, we can create the test suite using **testng.xml file**. This testng.xml file is very important factor in TestNG.
- This file could be used for configuring your test run, set test dependency, include or exclude any test, method, class or package and set priority etc.
- We will see, How we can create this file and use it to make our execution process easier .
- you will end up to a place where you need to execute so many test cases on a run. Running a set of test cases together is call executing a **Test Suite**.
- Those test cases can be dependent to each other or may have to be executed in a certain order. TestNg gives us the capability to manage our test execution.

How to create testng.xml file



How to create testng.xml file

1. Configure testng.xml to run both classes in single test.

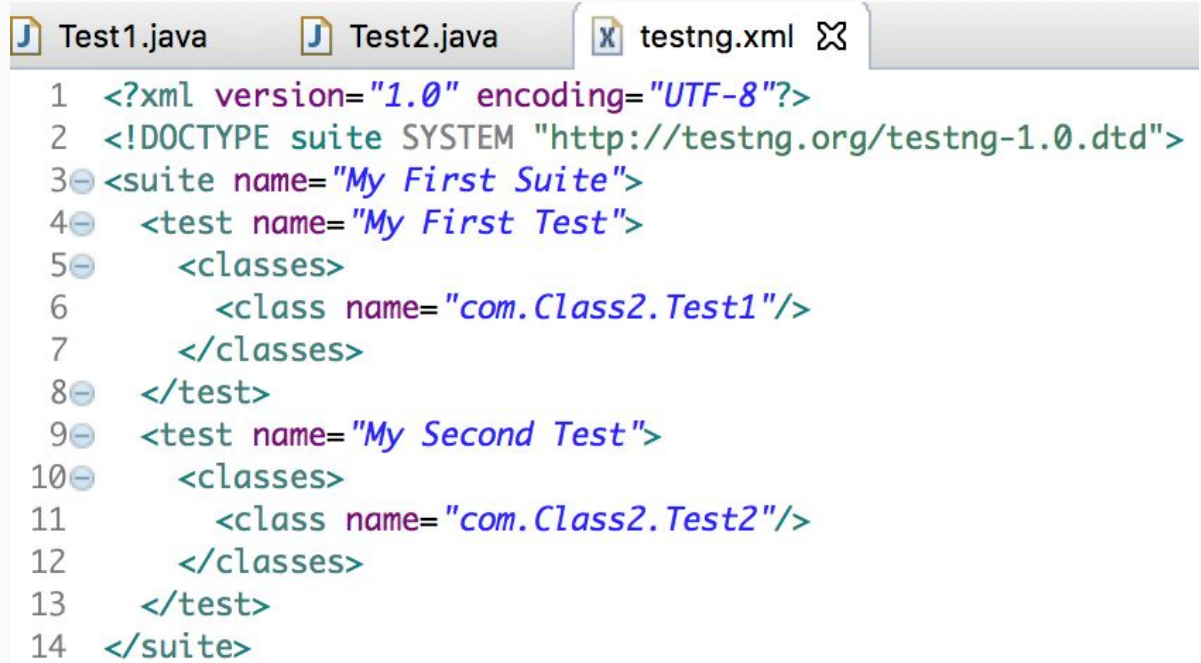


The screenshot shows an IDE with three tabs: Test1.java, Test2.java, and testng.xml. The testng.xml file is active and contains the following XML code:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3 <suite name="My First Suite">
4   <test name="My First Test">
5     <classes>
6       <class name="com.Class2.Test1"/>
7       <class name="com.Class2.Test2"/>
8     </classes>
9   </test> <!-- Test -->
10 </suite> <!-- Suite -->
```

How to create testng.xml file

2. Configure testng.xml to run both classes in single test.



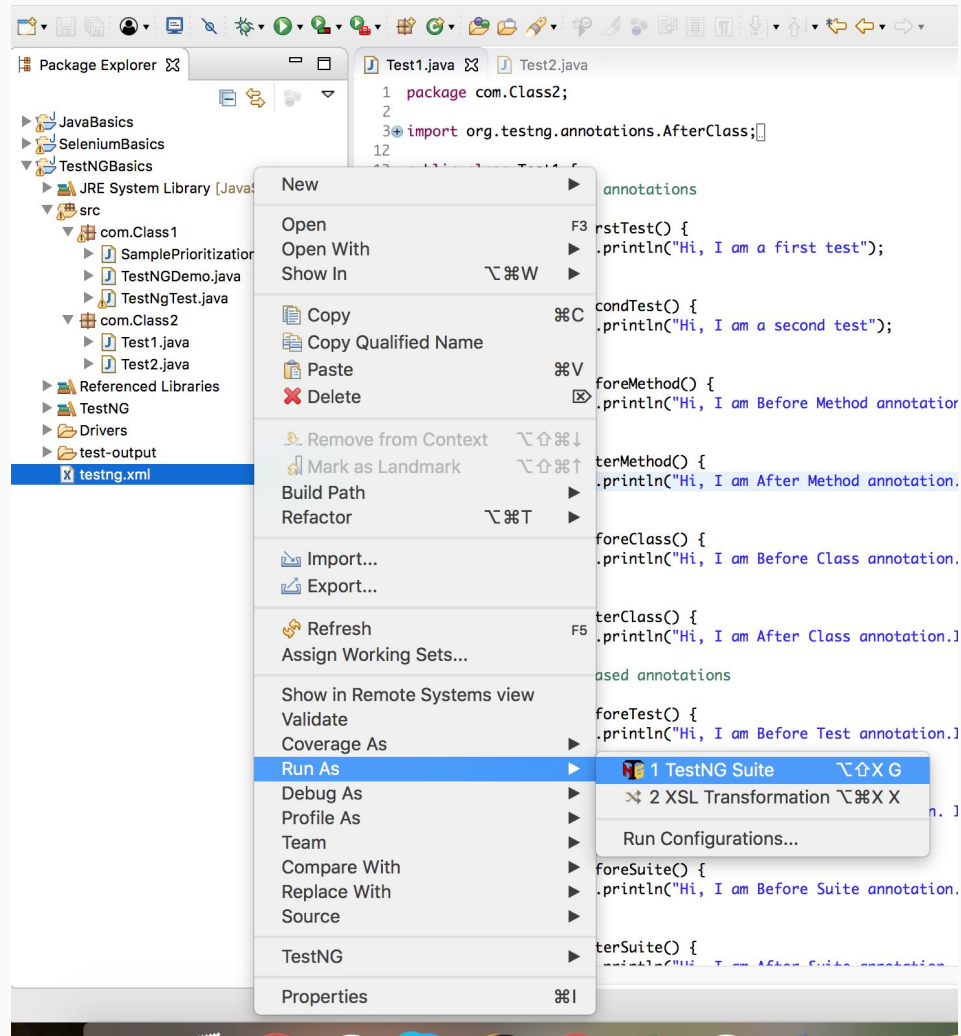
The screenshot shows an IDE with three tabs: Test1.java, Test2.java, and testng.xml. The testng.xml file is active and contains the following XML code:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3 <suite name="My First Suite">
