Bottom of Form

**How to Execute Failed Test Cases in TestNG: Selenium WebDriver**

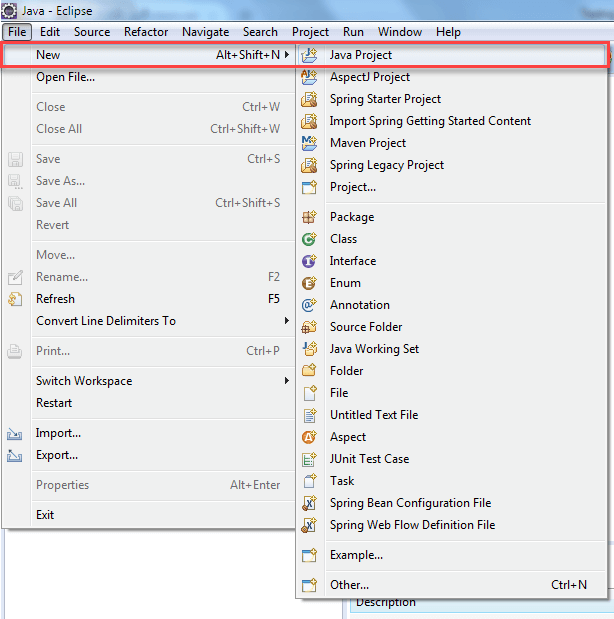
In this tutorial, you will learn-

* [Create a Selenium Project](https://www.guru99.com/run-failed-test-cases-in-testng.html#1)
* [Convert and Execute Selenium Project to TestNG](https://www.guru99.com/run-failed-test-cases-in-testng.html#2)
* [Execute TestNG via Command Line](https://www.guru99.com/run-failed-test-cases-in-testng.html#3)
* [How to run only failed test cases](https://www.guru99.com/run-failed-test-cases-in-testng.html#4)
* [Running testng-failed.xml file from Command line](https://www.guru99.com/run-failed-test-cases-in-testng.html#5)

**Create a Selenium Project**

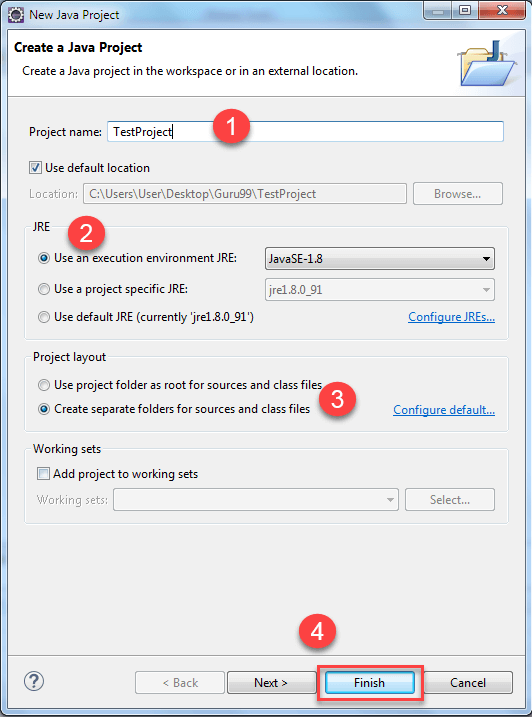
Prepare the Project

**Step 1)** In Eclipse, Create [Java](https://www.guru99.com/java-tutorial.html) project by clicking on new Java Project

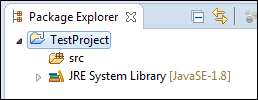
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor3.png)

**Step 2)** In this step,

1. Give project Name
2. Choose execution environment
3. Select Project layout option
4. Click on "Finish" button.

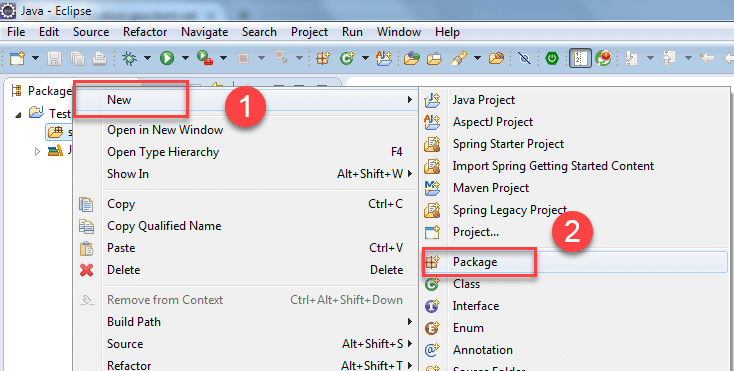
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor4.png)

When you click on finish button. The "TestProject" java project is created. The "TestProject" will look like this.

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor5.png)

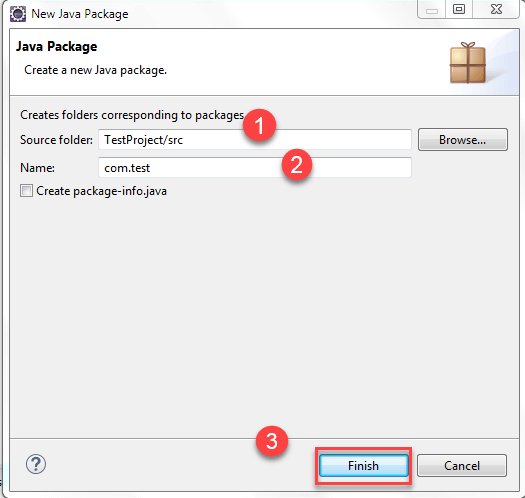
**Step 3)** Open that newly created project. You will able to see the "src" folder in Package Explorer.

1. Right click project and select "New"
2. Select option package

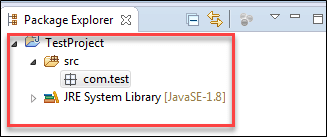
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor6.png)

**Step 4)** In this step,

1. Browse the src folder and select package
2. Give the package name and
3. Finally, click on finish button. (package Name->com.test)

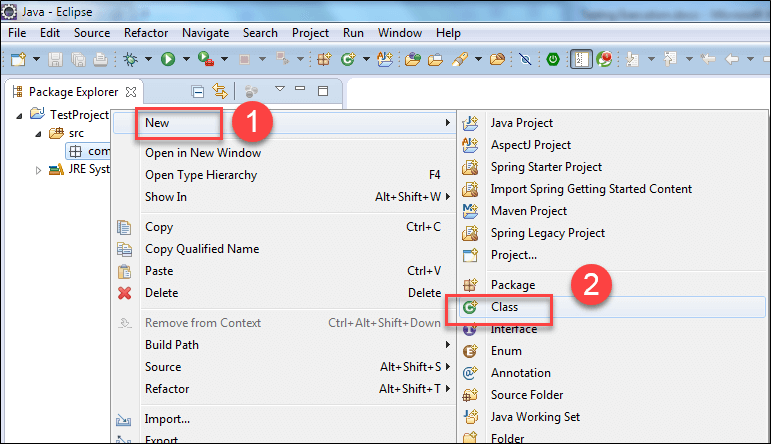
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor7.png)

After clicking on "finish" the project structure will look like this:

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor8.png)

**Step 5)** In this step,

1. Now, Right click on newly created package and
2. Select option "class."

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor9.png)

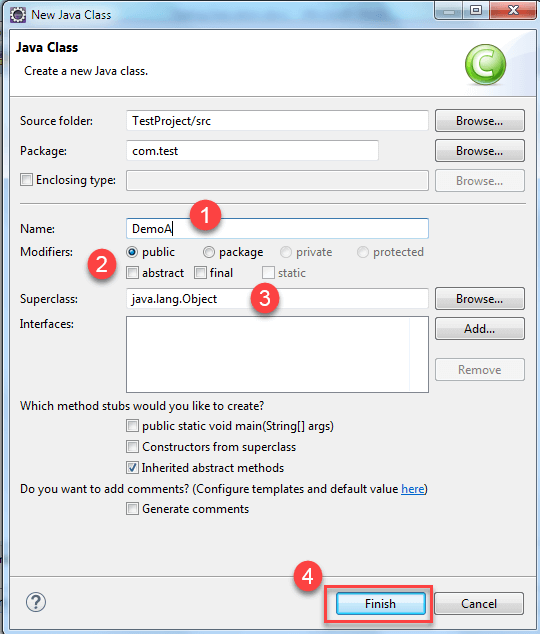
A new window will open.

**Step 6)** In this step,

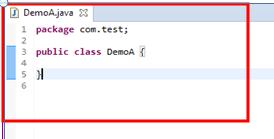
1. Give the "class" name
2. Select modifiers
3. Browse and select superclass java.lang.object
4. Click on finish button.

Here, you are creating two classes ex: **DemoA, DemoB**.

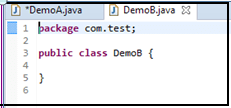
First Create Class DemoA.

