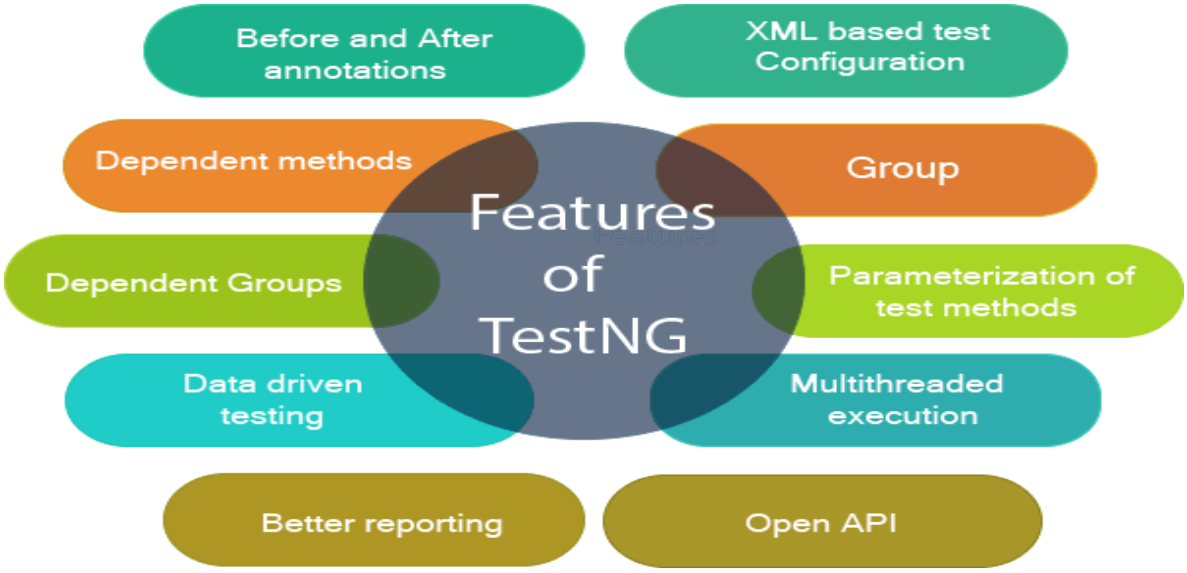
Testng
What is Testng?
Features of Testng
How to Install Testng?
Write first script in Testng
Define multiple tests in single class
Keywords
Annoatations
Testng Reports
Test suite
How to run failed test cases only
Listeners
Asseratation

What is TestNG

- TestNG is an open source automation testing framework where NG stands for "Next Generation".
- TestNG is inspired from JUnit and NUnit, which have some new functionalities that make it more powerful and easier to use.
- TestNG is an advanced framework that can be integrated with Selenium or any other automation tool.
- It provides multiple rich features for testing like assertions, reporting, parameterization, parallel test execution, etc.
- The creator of TestNG is **Cedric Beust.**



TestNG Features

- It gives the ability to produce **HTML Reports** of execution
- **Annotations** made testers life easy - @
- Testng have special keyswords/Parameters, for eg. Test cases can be **Grouped & Prioritized** more easily
- **Parallel** testing is possible
- Generates **Logs** through listeners
- Data **Parameterization** is possible
- No need of main method

How to Download and Install TestNG Framework in Eclipse?

a. **click on the Help option on the menu bar.**
b. **Choose the option “Eclipse Marketplace…”**

c. **type TestNG in the search box and click the Go button or press enter key.**
d. **Now click on the install button to install TestNG.**
e. **As soon as you click on the Install button, a new window will be open for feature selection but you do not have to change anything. Just click on the “Confirm” button.**
f. **In the next step, click on “I accept the terms of the license agreement” and then click on the “Finish” button.**
g. **After installation, restart Eclipse**

How to Install Testng?