4   <test name="My First Test">
5     <classes>
6       <class name="com.Class2.Test1"/>
7     </classes>
8   </test>
9   <test name="My Second Test">
10    <classes>
11      <class name="com.Class2.Test2"/>
12    </classes>
13  </test>
14 </suite>
```

How to create testng.xml file

- **<suite> : suite tag defines the TestNG suite.** You can give any name to your suite using 'name' attribute. In above given example, We have given "Suite One" to our test suite.
- **<test> : test tag defines the TestNG test.** You can give any name to your test using 'name' attribute. In above given example, We have given "Test One" to our test.
- **<classes> : classes tag defines multiple class.** We can multiple test class under classes tag. In our example we have only one class.
- **<class> : class tag defines the class name which you wants to consider in test execution.**

Executing testng.xml File



Task

Identify Priority of Test Cases

TC 1: OrangeHRM Verify Successful Login

Step 1: Open browser and navigate to OrangeHRM

Step 2: Enter valid UID and valid PWD and click login button

Step 3: Verify user successfully logged in

TC 2: OrangeHRM Add Employee

Step 1: Click on PIM link and Add Employee

Step 2: Provide Details and Save

Step 3: Verify Employee is added

TC 3: OrangeHRM Verify Employee Details

Step 1: Click on PIM link and Employee List

Step 2: Search for employee by it id

Step 3: Verify Employee details are displayed

Note: Create BeforeClass and AfterClass annotations to open browser and logout from the application