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor10.png)

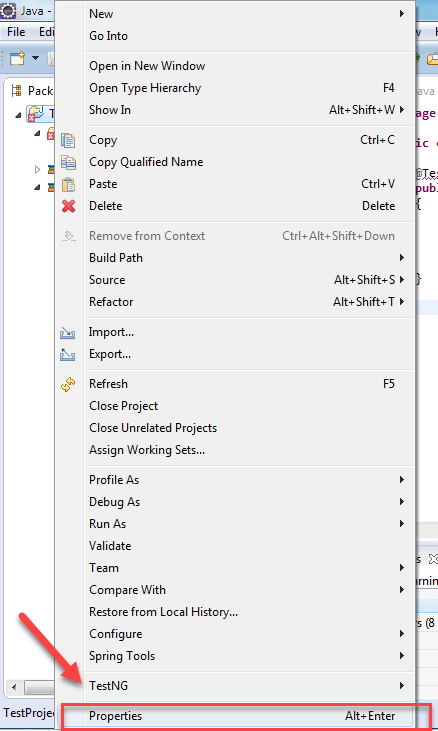
When you click on "Finish" Button. Then it will going to create class like this:

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor11.png)

Similarly, create class DemoB, When you click on "Finish" Button class will look like below-

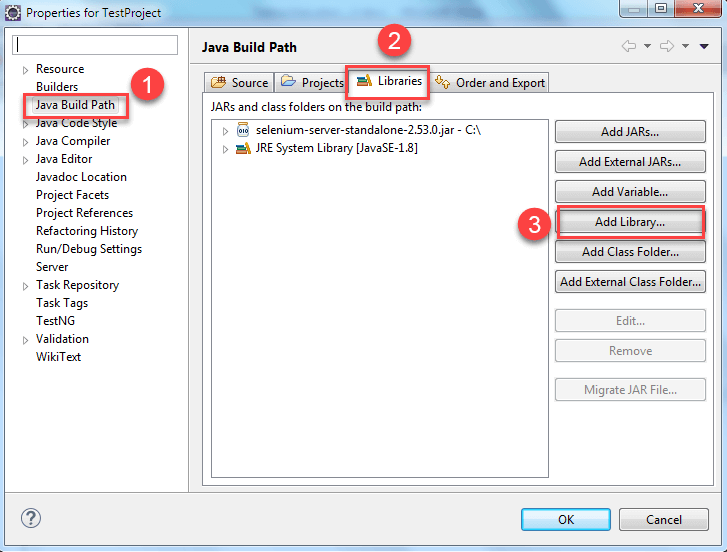
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor14.png)

**Step 7)** If you have installed TestNG library just click on Java Project--->Properties.

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor17.png)

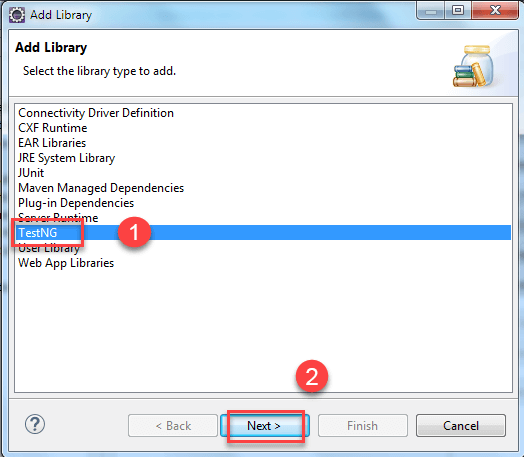
**Step 8)** Now in properties window,

1. Click on Java Build Path
2. Click on Libraries
3. Click on Add Library.

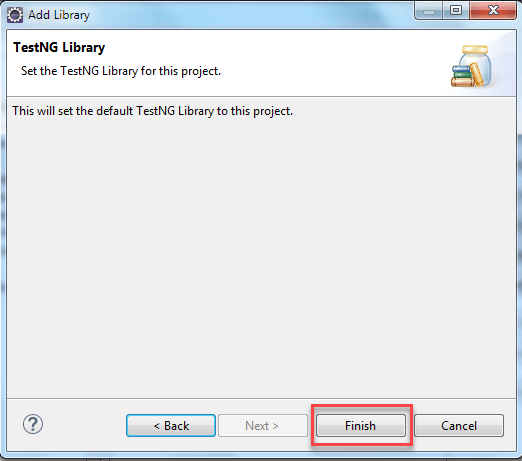
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor18.png)

Next

1. Click on "TestNg" and then
2. Click on "Next".

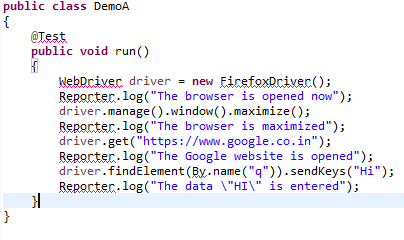
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor19.png)

Then Click on "finish" button.

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor20.png)

After this, write the following code.

**For Class DemoA**

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor21.png)

**Code Explanation:**

* Driver.manage().window().maximize();

This line will going to maximize the Browser window.

* driver.get(https://www.google.co.in);

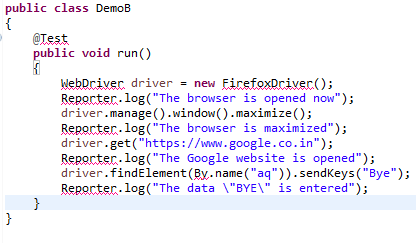
This line will enter the specified URL in the URL field.

* driver.findElement(By.name("q")).sendKeys("Hi");

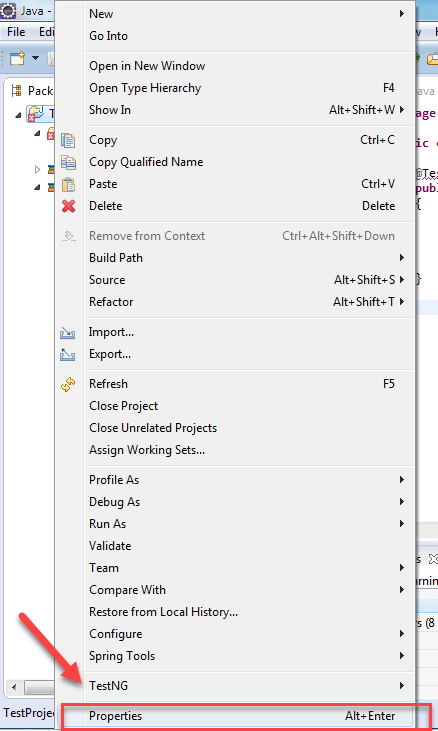
This line will going to identify the "Google" search box and enter the data you sent using sendKeys method.

**Output: The above program contains the errors, so it is not possible to execute.**

Similarly for Class DemoB,

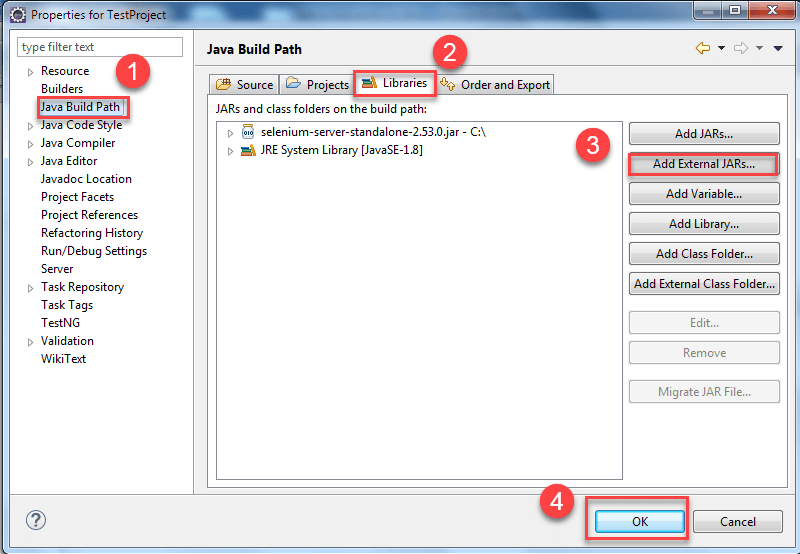
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor22.png)

The red color underlined words are the errors here. Because you have not yet added the jar file that contains these classes and interfaces. In order to remove the errors present in the above code add the corresponding jar file. Right click on the Java Project and Select "Properties".

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor23.png)

**Step 8)** In the property window,

1. Click on Java Build path on the left navigation pane.
2. Click on the libraries tab,
3. Click on the Add External JARs and Select the Selenium-standalone jar file
4. Click on "OK" button.

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor24.png)

After this step, all the errors will be removed automatically. If not then place the mouse over the code which is showing errors and import all necessary classes and interfaces.

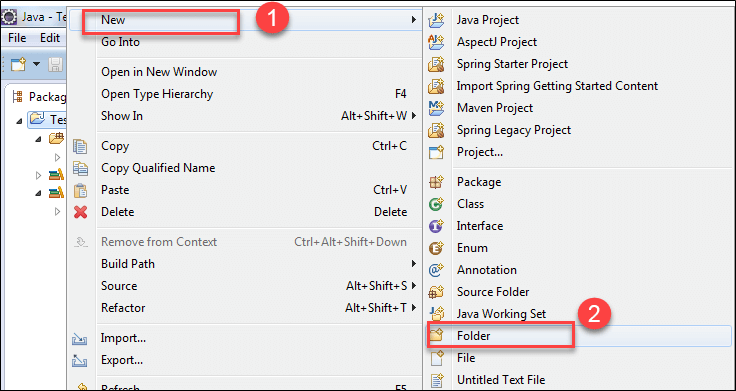
For @test if it is still showing errors then place the mouse over there. It will show the possible options. Then click on add TestNG Library option. Similarly do it for other 'class' also.

