**TestNG Tutorials**

Contents

[1. TestNG Introduction and Configuration Setup 2](#_Toc72485147)

[2. TestNG Installation 2](#_Toc72485148)

[3. Annotations 2](#_Toc72485149)

[4. Priority 3](#_Toc72485150)

[5. Assertions 3](#_Toc72485151)

[6. Parameters 3](#_Toc72485152)

[7. Data Providers 3](#_Toc72485153)

[8. Data Provider with Excel 4](#_Toc72485154)

[9. Execute Failed Test Cases 5](#_Toc72485155)

[10. Cross Browser Testing 5](#_Toc72485156)

[11. Parallel Execution 6](#_Toc72485157)

[12. Groups 6](#_Toc72485158)

[13. Reports 7](#_Toc72485159)

[14. Reporter Log 7](#_Toc72485160)

[15. Listeners 7](#_Toc72485161)

[16. How to ignore test case 7](#_Toc72485162)

[17. DependsOnMethod 7](#_Toc72485163)

[18. Parellel 7](#_Toc72485164)

[19. expectedExceptions 8](#_Toc72485165)

# TestNG Tutorial

## TestNG Introduction and Configuration Setup

***TestNG*** is a testing framework inspired from ***JUnit*** and ***NUnit*** but introducing some new functionality that makes it more powerful and easier to use

## TestNG Installation

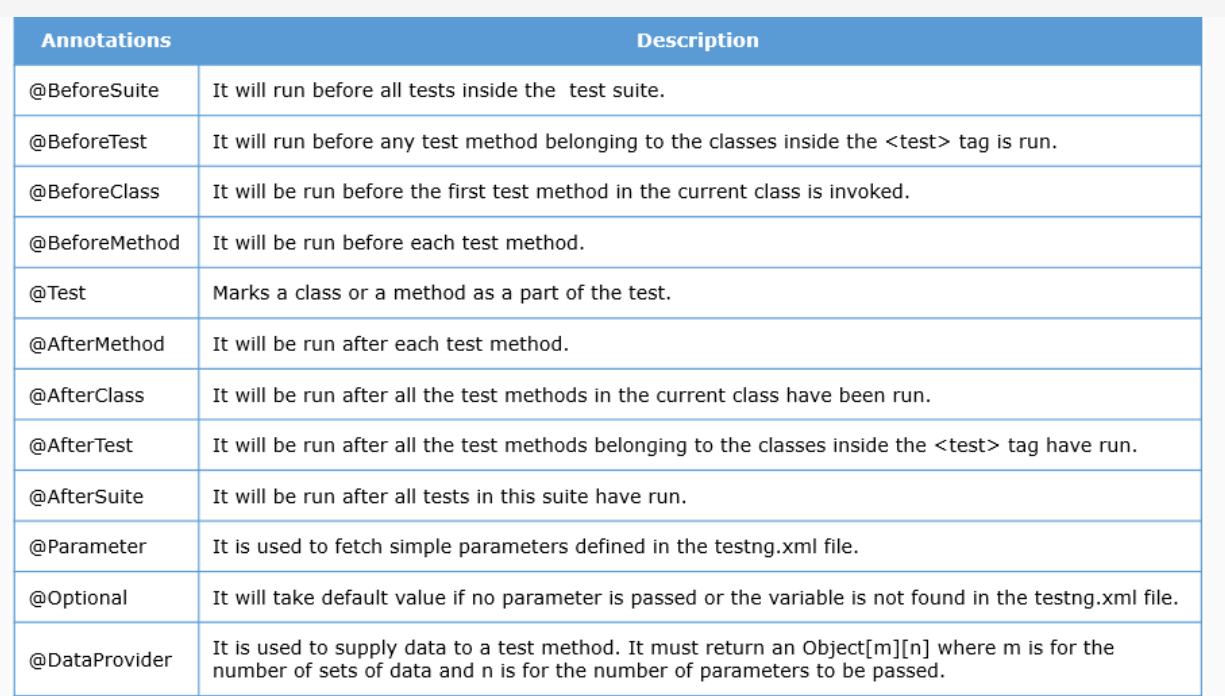
TestNG can be installed in two ways

1. Via Eclipse -> Help -> Marketplace and search for testing and install it
2. Via Eclipse -> Help -> Install New Software
   1. Click Add button
   2. Enter Name as “testNG” and give <https://testng.org/testng-eclipse-update-site> in Location
   3. Select TestNG and click install and continue

