https://www.tutorialspoint.com/testng/testng\_execution\_procedure.htm

Here is the list of annotations that TestNG supports −

|  |  |
| --- | --- |
| **Sr.No.** | **Annotation & Description** |
| 1 | **@BeforeSuite**  The annotated method will be run only once before all tests in this suite have run. |
| 2 | **@AfterSuite**  The annotated method will be run only once after all tests in this suite have run. |
| 3 | **@BeforeClass**  The annotated method will be run only once before the first test method in the current class is invoked. |
| 4 | **@AfterClass**  The annotated method will be run only once after all the test methods in the current class have run. |
| 5 | **@BeforeTest**  The annotated method will be run before any test method belonging to the classes inside the <test> tag is run. |
| 6 | **@AfterTest**  The annotated method will be run after all the test methods belonging to the classes inside the <test> tag have run. |
| 7 | **@BeforeMethod**  The annotated method will be run before each test method. |
| 18 | **@AfterMethod**  The annotated method will be run after each test method. |
| 9 | **@DataProvider**  Marks a method as supplying data for a test method. The annotated method must return an Object[ ][ ], where each Object[ ] can be assigned the parameter list of the test method. The @Test method that wants to receive data from this DataProvider needs to use a dataProvider name equals to the name of this annotation. |
| 10 | **@Factory**  Marks a method as a factory that returns objects that will be used by TestNG as Test classes. The method must return Object[ ]..  It is useful when you want to run multiple test classes through a single test class |
| 12 | **@Parameters**  Describes how to pass parameters to a @Test method. |
| 13 | **@Test**  Marks a class or a method as a part of the test. |

Benefits of Using Annotations

Following are some of the benefits of using annotations −

* TestNG identifies the methods it is interested in, by looking up annotations. Hence, method names are not restricted to any pattern or format.
* We can pass additional parameters to annotations.
* Annotations are strongly typed, so the compiler will flag any mistakes right away.

import org.testng.annotations.Test;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeSuite;

import org.testng.annotations.AfterSuite;

public class TestngAnnotation {

// test case 1

@Test

public void testCase1() {

System.out.println("in test case 1");

}

// test case 2

@Test

public void testCase2() {

System.out.println("in test case 2");

}

@BeforeMethod

public void beforeMethod() {

System.out.println("in beforeMethod");

}

@AfterMethod

public void afterMethod() {

System.out.println("in afterMethod");

}

@BeforeClass

public void beforeClass() {

System.out.println("in beforeClass");

}

@AfterClass

public void afterClass() {

System.out.println("in afterClass");

}

@BeforeTest

public void beforeTest() {

System.out.println("in beforeTest");

}

@AfterTest

public void afterTest() {

System.out.println("in afterTest");

}

@BeforeSuite

public void beforeSuite() {

System.out.println("in beforeSuite");

}

@AfterSuite

public void afterSuite() {

System.out.println("in afterSuite");

}

}

* First of all, beforeSuite() method is executed only once.
* Lastly, the afterSuite() method executes only once.
* Even the methods beforeTest(), beforeClass(), afterClass(), and afterTest() methods are executed only once.
* beforeMethod() method executes for each test case but before executing the test case.
* afterMethod() method executes for each test case but after executing the test case.
* In between beforeMethod() and afterMethod(), each test case executes.