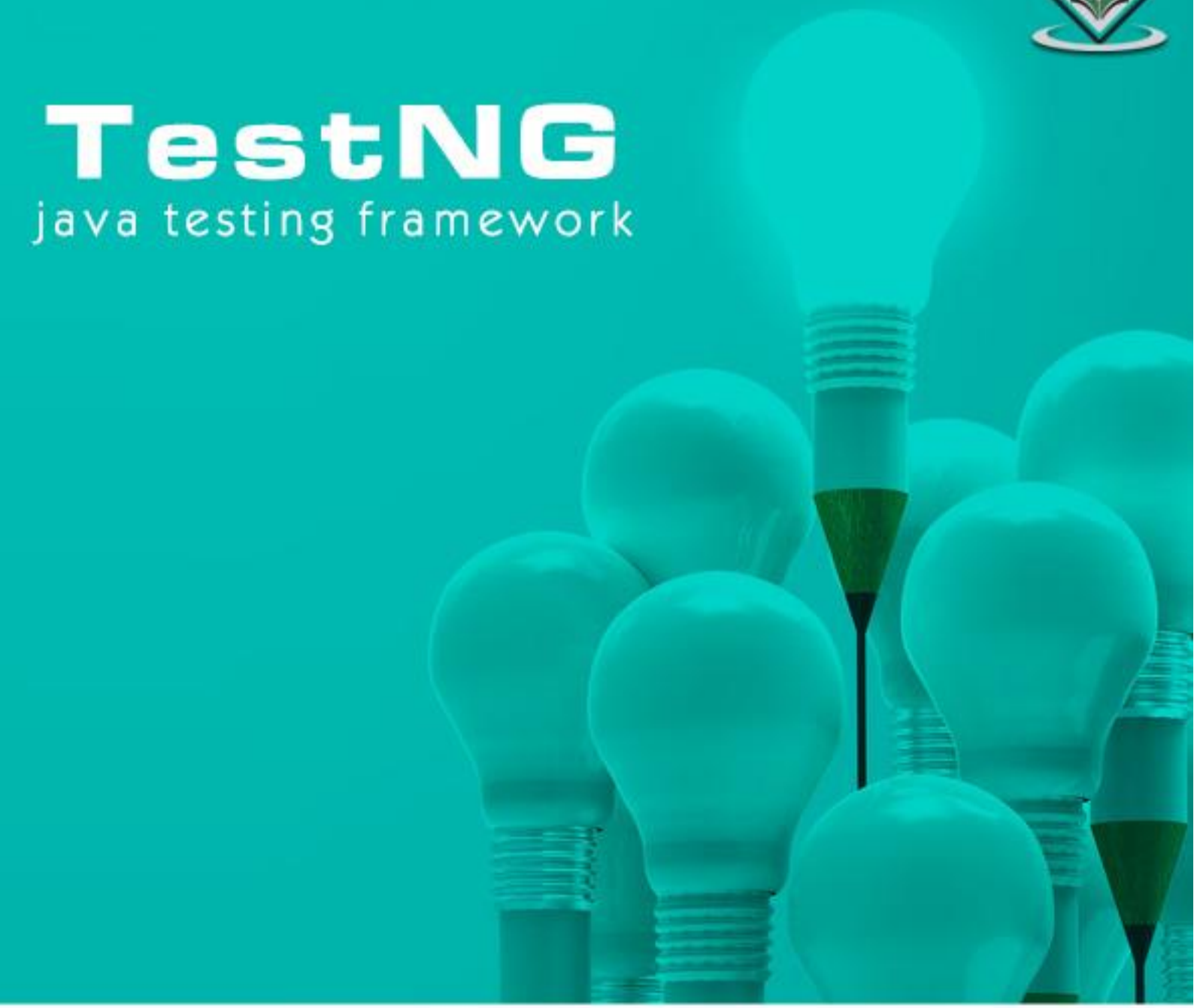
TestNG

java testing framework



tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

TestNG is a testing framework developed in the lines of JUnit and NUnit, however it introduces some new functionalities that make it more powerful and easier to use.

TestNG is designed to cover all categories of tests: unit, functional, end-to-end, integration, etc., and it requires JDK 5 or higher.

This tutorial provides a good understanding on TestNG framework needed to test an enterprise-level application to deliver it with robustness and reliability.

Audience

This tutorial is designed for software professionals interested in learning the features of TestNG Framework in simple and easy steps and implement it in practice.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding of Java programming language, text editor, and execution of programs, etc. As you are going to use TestNG to handle all levels of Java project testing, it will be helpful if you have a prior knowledge of software development and software testing processes.

Copyright & Disclaimer

Table of Contents

1. TESTNG – OVERVIEW

Testing is the process of checking the functionality of an application to ensure it works as per requirements. Unit testing comes into picture at the developer level where adequate measures are taken to test every single entity (class or method) to ensure the final product meets the requirements.