After writing your code inside both the classes i.e. DemoA, DemoB go to the next step.

**Step 9)** In this step,

1. Right-click on the Project and
2. Select option folder to create a folder called 'lib' and paste testNG jar file and selenium jar file (selenium-server-standalone<version>).

(Testng and Selenium-server-standalone jar file need to be downloaded from web and store it inside your system. You need to manually go to that folder and copy these two jars and right click on 'lib' folder present in eclipse and click on paste)

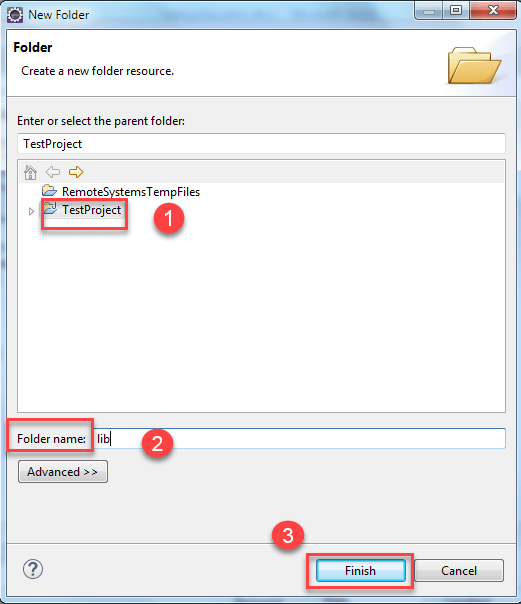
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor25.png)

**Step 10)** In this step,

1. Select the parent folder

2. Give the Folder name as 'lib' and (The primary purpose of adding these two jar files in the lib folder is that, while executing from the command prompt you can tell the compiler that the required jar files for the execution of the program are present in this location. If you want to execute testng.xml from eclipse then this lib folder is not at all required)

3. Click on 'finish' button

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor26.png)

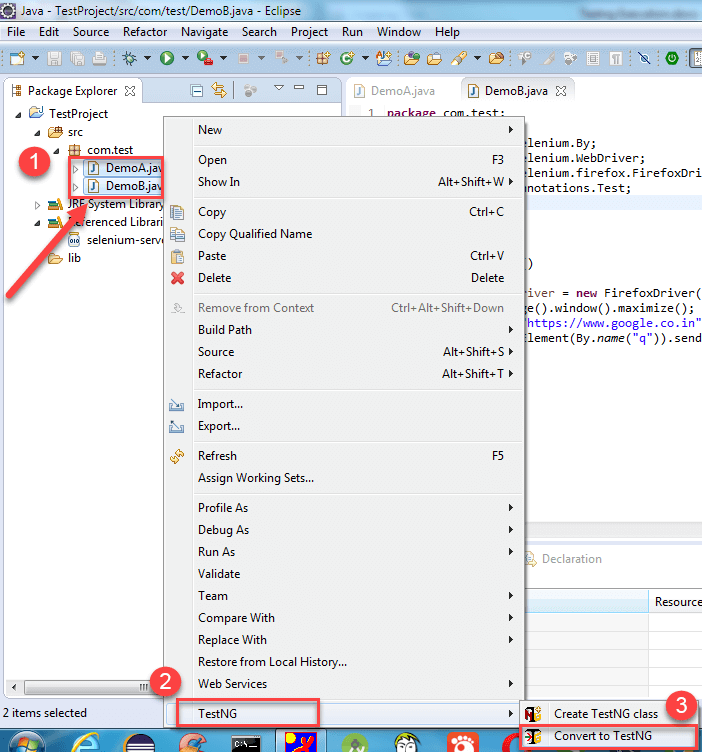
Testng and Selenium-server-standalone jar file need to be downloaded from web and store it inside your system. You need to manually go to that folder and copy these two jars and right click on 'lib' folder present in eclipse and click on paste

After creating a folder, next step will be converting our programs that are DemoA and DemoB into Testng.xml file.

**Convert and Execute Selenium Project to TestNG**

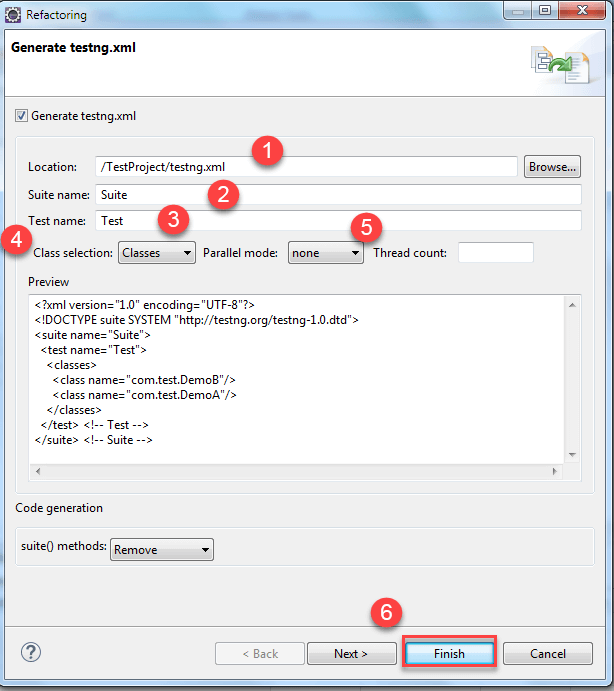
**Step 1)** In this step,

1. Select two java files present under the package and right click.
2. Select the option called "testNG."
3. Click on convert to "testNG".

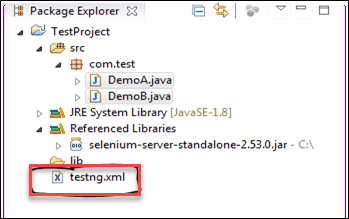
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor27.png)

**Step 2)** A new window will open. In this window, enter details like

1. Location
2. Suite Name
3. Test Name
4. Class Selection
5. Parallel Mode
6. Click on "Finish" button

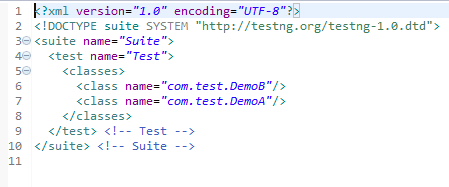
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor29.png)

Then the testng.xml file created under the java project, and it will look like this.

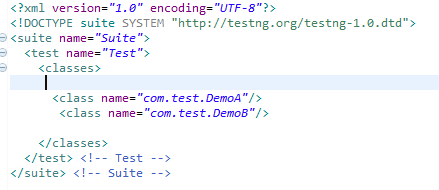
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor30.png)

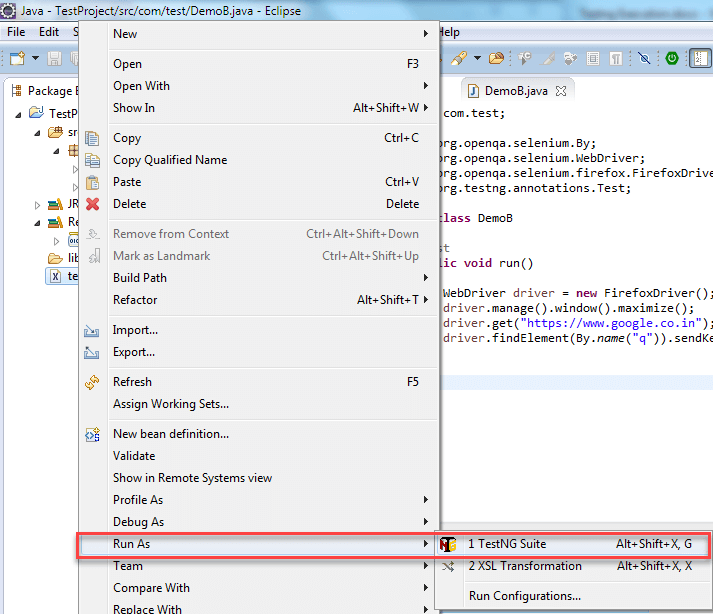
(if you want to run that testng.xml file then Right click on that testng.xml file present on the left navigation pane in Package Explorer and click on run as TestNG Suite. )

**Step 3)** The testng.xml suite file will look like this:

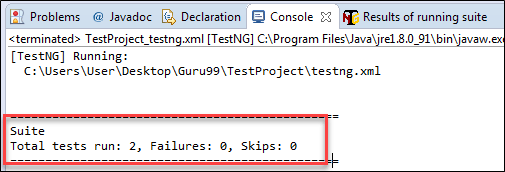
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor31.png)

If you want to execute DemoA class first, then remove that complete line and add it before DemoB class like this:

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor32.png)

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor33.png)

**Step 4)** After executing the testng.xml file, it will display the result in the following manner. (Since you have not written any executable statements using System.out.println() so it is not printing anything on the console window).