**Reference:**

<https://www.toolsqa.com/testng/install-testng/>

## Annotations

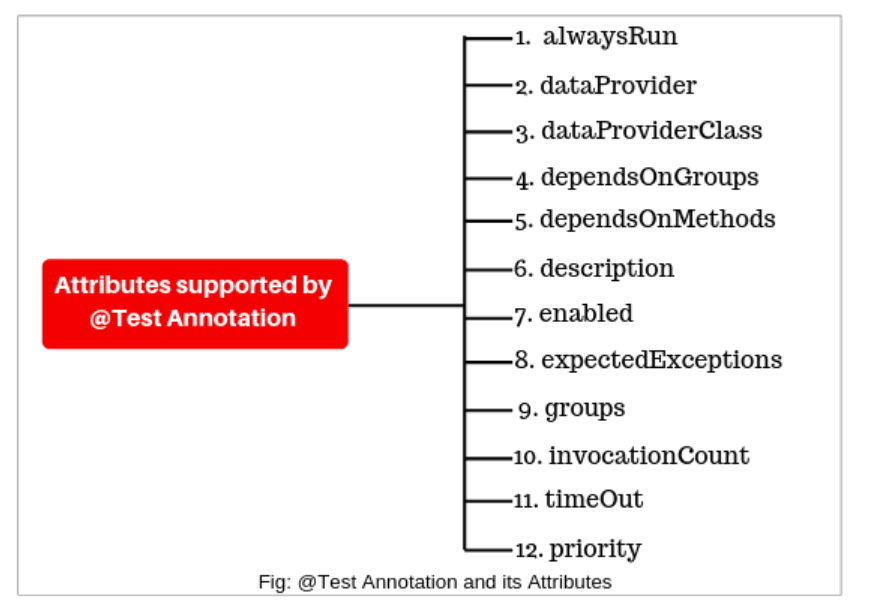


Order of Execution:

* 1. Suite - @BeforeSuite - 1 @AfterSuite - 12
  2. Test - @BeforeTest - 2, @AfterTest – 11

1. Class - @BeforeClass - 3, @AfterClass – 10
2. Method - @BeforeMethod – 4, 7, @AfterMethod – 6, 9
3. @Test 1 - 5 , (displayMessage1)
4. @Test 2 – 8 (displayMessage2)
5. -> **alphabetic order** of execution of @Test name of the methods

## Attributes in @Test method:



## Priority

@Test(priority=1)

Public void login(){

}

## Assertions

It is used to validate the test cases passed or failed

**Two Types:**

1. Hard Assert - Assert
2. Soft Assert - Verify

Assert.equals(actualResult, expectedResult)

## Parameters

 @Test

    @Parameters ({"val1", "val2"})

    public void Sum(int v1, int v2) {

     int finalsum = v1 + v2;

        System.out.println("The final sum of the given values is " + finalsum);

    }

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

<suite name="TestNG Parameters Suite">

   <test name="Params">

      <parameter name="val1" value="2" />

      <parameter name="val2" value="3" />

      <classes>

         <class name="Params" />

      </classes>

   </test>

</suite>

## Data Providers

@Test(dataProvider="SearchProvider")

public void testMethod(String author,String searchKey) throws InterruptedException{

{

WebElement searchText = driver.findElement(By.name("q"));

//search value in google searchbox

searchText.sendKeys(searchKey);

System.out.println("Welcome ->"+author+" Your search key is->"+searchKey);

Thread.sleep(3000);

String testValue = searchText.getAttribute("value");

System.out.println(testValue +"::::"+searchKey);

searchText.clear();

//Verify if the value in google search box is correct

Assert.assertTrue(testValue.equalsIgnoreCase(searchKey));

}

}

/\*\*

\* @return Object[][] where first column contains 'author'

\* and second column contains 'searchKey'

\*/

@DataProvider(name="SearchProvider")

public Object[][] getDataFromDataprovider(){

return new Object[][]

{

{ "Guru99", "India" },

{ "Krishna", "UK" },

{ "Bhupesh", "USA" }

};

}

## Data Provider with Excel

FileInputStream fs = new FileInputStream("D:\\DemoFile.xlsx");

//Creating a workbook

XSSFWorkbook workbook = new XSSFWorkbook(fs);

XSSFSheet sheet = workbook.getSheetAt(0);

Row row = sheet.getRow(0);

Cell cell = row.getCell(0);

System.out.println(sheet.getRow(0).getCell(0));

Row row1 = sheet.getRow(1);

Cell cell1 = row1.getCell(1);

System.out.println(sheet.getRow(0).getCell(1));

Row row2 = sheet.getRow(1);

Cell cell2 = row2.getCell(1);

System.out.println(sheet.getRow(1).getCell(0));

Row row3 = sheet.getRow(1);

Cell cell3 = row3.getCell(1);

System.out.println(sheet.getRow(1).getCell(1));

//String cellval = cell.getStringCellValue();