Adding jars (7.9.0)

1. visit maven repo

2. search testng

3. copy and paste 4 line code

Install Testng plugin

1. Help >> Eclipse Marketplace

2. search with testng

3. Click on install

4. licence accept and trust all >> Finish

Write first script in Testng -

1. Write below code

```
@Test
public void abcd()
{
    sop("This is abcd method");
}
```

2. add testng libary - by mouse hover on @Test

3. Import dependency

4. Run as TestNg Test

Outputs -

Eclipse will going to provide two console for output

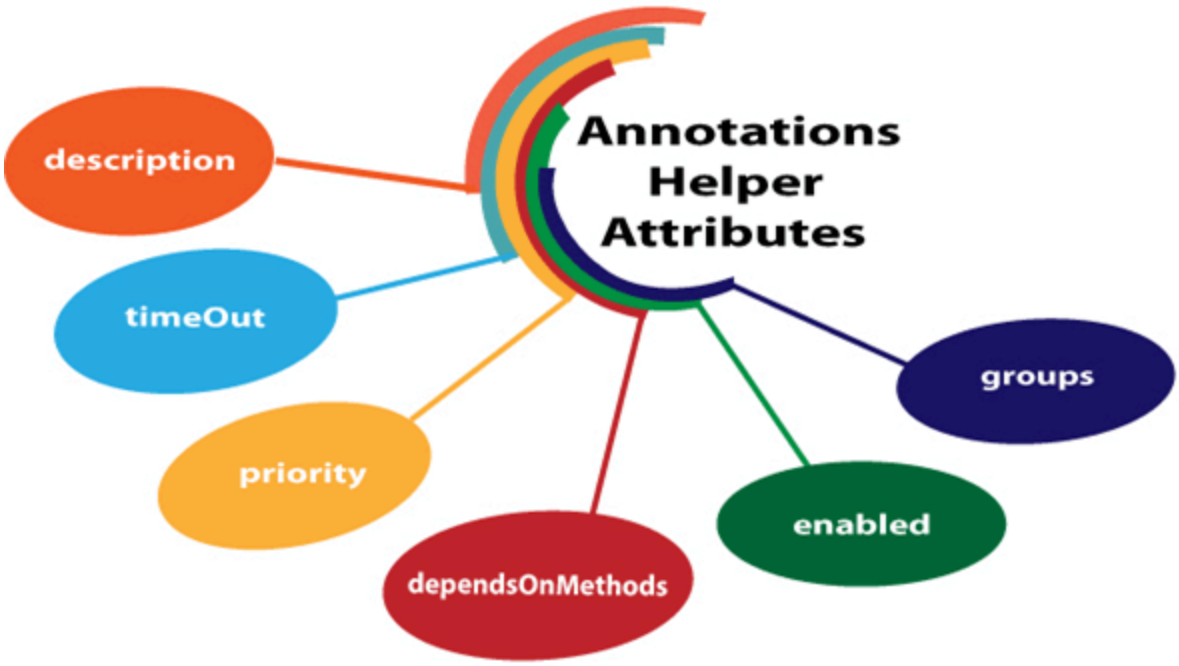
- Console Window - (Text report)
- Testng result window - (Graphical report)

TestNG Annotation Attributes

We can explicitly specify the attributes in a **@Test** annotation. Test attributes are the test specific, and they are specified at the right next to the **@Test** annotation.

```
@Test(attribute="value")
public void testcase2()
{
 System.out.println("This is testcase2");
}
```

Some of the common attributes are described below:



- **description**
- **timeOut**
- **priority**
- **dependsOnMethods**
- **enabled**
- **Groups**

priority

When no 'priority' attribute is specified then the TestNG will run the test cases in alphabetical order. Priority determines the sequence of the execution of the test cases. The priority can hold the integer values between -5000 and 5000. When the priority is set, the lowest priority test case will run first and the highest priority test case will be executed last.

enabled

The 'enabled' attribute contains the boolean value. By default, its value is **true**. If you want to skip some test method, then you need to explicitly specify '**false**' value.

description

It is a string which is attached to the @Test annotation that describes the information about the test.

```
public class Class1
{
@Test(description="This is testcase1")
public void testcase1()
{
System.out.println("HR");
}
}
```

dependsOnMethods

When the second test method wants to be dependent on the first test method, then this could be possible by the use of "**dependOnMethods**" attribute. If the first test method fails, then the dependent method on the first test method, i.e., the second test method will not run.

groups

The 'groups' attribute is used to group the different test cases that belong to the same functionality.

```
public class Software_Company
{
@Test(groups= {"software company"})
public void infosys()
{
System.out.println("Infosys");
```

```
}
@Test
public void technip()
{
System.out.println("Technip India Ltd");
}
@Test(groups= {"software company"})
public void wipro()
{
System.out.println("Wipro");
}
}
```

timeOut

If one of the test cases is taking a long time due to which other test cases are failing. To overcome such situation, you need to mark the test case as fail to avoid the failure of other test cases. The timeOut is a time period provided to the test case to completely execute its test case.