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor34.png)

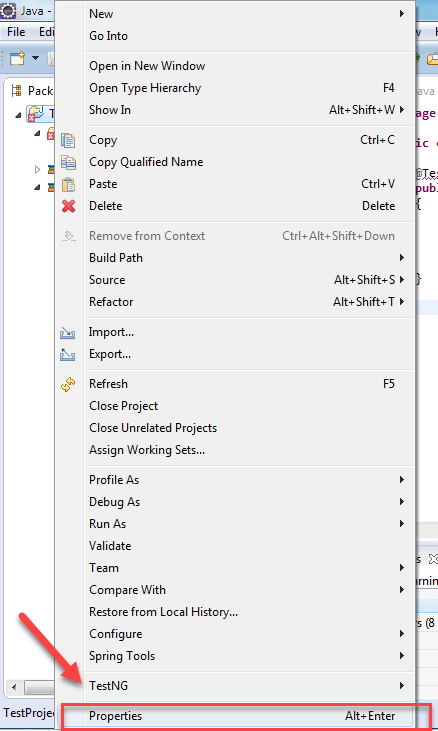
This is one of the ways you will execute test through the eclipse, if you want to execute the same testng.xml suite file that contains two class file such as DemoA, DemoB from the command prompt you need to follow the below steps.

Now open the command prompt and go to the Project workspace.

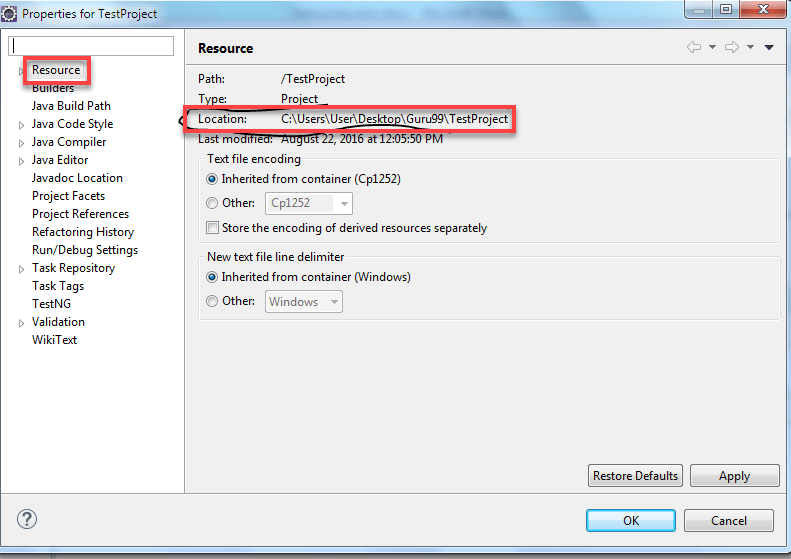
**Execute TestNG via Command Line**

You need to find the location of the Project Space

**Step 1)** First right click on the Java Project and Select the properties option. In properties window select option resource.

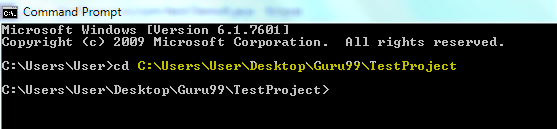
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor35.png)

**Step 2)** Now when you click on "resource" link, present in the left navigation pane. It will show where exactly the project is stored

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor36.png)

The project workspace is **C:\Users\User\Desktop\Guru99\TestProject**. So you are changing directory to that particular location.

**Step 3)** Then type the following command. (This command is generic command)

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor37.png)

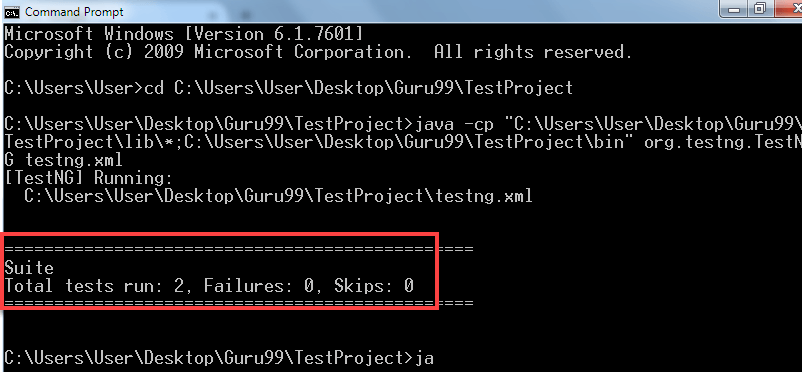
java –cp "path of lib folder present in workspace\\*; path of bin folder present in project workspace; path of testng.jar file present in lib folder of project workspace" org.testng.TestNG testng.xml

but for our project you need to write the following command.

Java –cp "C:\Users\User\Desktop\Guru99\TestProject\lib\\*;

C:\Users\User\Desktop\Guru99\TestProject\bin" org.testng.TestNG testng.xml

**Step 4)** Then press the Enter. Both the classes DemoA and DemoB will start their execution. Finally, it will display the result on the command prompt.

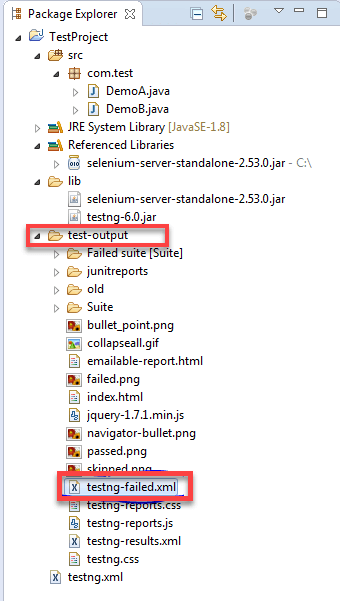
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor38.png)

**How to run only failed test cases**

If you want to execute only failed test cases through the Eclipse, then first refresh the project.

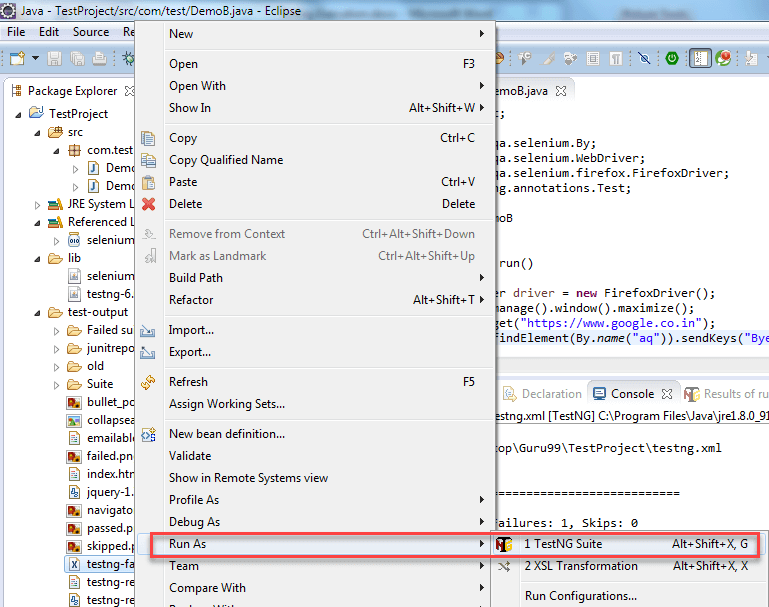
**Step 1)** Right click on the java project (Demo A and B). Select the refresh option or simply select the java project and press F5.

**Step 2)** Then you will able to see the test-output folder. In that folder, you have a file with name testng-failed.xml.

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor39.png)

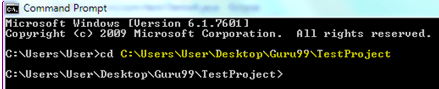
**Step 3)** Right-click on this file and click on run as and select the option called "testNG suite".

Suppose if you have three test cases if all the test cases are executed successfully means you are not able to see this folder under the test-output folder. This folder will appear only when one of the test case is failed. Then run this file, it will going to run only failed test cases.

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor40.png)

**Running testng-failed.xml file from Command line**

**Step 1)** To run failed test cases in command prompt. Open the command prompt and go to the Project workspace.

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor41.png)

My project workspace is **C:\Users\User\Desktop\Guru99\TestProject**. So you will be changing directory to that particular location.

**Step 2)** Type the following command. (This command is generic command)

java –cp "path of lib folder present in workspace\\*; path of bin folder present in project workspace; path of testing.jar file present in lib folder of project workspace" org.testng.TestNG test-output/testng-failed.xml

For our project, you need to write the following command.

Java –cp "C:\Users\User\Desktop\Guru99\TestProject\lib\\*;

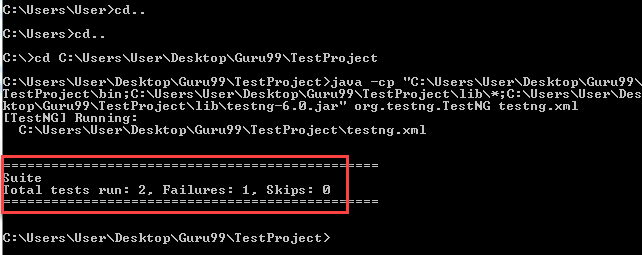
C:\Users\User\Desktop\Guru99\TestProject\bin" org.testng.TestNG test-output/testng-failed.xml

**Step 3)** Then press the Enter. It will going to run only failed classes and display corresponds to that class only.

In the above class i.e. DemoB just change the code like this.

driver.findElement(By.name("a")).sendKeys("Bye");

The above code will not going to find any element. So, it will throw the exception. But in our program, you are not handling the exception so our program will stop executing the remaining code. Because of this reason, the ClassB will going to fail.