//System.out.println(cellval);

}

}

## Reports

1. Index.html
2. Emailable.html

## Execute Failed Test Cases

Refresh the project folder

Look for “testing-failed.xml” file and execute it again after fixing the error in the failed test cases

## Cross Browser Testing

|  |
| --- |
| public class MultiBrowser {    public WebDriver driver;      @Parameters("browser")      @BeforeClass      // Passing Browser parameter from TestNG xml      public void beforeTest(String browser) {      // If the browser is Firefox, then do this      if(browser.equalsIgnoreCase("firefox")) {    //Initializing the firefox driver (Gecko)    driver = new FirefoxDriver();      }else if (browser.equalsIgnoreCase("chrome")) {      //Initialize the chrome driver      driver = new ChromeDriver();      }      // Enter the website address in the browser      driver.get("https://www.demoqa.com");      }      // Once Before method is completed, Test method will start      @Test public void login() throws InterruptedException {    driver.findElement(By.xpath("//\*[@id=\"app\"]/div/div/div[2]/div/div[1]/div/div[1]")).click();    }      @AfterClass public void afterTest() {    driver.quit();    }    } |

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

<suite name="Suite" parallel="none">

<test name="FirefoxTest">

<parameter name="browser" value="firefox" />

<classes>

<class name="MultiBrowser" />

</classes>

</test>

<test name="ChromeTest">

<parameter name="browser" value= "chrome" />

<classes>

<class name="MultiBrowser" />

</classes>

</test>

</suite>

## Parallel Execution

Parallel Execution can be done with following options

1. Methods
2. Classes
3. Tests

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

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

<suite name="softwaretestingmaterial" parallel="methods" thread-count="2">

<test name="testngTest">

<classes>

<class name="softwareTestingMaterial.ParallelTests" />

</classes>

</test>

</suite>

## Groups

*@Test* (groups = { "smokeTest", "functionalTest" })

TestNG.xml

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

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

<suite name="softwaretestingmaterial">

<test name="testngTest">

<groups>

<run>

<include name="smokeTest" />

</run>

</groups>

<classes>

<class name="softwareTestingMaterial.TestCase1" />

<class name="softwareTestingMaterial.TestCase2" />

</classes>

</test>

</suite>

## Reporter Log

## Listeners

## How to ignore test case

1. *@Test (enabled = false).*

## DependsOnMethod

import org.testng.annotations.Test;

public class DependsOnMethodsTestCase {

*@Test*(dependsOnMethods = {"testCase2"})

public void testCase1(){

System.out.println("Test Case 1");

}

*@Test*

public void testCase2(){

System.out.println("Test Case 2");

}

}

## expectedExceptions

|  |  |
| --- | --- |
| 3  4  5  6  7  8  9  10 | //package softwareTestingMaterial;  import org.testng.annotations.Test;  public class TestNGException {    *@Test*(expectedExceptions = ArithmeticException.class)  public void testException() {  System.out.println("SoftwareTestingMaterial.com");  int i = 1 / 0; // the arithmetic exception is handled here because of usage expectedExceptions in //@Test annotations)  }  } |
|  |  |

Timeouts

public class TimeoutSuite

{

    @Test

    public void timeTestOne() throws InterruptedException {

        Thread.sleep(1000);

        System.out.println("Time test method one");

    }

    @Test

    public void timeTestTwo() throws InterruptedException {

        Thread.sleep(400);

        System.out.println("Time test method two");

    }

}

TestNG.xml

<suite name="Time test Suite" time-out="500" verbose="1" >

  <test name="Timeout Test" >

    <classes>

      <class name="com.howtodoinjava.test.TimeoutSuite" />

    </classes>

  </test>

</suite>

## Junit vs TestNG

| **Basis of** | **JUnit** | **TestNG** |
| --- | --- | --- |
| Developed by | JUnit was developed by Kent Beck, David Saff, Erich Gamma. Erich Gamma, and Kris Vasudevan. | TestNG is a testing framework that was developed by Cédric Beust. |
| Open-Source | JUnit is an open-source framework used to trigger and write tests. | TestNG is a Java-based framework that is an upgraded option for running tests. |
| Parallel test Runs | JUnit does not support to run parallel tests. | TestNG can run parallel tests. |
| Supports Annotation | It does not support advanced annotation. | It supports advanced annotation. |
| Dependency tests | The dependency tests are missing in JUnit. | Dependency tests are present in TestNG. |
| Grouping tests | Grouping tests together is not possible in JUnit. | Tests can be grouped together and run parallel. |
| Ease of Use | Running tests need a certain dependency on JUnit. | Writing tests and configuring them is easy in TestNG than JUnit. |