Annotations in TestNG

test code logic to control the flow of the execution of tests.
TestNG Annotation is a piece of code which is inside a program or business logic used to control the flow of execution of test methods.
A TestNG annotation starts from the symbol "**@**" and whatever follows is the annotation name.

- **@BeforeSuite** - The @BeforeSuite method in TestNG runs before the execution of all other test methods.
- **@AfterSuite** - The @AfterSuite method in TestNG runs after the execution of all other test methods.
- **@BeforeTest** - The @BeforeTest method in TestNG runs before the execution of all the test methods that are inside that folder.

- **@AfterTest** - The @AfterTest method in TestNG executes after the execution of all the test methods that are inside that folder.
- **@BeforeClass** - The @BeforeClass method in TestNG will run before the first method invokes of the current class.
- **@AfterClass** - The @AfterClass method in TestNG will execute after all the test methods of the current class execute.
- **@BeforeMethod** - The @BeforeMethod method in TestNG will execute before each test method.
- **@AfterMethod** - The @AfterMethod method in TestNG will run after each test method is executed.
- **@BeforeGroups** - The @BeforeGroups method in TestNG run before the test cases of that group execute. It executes just once.
- **@AfterGroups** - The @AfterGroups method in TestNG run after the test cases of that group execute. It executes only once.
- ***@Test**: The annotated method is a part of a test case.*

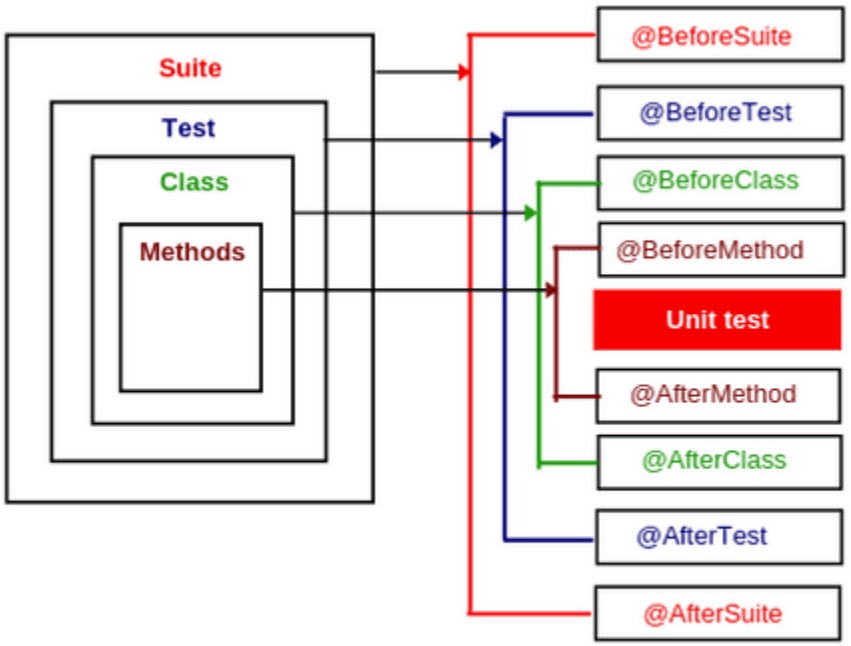


Fig: TestNG test life cycle & corresponding annotations

- **'TestNG Reports' Dashboard in Eclipse**
    - **Console Report -** short and simple, which denote the overall summary of the test.