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor42.png)

**Summary**:

* TestNG is Automation Testing Framework which is inspired from the Junit and contains different annotations.
* TestNG generates the reports in its standard report,which contains the following things:
  + How many test cases are there ?
  + How many test cases are passed ?
  + How many test cases are failed ?
  + How many test cases are skipped ?
* Multiple test cases can be grouped easily and executed them by converting test classes into testing suite file.
* TestNG can be easily integrated with other third-party tools. It provides different features like assigning the priority to the test cases, execution of same test case multiple times using invocation count.
* If any of the test case fails while executing multiple test cases, you can execute that specific test case separately.
* The static method "log" of Reporter class can be used to store logging information which is present in org.testng

This article is contributed by **Sandeep Batageri**

[](https://www.guru99.com/top-100-selenium-interview-questions-answers.html)

**[Top 100 Selenium Interview Questions & Answers](https://www.guru99.com/top-100-selenium-interview-questions-answers.html)**

**TestNG Report Generation in Selenium WebDriver**

Report generation is very important when you are doing the Automation Testing as well as for Manual Testing.

* By looking at the result, you can easily identify how many test cases are passed, failed and skipped.
* By looking at the report, you will come to know what the status of the project is.

Selenium web driver is used for automating the web-application, but it won't generate any reports.

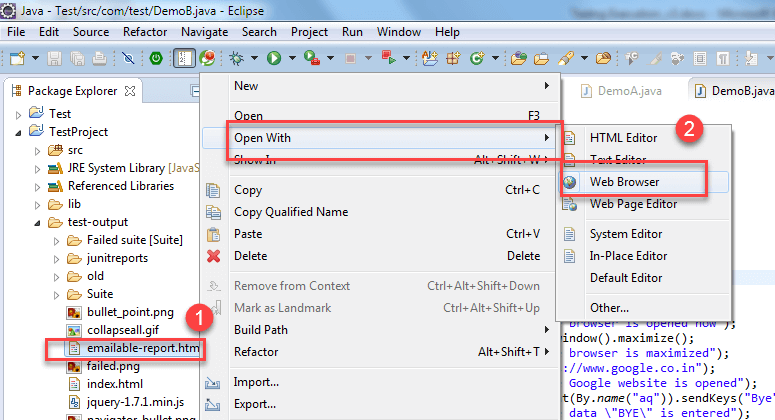
* The TestNG will generate the default report.
* When you execute testng.xml file, and refresh the project. You will get test-output folder in that folder.
* Right click on the emailable-report.html and select the option. Open with the web browser.

In this tutorial, you will learn-

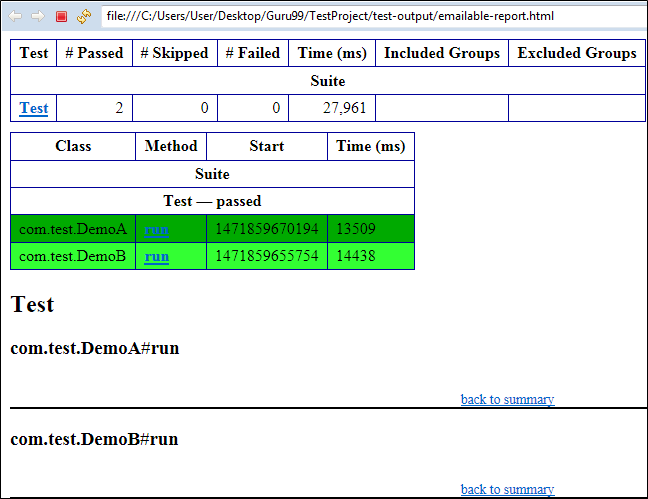
* [Method-1: emailable-report.html](https://www.guru99.com/testng-report.html#7)
* [Method-2: index.html](https://www.guru99.com/testng-report.html#8)
* [Method-3: Reporter Class](https://www.guru99.com/testng-report.html#9)

**Method-1: emailable-report.html**

1. Click on option "emailable-report.html"
2. Click on option web browser

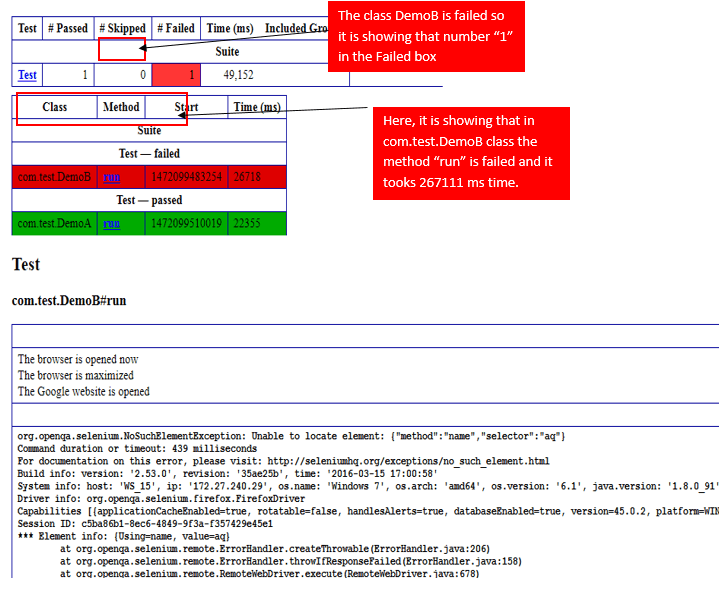
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor43.png)

The output report of testng will look like below if both the classes are passed:

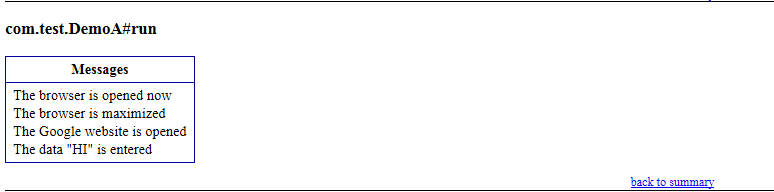
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor44.png)

Consider the scenario in where you are intentionally failing the test case i.e. DemoB class. Then convert both the classes into testng.xml suite file and run it. Then the result will look like this. It will show the failed test cases.

This is result for DemoB class:

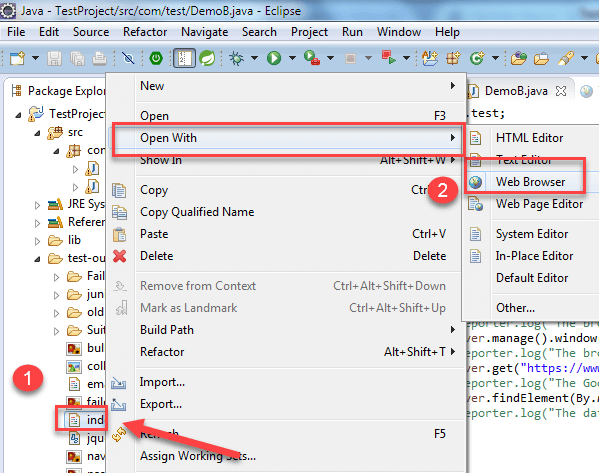
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor51.png)

Similarly, result for the Class DemoA will look like this:

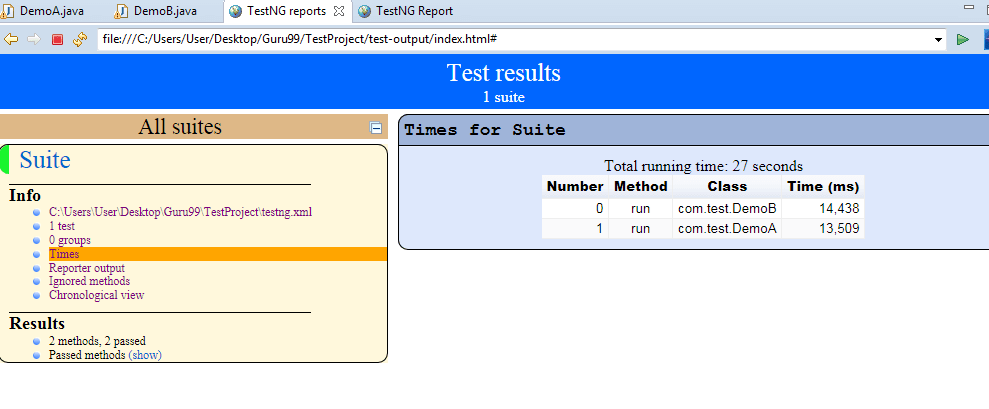
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor52.png)

**Method-2 index.html**

1. Right click on the index.html from the project directory.
2. Select option open with web browser option. It will display the result in the following order.

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor53.png)

The result will look like this:

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor54.png)

**Method-3 Reporter Class**

Along with these report generated methods, you can use object.properties file to store the system generated logs as well as user generated logs. But one of the simplest ways to store log information in testing is using **Reporter Class**.

Reporter is a class present in TestNG. It provides 4 different methods to store log information they are:

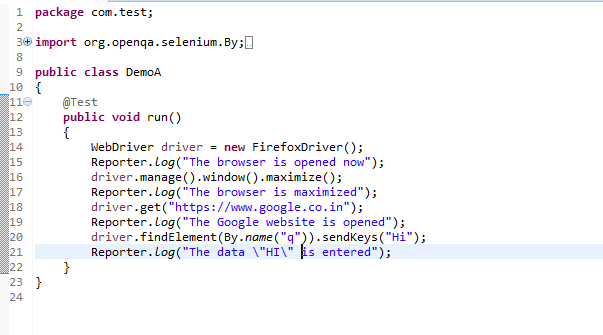
1. Reporter.log(String s);
2. Reporter.log(String s, Boolean logToStandardOut);
3. Reporter.log(String s, int level);
4. Reporter.log(String s, int level, Boolean logToStandardOut);

**Example**:

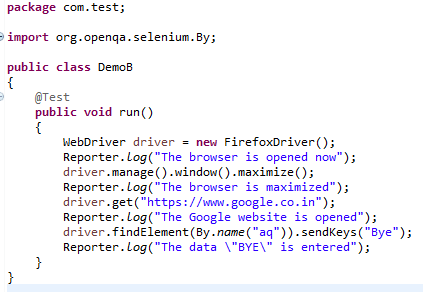
Create Two classes such as DemoA and DemoB and write the following code inside the classes.