JUnit has driven developers to understand the usefulness of tests, especially of unit tests, when compared to any other testing framework. Leveraging a rather simple, pragmatic, and strict architecture, JUnit has been able to "infect" great number of developers. Do take a look at our tutorial on JUnit to have a good understanding of its features. JUnit, at the same time, has some shortcomings as well, which are listed below:

- Initially designed to enable unit testing only, now used for all kinds of testing.
- Cannot do dependency testing.
- Poor configuration control (setUp/tearDown).
- Intrusive (forces you to extend classes and name your methods a certain way).
- Static programming model (forces you to recompile unnecessarily).
- The management of different suites of tests in complex projects can be very tricky.

What is TestNG?

Definition of TestNG as per its documentation is as follows:

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

TestNG is an open source automated testing framework; where **NG** means **N**ext**G**eneration. TestNG is similar to JUnit (especially JUnit 4), but it is not a JUnit extension. It is inspired by JUnit. It is designed to be better than JUnit, especially when testing integrated classes. The creator of TestNG is *Cedric Beust*.

Eliminating most of the limitations of the older framework, TestNG gives the developer the ability to write more flexible and powerful tests. As it heavily borrows from Java Annotations (introduced with JDK 5.0) to define tests, it can also show you how to use this new feature of the Java language in a real production environment.

TestNG Features

- Supports annotations.
- TestNG uses more Java and OO features.

- Supports testing integrated classes (e.g., by default, no need to create a new test class instance for every test method).
- Separates compile-time test code from run-time configuration/data info.
- Flexible runtime configuration.
- Introduces ‘test groups’. Once you have compiled your tests, you can just ask TestNG to run all the "front-end" tests, or "fast", "slow", "database" tests, etc.
- Supports Dependent test methods, parallel testing, load testing, and partial failure.
- Flexible plug-in API.
- Support for multi-threaded testing.

2. TESTNG – ENVIRONMENT

TestNG is a framework for Java, so the very first requirement is to have JDK installed in your machine.

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 http://www.oracle.com/technetwork/java/javase/downloads/index.html. We are assuming Java 1.7.0_25 as the installed version for this tutorial.

Step 2: Set JAVA Environment

Set the **JAVA_HOME** environment variable to point to the base directory location, where Java is installed on your machine. For example,

OS	Output
Windows	Set the environment variable JAVA_HOME to C:\Program Files\Java\jdk1.7.0_25.
Linux	Export JAVA_HOME=/usr/local/java-current.
Mac	Export JAVA_HOME=/Library/Java/Home.

Append Java compiler location to System Path.

OS	Output
Windows	Append the string C:\Program Files\Java\jdk1.7.0_25\bin at the end of the system variable, Path.
Linux	Export PATH=$PATH:$JAVA_HOME/bin/
Mac	Not required

Verify Java Installation using the command **java -version** as explained above.

Step 3: Download TestNG Archive

Download the latest version of TestNG jar file from http://www.testng.org. At the time of writing this tutorial, we have downloaded *testng-6.8.jar* and copied it onto C:\>TestNG folder.

OS	Archive name
Windows	testng-6.8.jar
Linux	testng-6.8.jar
Mac	testng-6.8.jar

Step 4: Set TestNG Environment

Set the **TESTNG_HOME** environment variable to point to the base directory location, where TestNG jar is stored on your machine. The following table shows how to set the environment variable in Windows, Linux, and Mac, assuming that we've stored testng-6.8.jar at the location C:\>TestNG.

OS	Description
Windows	Set the environment variable TESTNG_HOME to C:\TESTNG.
Linux	Export TESTNG_HOME=/usr/local/TESTNG.
Mac	Export TESTNG_HOME=/Library/TESTNG.

Step 5: Set CLASSPATH Variable

Set the **CLASSPATH** environment variable to point to the TestNG jar location.

OS	Description
Windows	Set the environment variable CLASSPATH to %CLASSPATH%;%TESTNG_HOME%\testng-6.8.jar.
Linux	Export CLASSPATH=$CLASSPATH:$TESTNG_HOME/testng-6.8.jar.

Mac	export CLASSPATH=$CLASSPATH:$TESTNG_HOME/testng-6.8.jar.

Step 6: Test TestNG Setup

Create a java class file named TestNGSimpleTest at C:\>TestNG_WORKSPACE.

```
import org.testng.annotations.Test;
import static org.testng.Assert.assertEquals;

public class TestNGSimpleTest {
      @Test
      public void testAdd() {
            String str = "TestNG is working fine";
            assertEquals("TestNG is working fine", str);
      }
}
```

TestNG can be invoked in several different ways:

- With a testng.xml file.
- With ANT.
- From the command line.

Let us invoke using the testng.xml file. Create an xml file with the name testng.xml in C:\>TestNG_WORKSPACE to execute Test case(s).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="Suite1">
  <test name="test1">
    <classes>
       <class name="TestNGSimpleTest"/>
    </classes>
  </test>
</suite>
```

Step 7: Verify the Result

Compile the class using **javac** compiler as follows:

```
C:\TestNG_WORKSPACE>javac TestNGSimpleTest.java
```

Now, invoke the testng.xml to see the result:

```
C:\TestNG_WORKSPACE>java -cp "C:\TestNG_WORKSPACE" org.testng.TestNG testng.xml
```

Verify the output.

```
===============================================
Suite1
Total tests run: 1, Failures: 0, Skips: 0
===============================================
```

End of ebook preview

If you liked what you saw…

Buy it from our store @ **https://store.tutorialspoint.com**