## TestNG: it is a testing framework inspired from JUnit and NUnit, but introducing some new functionalities that make it more powerful and easier to use*.* System Requirement

|  |  |
| --- | --- |
| JDK | 1.5 or above. |
| Memory | No minimum requirement. |
| Disk Space | No minimum requirement. |
| Operating System | No minimum requirement. |

## Step 1 - Verify Java Installation in Your Machine

Open the console and execute a java command based on the operating system you have installed on your system.

|  |  |  |
| --- | --- | --- |
| **OS** | **Task** | **Command** |
| Windows | Open Command Console | c:\> java -version |
| Linux | Open Command Terminal | $ java -version |
| Mac | Open Terminal | machine:~ joseph$ java -version |

Let's verify the output for all the operating systems −

|  |  |
| --- | --- |
| **OS** | **Output** |
| Windows | java version "1.7.0\_25"  Java(TM) SE Runtime Environment (build 1.7.0\_25-b15)  Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed mode) |
| Linux | java version "1.7.0\_25"  Java(TM) SE Runtime Environment (build 1.7.0\_25-b15)  Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed mode) |
| Mac | java version "1.7.0\_25"  Java(TM) SE Runtime Environment (build 1.7.0\_25-b15)  Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed mode) |

If you do not have Java, install the Java Software Development Kit (SDK) from <https://www.oracle.com/technetwork/java/javase/downloads/index.html>. We are assuming Java 1.7.0\_25 as the installed version for this tutorial.

Suppose we have an application called Gmail, so the testing flow which actually happens in the company is like that

From developer side

Automate unit test case using unit Testing tools

Create unit Test Cases

Develop code To perform sign up

New User Registration

Develop code

Results Report

Automate unit test case using unit Testing tools

Create unit Test Cases

Develop code to perform sig in

Sign in

| | | |

| | | |

100 test cases

Inbox

C0llect all the unit automated test cases of all the modules and create a build

Install on the test server

.exe  
.zip  
.war

All Modules

Build software

Black box Testing

Smoke   
 Regression

TestNg:(Test Next Generation)  
\*\*\*\*Steps to install TestNg:-  
TestNg is available as plug-in in Eclipse

Step1:- Open Eclipse go to help option and choose go to Eclipse marketplace,

Step2:- Search for TestNg

Step3:-Click on install button available under TestNg for eclipse

Step4:-Click On Conform

Step5:- accept License agreement and click finish

---🡪 During installation click ok button in popup  
---🡪 finally Restart the Eclipse.

\*\*\*Steps to Integrate TestNg to Current Project…

* Right click on current project
* Go to the build path and choose configured build path
* Click on libraries
* Click on add libraries
* Select TestNg and click on next button
* Click on finish button
* Click on Ok button

\*\*Automatically TestNg Component is listed under project

Some important guideline:

Class Demo{

psvm(String[] args){

…………………..  
……………………….  
……………………………  
}  
}

For java execution it is necessary to present main method

Java

Simillary:

Auto method execution

Present inside TestNg class

It should be public and void

Define methods

@Test

TestNg

Need to present @test annotation

Steps to Create TestNg Class:-

Approach1:-Create java Class And Add @Test annotation

Approach2:-Right Click On Package go to TestNg and Choose Create TestNg class  
Provide class name and click on finish button.

Annotations of TestNg:

@Test annotation

* Annotation is define inside a class
* Annotation is always attach with the method
* Method should be public and it can be static or non static
* Method will not return any data or information therefore return type will be void
* Automatically the method will get executed through testNg
* After execution it will generate three different type of report
* 1. Eclipse console report
* 2.TestNg can console sole report
* 3. Html report or emailable report

\*emailable report present in test output folder.

\*Different annotations in TestNg:

|  |  |
| --- | --- |
| 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 | **@BeforeGroups**  The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked. |
| 8 | **@AfterGroups**  The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked. |
| 9 | **@BeforeMethod**  The annotated method will be run before each test method. |
| 10 | **@AfterMethod**  The annotated method will be run after each test method. |
| 11 | **@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. |
| 12 | **@Factory**  Marks a method as a factory that returns objects that will be used by TestNG as Test classes. The method must return Object[ ]. |
| 13 | **@Listeners**  Defines listeners on a test class. |
| 14 | **@Parameters**  Describes how to pass parameters to a @Test method. |
| 15 | **@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. * Test classes no longer need to extend anything   Execution procedure:  beforeSuite  beforeTest  beforeClass  beforeMethod  test   afterMethod  afterClass  afterTest  afterSuite  **Plug the TestNG with Eclipse:**   1. Download the latest version of TestNG jar file from [http://www.testng.org](http://www.testng.org/).  |  |  | | --- | --- | | **OS** | **Archive name** | | Windows | testng-6.8.jar |  * Open eclipse → right click on the project and go to property → Build Path → Configure Build Path and add the testng-6.8.jar in the libraries using *Add External Jar* button.   Add testng-6.8.jar in liraries.   * We assume that your Eclipse has inbuilt TestNG plug-in; if it is not available, then please get the latest version using the update site.   + In your Eclipse IDE, select *Help / Software updates / Find and* Install.   + Search for newfeaturesto install.   + Newremotesite*.*   + For Eclipse 3.4 and above, enter <http://beust.com/eclipse>.   + For Eclipse 3.3 and below, enter <http://beust.com/eclipse1>.   + Make sure the check box next to the URL is checked and click Next. |