For Class DemoA;

* The Code for DemoA is already explained above. Here you are using **log method of Reporter class.** (For implementing a reporting class, the class has to implement an org.testng.IReporter interface).
* The log method is a static method of Reporter class. So you are accessing that method through the Reporter class.
* The log method is used to store log information that is written inside the program. By looking at the log information, you will easily come to know where exactly the execution of the program is stopped.

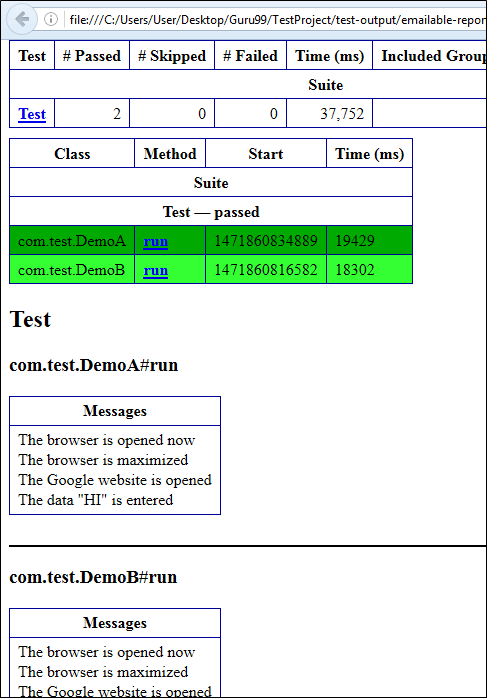
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor55.png)

For Class DemoB:

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor56.png)

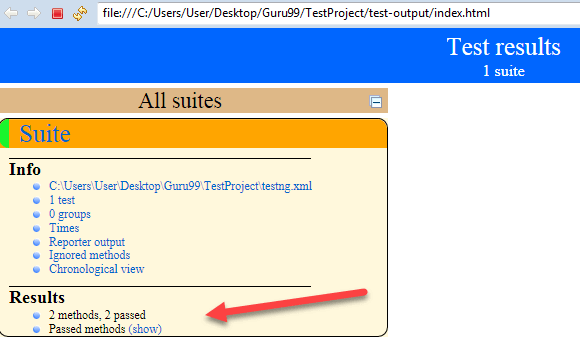
* Now, Create testng.xml file by selecting these two classes and
* Select run as and
* Click on the convert to testng.
* Then run this testng.xml file by selecting run as and select testng suite.
* Then refresh the project open the test-output folder.

1. In the test-output folder, open the emailable-report.html. It will look like:

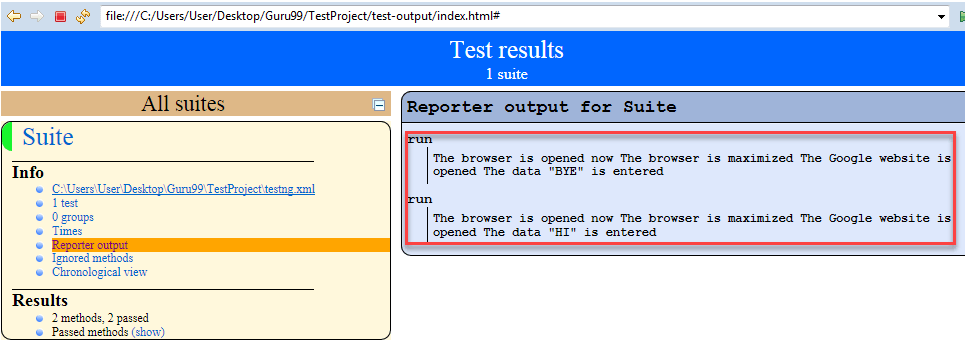
[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor57.png)

Similarly, you will have an Output for Demo B project as well.

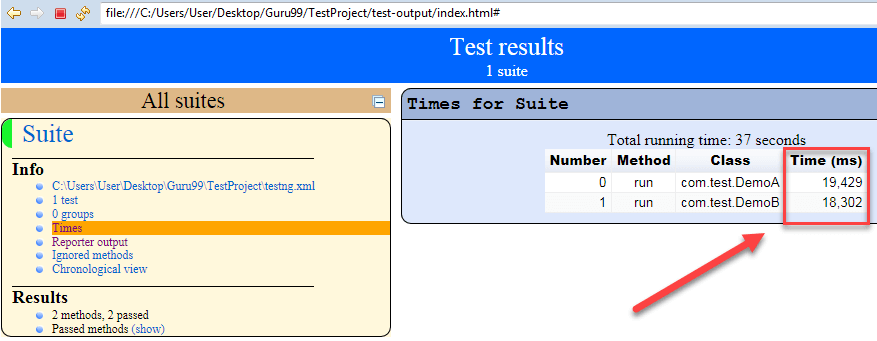
2. In test-output folder open the index.html. It will look like:

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor58.png)

Click on reporter output. It will open logging info whatever written in the test methods.

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor59.png)

Click on the Times. It will going to show how much time it took to run the test method present in class.

[](https://www.guru99.com/images/8-2016/090116_1045_TestNGTutor60.png)

* [Prev](https://www.guru99.com/run-failed-test-cases-in-testng.html)

[**Report a Bug**](javascript:void(%20window.open(%20'https://form.jotform.me/72391811797466',%20'blank',%20'scrollbars=yes,%20toolbar=no,%20width=700,%20height=700'%20)%20))

* [Next](https://www.guru99.com/pdf-emails-and-screenshot-of-test-reports-in-selenium.html)

[https://www.guru99.com/images/logo/logo.png](https://www.guru99.com/)

≡

# Customize, PDF & Email TestNG Reports in Selenium WebDriver

Before we look into anything else, let's first understand -

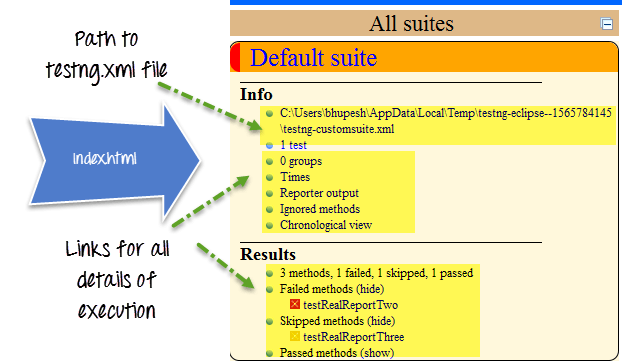
## Why do we need reporting?

When we are using Selenium or any other automation tool, we are performing operations on the web application. But our purpose of automation is not just to exercise the Application Under Test. We, as an automation tester are supposed to test the application, find bugs and report it to the development team or higher management. Here the reporting gets importance for software [Testing](https://www.guru99.com/software-testing.html) process

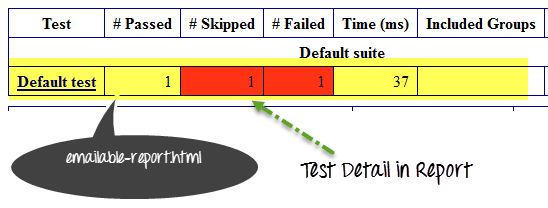
## TestNG Reporting

TestNG library provides a very handy reporting feature. After execution, [Testng](https://www.guru99.com/all-about-testng-and-selenium.html) will generate a test-output folder at the root of the project. This folder contains two type of Reports-

**Index.html:** This is the complete report of current execution which contains information like an error, groups, time, reporter logs, testng XML files.

[](https://www.guru99.com/images/AdvanceSelenium/071514_0728_PDFEmailsan1.png)

**emailable-report.html:** This is the summarize report of the current test execution which contains [Test Case](https://www.guru99.com/test-case.html) message in green (for pass test cases) and red(for failed test cases) highlight.

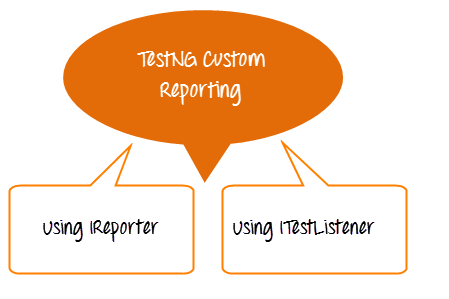
[](https://www.guru99.com/images/AdvanceSelenium/071514_0728_PDFEmailsan2.png)

## How to customize TestNG Report

TestNG reporting is quite handy but still, sometimes we need some less data in reports or want to display reports in some other format like pdf, excel, etc. or want to change report's layout.