    - **TestNG Report Section In Eclipse - Graphical view (**generates a more in-depth view than what we had in the console.)
    - It has default view that comes under the "***All Tests***" part. Also have ***Failed Tests*** and ***Summary***
- **How To Generate and View Emailable Report In TestNG?**
- **how To Generate and View Index Report In TestNG?**
    - Emailable reports is like summary reports
    - Index reports contains the index-like structure of different parts of the report, such as failed tests, test file, passed test, etc.
    - Emailable reports used to share test reports to other team members. It do not require any extra work from the tester, and they are a part of test execution. To generate emailable reports, first, run the TestNG test class if you have not already. Once we have run the test case, a new folder generates in the same directory with the name ***test-output***.

How To Use Reporter Class To Generate TestNG Reports?

Reporter class is an inbuilt class in TestNG. The reporter class in TestNG helps in storing the logs inside the reports that are user-generated or system-generated so that in the future, when we look at the report, we can view the logs directly from there rather than rerunning the test case.

To use the reporter class, we use the following syntax:

```
Reporter.log(string);
```

Simply we are calling the "***log***" function of the Reporter class of TestNG

Test suite

- the collection of ***TestNG Tests*** together is called a ***Test Suite***.
- A test suite can run multiple tests at once by executing the test suite
- these test cases are independent on each and executed in a specific order independently.

How To Create and Run TestNG Test Suite?

- Running a test suite in TestNG requires us to create a TestNG XML file
- we will be able to create and handle multiple test classes only with TestNG XML file
- the XML file will be the target file where we can configure our test run, set test dependency, include or exclude any test, method, class or package and set priority, etc.

1.Right-click on the Project folder, go to ***New*** and select ***File*** as shown in the below image.
2. In New file wizard, add filename as '*testng.xml*' and click on the ***Finish*** button.
3. Finally, it will add a ***testng.xml*** file under your project folder, and we are all set to write our first TestNG XML to run TestNG test suites.

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >
<suite name="TestSuite" >
  <test name="DemoTest" >
    <classes>
      <class name="package.class" />
    </classes>
  </test>