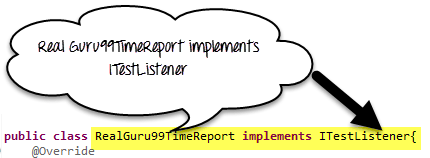
There can be two ways we can customize TestNG report

* Using ITestListener Interface:
* Using IReporter Interface:

[](https://www.guru99.com/images/AdvanceSelenium/071514_0728_PDFEmailsan3.png)

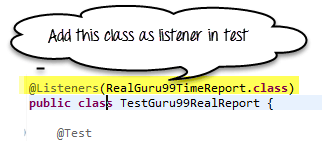
### ITestListener Interface

We use this interface when we need to customize real time report. In other words, if we are executing the bunch of test cases in a TetNG suite and we want to get the report of each test case, then after each test case we need to implement ITestListener interface. This interface will override onTestFailure, onTestStart, onTestSkipped method to send the correct status of the current test case.

[](https://www.guru99.com/images/AdvanceSelenium/071514_0728_PDFEmailsan4.png)

Here are the steps we will follow

* Create a class say RealGuru99Report and implement iTestListener in it.
* Implement methods of iTestListener
* Create test method and add RealGuru99Report class as a listener in Test Method class.

[](https://www.guru99.com/images/AdvanceSelenium/071514_0728_PDFEmailsan5.png)

**Code Example**

RealGuru99TimeReport.java is the real time reporting class. It will implement ITestListener interface for reporting

package testNGReport.realTimeReport;

import org.testng.ITestContext;

import org.testng.ITestListener;

import org.testng.ITestResult;

public class RealGuru99TimeReport implements ITestListener{

@Override

public void onStart(ITestContext arg0) {

System.out.println("Start Of Execution(TEST)->"+arg0.getName());

}

@Override

public void onTestStart(ITestResult arg0) {

System.out.println("Test Started->"+arg0.getName());

}

@Override

public void onTestSuccess(ITestResult arg0) {

System.out.println("Test Pass->"+arg0.getName());

}

@Override

public void onTestFailure(ITestResult arg0) {

System.out.println("Test Failed->"+arg0.getName());

}

@Override

public void onTestSkipped(ITestResult arg0) {

System.out.println("Test Skipped->"+arg0.getName());

}

@Override

public void onFinish(ITestContext arg0) {

System.out.println("END Of Execution(TEST)->"+arg0.getName());

}

@Override

public void onTestFailedButWithinSuccessPercentage(ITestResult arg0) {

// TODO Auto-generated method stub

}

}

**TestGuru99RealReport.java is the test case for real report**

package testNGReport.realTimeReport;

import org.testng.Assert;

import org.testng.annotations.Listeners;

import org.testng.annotations.Test;

@Listeners(RealGuru99TimeReport.class)

public class TestGuru99RealReport {

@Test

public void testRealReportOne(){

Assert.assertTrue(true);

}

@Test

public void testRealReportTwo(){

Assert.assertTrue(false);

}

//Test case depends on failed testcase= testRealReportTwo

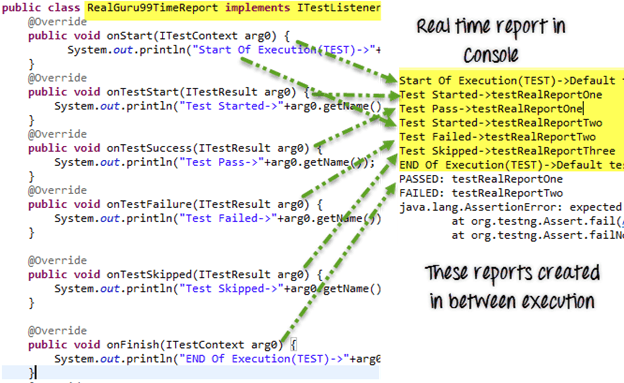
@Test(dependsOnMethods="testRealReportTwo")

public void testRealReportThree(){

}

}

The output will look like-

[](https://www.guru99.com/images/AdvanceSelenium/071514_0728_PDFEmailsan6.png)

### IReporter Interface

If we want to customize final test report generated by TestNG, we need to implement IReporter interface. This interface has only one method to implement generateReport. This method has all the information of a complete test execution in the List<ISuite>, and we can generate the report using it.

**Code Example**

**Guru99Reporter.java is the file used to customize report**

package testNGReport.iReporterReport;

import java.util.Collection;

import java.util.Date;

import java.util.List;

import java.util.Map;

import java.util.Set;

import org.testng.IReporter;

import org.testng.IResultMap;

import org.testng.ISuite;

import org.testng.ISuiteResult;

import org.testng.ITestContext;

import org.testng.ITestNGMethod;

import org.testng.xml.XmlSuite;

public class Guru99Reporter implements IReporter{

@Override

public void generateReport(List<XmlSuite> arg0, List<ISuite> arg1,

String outputDirectory) {

// Second parameter of this method ISuite will contain all the suite executed.

for (ISuite iSuite : arg1) {

//Get a map of result of a single suite at a time

Map<String,ISuiteResult> results = iSuite.getResults();

//Get the key of the result map

Set<String> keys = results.keySet();

//Go to each map value one by one

for (String key : keys) {

//The Context object of current result

ITestContext context = results.get(key).getTestContext();

//Print Suite detail in Console

System.out.println("Suite Name->"+context.getName()

+ "::Report output Ditectory->"+context.getOutputDirectory()

+"::Suite Name->"+ context.getSuite().getName()

+"::Start Date Time for execution->"+context.getStartDate()

+"::End Date Time for execution->"+context.getEndDate());

//Get Map for only failed test cases

IResultMap resultMap = context.getFailedTests();

//Get method detail of failed test cases

Collection<ITestNGMethod> failedMethods = resultMap.getAllMethods();

//Loop one by one in all failed methods

System.out.println("--------FAILED TEST CASE---------");

for (ITestNGMethod iTestNGMethod : failedMethods) {

//Print failed test cases detail

System.out.println("TESTCASE NAME->"+iTestNGMethod.getMethodName()

+"\nDescription->"+iTestNGMethod.getDescription()

+"\nPriority->"+iTestNGMethod.getPriority()

+"\n:Date->"+new Date(iTestNGMethod.getDate()));

}

}

}

}

}

**TestGuru99ForReporter.java is a demo for Custom reporting**

package testNGReport.iReporterReport;

import org.testng.Assert;

import org.testng.annotations.Listeners;

import org.testng.annotations.Test;

//Add listener to listen report and write it when testcas finished

@Listeners(value=Guru99Reporter.class)

public class TestGuru99ForReporter {

@Test(priority=0,description="testReporterOne")

public void testReporterOne(){

//Pass test case

Assert.assertTrue(true);

}

@Test(priority=1,description="testReporterTwo")

public void testReporterTwo(){

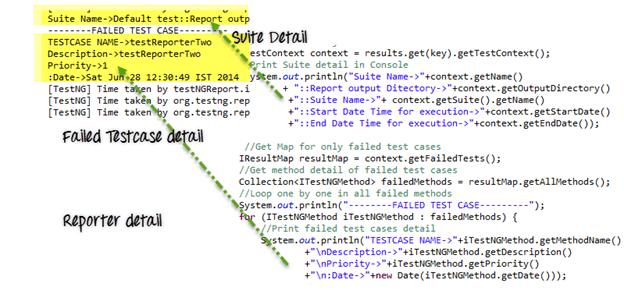
//Fail test case

Assert.assertTrue(false);

}

}

Output will be like-

[](https://www.guru99.com/images/AdvanceSelenium/071514_0728_PDFEmailsan7.png)

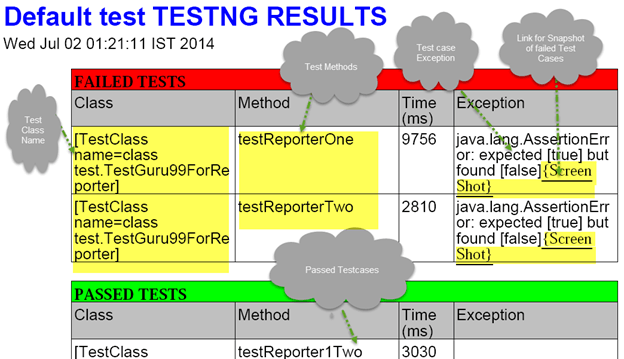
## PDF and Email of Reports

The above report implementation is quite simple and clear to get you started with report customization.