</suite>
```

- ***<suite>*** - The suite tag can be given any name and denotes the test suite name.
- ***<test>*** - The test tag can be given any name and indicates your test sets.
- ***<classes>*** - This is the combination of your package name and test case name and cannot write anything else.

We can select any name for your Test Suite & Test Name as per your requirement.
Now it's time to run the TestNG XML file we just created. Subsequently, run the test by right click on the *testng.xml* file and select ***Run As > TestNG Suite***.

TestNG Listeners

"***listeners***" means listening to something in the code
TestNG listeners are the piece of code that listens to the events occurring in the TestNG

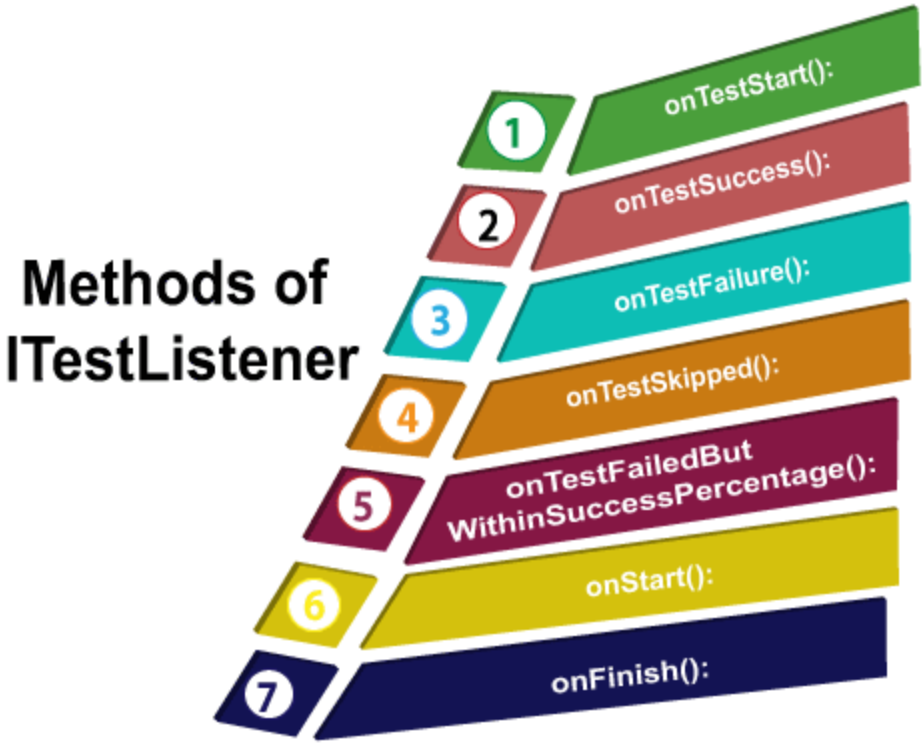
- Listeners are activated either before the test or after the test case.
- It is an interface that modifies the TestNG behavior.
- For example, when you are running a test case through selenium and suddenly a test case fails. We need a screenshot of the test case that has failed, to achieve such scenario, TestNG provides a mechanism, i.e., Listeners.

Types Of Listeners In TestNG

TestNG provides a bunch of listeners as a part of its testing environment. These listeners are as follows:

1. ***ITestListener***
2. *IReporter*
3. *ISuiteListener*
4. *IInvokedMethod*
5. *IHookable*
6. *IConfigurationListener*
7. *IConfigurableListener*
8. *IAnnotationTransformer*
9. *IExecution*
10. *IMethodInterceptor*

An ITestListener interface has the following methods:



1. **onTestStart():** An onTestStart() is invoked only when any test method gets started.
2. **onTestSuccess():** An onTestSuccess() method is executed on the success of a test method.
3. **onTestFailure():** An onTestFailure() method is invoked when test method fails.
4. **onTestSkipped():** An onTestSkipped() run only when any test method has been skipped.
5. **onTestFailedButWithinSuccessPercentage():** This method is invoked each time when the test method fails but within success percentage.
6. **onStart():** An onStart() method is executed on the start of any test method.
7. **onFinish():** An onFinish() is invoked when any test case finishes its execution

How to create the TestNG Listeners

Listeners can implement in two ways in TestNG:

- ***At the class level***: *Annotating listeners on each class in the test code.*
- ***At the suite level***: *Define the class names to implement listeners in the TestNG XML file*

```
package listeners;

import org.testng.Assert;
import org.testng.annotations.Listeners;
import org.testng.annotations.Test;


@Listeners(listeners.TestListener.class)
public class Sample {
   @Test
   public void sum()
   {
         int sum=0;
         int a=5;
         int b=7;
         sum=a+b;
         System.out.println(sum);
   }
   @Test
   public void testtofail()
   {
         System.out.println("Test case has been failed");
         Assert.assertTrue(false);
   }
}
```

```
package listeners;

import org.testng.ITestContext;
import org.testng.ITestListener;
import org.testng.ITestResult;

public class TestListener implements ITestListener{

   @Override
   public void onFinish(ITestContext context)
   {

   }

   @Override
   public void onStart(ITestContext context)
   {
```

```
    }

    @Override
    public void onTestFailedButWithinSuccessPercentage(ITestResult result)
    {

    }

    @Override
    public void onTestFailedWithTimeout(ITestResult result)
    {

    }

    @Override
    public void onTestFailure(ITestResult result)
    {
            System.out.println("failure of test cases and its details are : "+ result.getName());
    }

    @Override
    public void onTestSkipped(ITestResult result)
    {

    }

    @Override
    public void onTestStart(ITestResult result)
    {
            System.out.println("start of test cases and its details are : "+result.getName());
    }

    @Override
    public void onTestSuccess(ITestResult result)
    {
            System.out.println("Success of test cases and its details are : "+result.getName());
    }
}
```

- second way to use the within the suite.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite">
   <listeners>
                              <listener class-name="listeners.TestListener"/>
   </listeners>
  <test thread-count="5" name="Test">
          <classes>
          <class name="listeners.Sample"/>
          </classes>
  </test> <!-- Test -->
</suite> <!-- Suite -->
```

Note: When listeners are added to multiple classes then it could be an error prone. If listeners are implemented through the testng.xml file, then they are applied to all the classes.