But in corporate environment, you will need to create highly customized reports. Here is the scenario we will be dealing with

1. Create Custom Report in PDF form
2. Take Screenshots ONLY on Errors. Link to screenshots in PDF
3. Send Email of the PDF

The PDF report looks like this

[](https://www.guru99.com/images/AdvanceSelenium/071514_0728_PDFEmailsan9.png)

To create pdf report we need a [Java](https://www.guru99.com/java-tutorial.html) API **IText**. Download it [here](http://mirrors.ibiblio.org/pub/mirrors/maven2/com/lowagie/itext/2.1.7/itext-2.1.7.jar) . There is another custom listener class which is actually implementing this IText jar and creating a pdf report for us. Download it [here](https://groups.google.com/group/testng-users/attach/3bc22fcb3e8cc39c/JyperionListener.java?part=0.1&authuser=0)

Above figure shows the default format of the PDF report generated. You can customize it

Here is how we will approach this

Step 1) Create a Base Class

Step 2) Customize JypersionListerner.Java (PDF creation code)

Step 3) Create a TestGuru99PDFEmail.java which will execute test cases , create PDF

Step 4) Append code to TestGuru99PDFEmail.java to send PDF report via email

**Let's look into these steps**

Step 1) Create Base Class

This base class has functions to create WebDriver and Take Screenshot

package PDFEmail;

import java.io.File;

import org.apache.commons.io.FileUtils;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

public class BaseClass {

static WebDriver driver;

public static WebDriver getDriver(){

if(driver==null){

WebDriver driver ;

System.setProperty("webdriver.firefox.marionette","C:\\geckodriver.exe");

driver = new FirefoxDriver();

}

return driver;

}

/\*\*

\* This function will take screenshot

\* @param webdriver

\* @param fileWithPath

\* @throws Exception

\*/

public static void takeSnapShot(WebDriver webdriver,String fileWithPath) throws Exception{

//Convert web driver object to TakeScreenshot

TakesScreenshot scrShot =((TakesScreenshot)webdriver);

//Call getScreenshotAs method to create image file

File SrcFile=scrShot.getScreenshotAs(OutputType.FILE);

//Move image file to new destination

File DestFile=new File(fileWithPath);

//Copy file at destination

FileUtils.copyFile(SrcFile, DestFile);

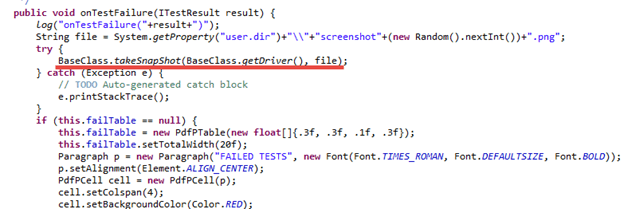
}

}

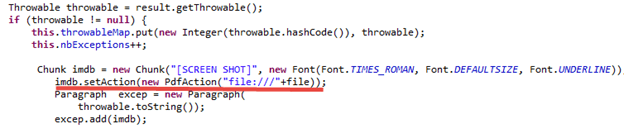
Step 2) Customize JypersionListener.java

We will stick with the default report format. But we will make 2 customizations

* Adding code to instruct JypersionListener to take screenshot on Error
* Attaching the link of the screenshot take in the PDF report

[](https://www.guru99.com/images/AdvanceSelenium/071514_0728_PDFEmailsan10.png)

Add code to attach the screenshot to the PDF report

[](https://www.guru99.com/images/AdvanceSelenium/071514_0728_PDFEmailsan11.png)

Step 3) Create a TestGuru99PDFEmail.java which will execute test cases , create PDF

* Here we will add JyperionListener.class as listener
* We will Execute 3 test cases.
* Using Assert.assertTrue we will fail 2 test cases while passing just one.
* Screenshot will be taken for the failed test cases only as per our customizations

package PDFEmail;

import java.util.Properties;

import javax.activation.DataHandler;

import javax.activation.DataSource;

import javax.activation.FileDataSource;

import javax.mail.BodyPart;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.Multipart;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.AddressException;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeBodyPart;

import javax.mail.internet.MimeMessage;

import javax.mail.internet.MimeMultipart;

import org.openqa.selenium.WebDriver;

import org.testng.Assert;

import org.testng.annotations.AfterSuite;

import org.testng.annotations.Listeners;

import org.testng.annotations.Test;

import reporter.JyperionListener;

//Add listener for pdf report generation

@Listeners(JyperionListener.class)

public class TestGuru99PDFReport extends BaseClass {

WebDriver driver;

//Testcase failed so screen shot generate

@Test

public void testPDFReportOne(){

driver = BaseClass.getDriver();

driver.get("http://google.com");

Assert.assertTrue(false);

}

//Testcase failed so screen shot generate

@Test

public void testPDFReporTwo(){

driver = BaseClass.getDriver();

driver.get("http:/guru99.com");

Assert.assertTrue(false);

}

//Test test case will be pass, so no screen shot on it

@Test

public void testPDFReportThree(){

driver = BaseClass.getDriver();

driver.get("http://demo.guru99.com");

Assert.assertTrue(true);

}

Step 4) Append code to TestGuru99PDFEmail.java to send PDF report via email

* We will use the annotation @AfterSuite to send email of the PDF report
* We will be sending email using Gmail
* To enable Email, need to import many library files like mail.jar, pop3.jar, smptp.jar, etc
* Before you execute this, do enter the from, to email address and password

//After complete execution send pdf report by email

@AfterSuite

public void tearDown(){

sendPDFReportByGMail("[FROM@gmail.com](mailto:FROM@gmail.com)", "password", "[TO@gmail.com](mailto:TO@gmail.com)", "PDF Report", "");

}

/\*\*

\* Send email using java

\* @param from

\* @param pass

\* @param to

\* @param subject

\* @param body

\*/

private static void sendPDFReportByGMail(String from, String pass, String to, String subject, String body) {

Properties props = System.getProperties();

String host = "smtp.gmail.com";

props.put("mail.smtp.starttls.enable", "true");

props.put("mail.smtp.host", host);

props.put("mail.smtp.user", from);

props.put("mail.smtp.password", pass);

props.put("mail.smtp.port", "587");

props.put("mail.smtp.auth", "true");

Session session = Session.getDefaultInstance(props);

MimeMessage message = new MimeMessage(session);

try {

//Set from address

message.setFrom(new InternetAddress(from));

message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));

//Set subject

message.setSubject(subject);

message.setText(body);

BodyPart objMessageBodyPart = new MimeBodyPart();

objMessageBodyPart.setText("Please Find The Attached Report File!");

Multipart multipart = new MimeMultipart();

multipart.addBodyPart(objMessageBodyPart);

objMessageBodyPart = new MimeBodyPart();

//Set path to the pdf report file

String filename = System.getProperty("user.dir")+"\\Default test.pdf";

//Create data source to attach the file in mail

DataSource source = new FileDataSource(filename);

objMessageBodyPart.setDataHandler(new DataHandler(source));

objMessageBodyPart.setFileName(filename);

multipart.addBodyPart(objMessageBodyPart);

message.setContent(multipart);

Transport transport = session.getTransport("smtp");

transport.connect(host, from, pass);

transport.sendMessage(message, message.getAllRecipients());

transport.close();

}

catch (AddressException ae) {

ae.printStackTrace();

}

catch (MessagingException me) {

me.printStackTrace();

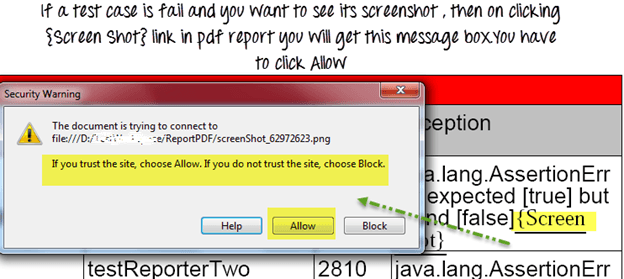
}

}

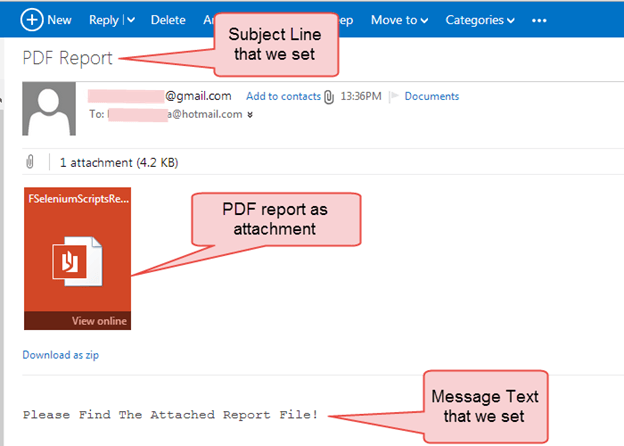
}

**Download the complete project here**

Note: When we click on the screen shot link in pdf, it shows security dialog. We have to allow this dialog to open pdf.

[](https://www.guru99.com/images/AdvanceSelenium/071514_0728_PDFEmailsan12.png)

The email so generated will look like this

[](https://www.guru99.com/images/AdvanceSelenium/071514_0728_PDFEmailsan13.png)

Summary:

* TestNG has an inbuilt reporting ability in it.
* After a complete execution of test cases, TestNG generates a test-output folder in the root of the project.
* In the test-output folder, there are two main reports, index.html, and emailable-report.html.
* To customize TestNG report we need to implement two interfaces, ITestListener and IReporter.
* If we need to get a report in between execution, we need ITestListener.
* For creating a final report after complete execution, we need to implement IReporter.
* Taking the screenshot, in Selenium WebDriver, we need to type cast WebDriver to TakesScreenShot interface.
* To generate pdf reports we need to add IText jar in the project.

[Download the Selenium Project Files for the Demo in this Tutorial](https://drive.google.com/uc?export=download&id=0B_vqvT0ovzHcUFVqWUJWRGpibmc)

* [Prev](https://www.guru99.com/testng-report.html)

[**Report a Bug**](javascript:void(%20window.open(%20'https://form.jotform.me/72391811797466',%20'blank',%20'scrollbars=yes,%20toolbar=no,%20width=700,%20height=700'%20)%20))

* [Next](https://www.guru99.com/page-object-model-pom-page-factory-in-selenium-ultimate-guide.html)

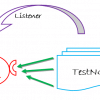
YOU MIGHT LIKE:

###### Selenium

[](https://www.guru99.com/selenium-csharp-tutorial.html)

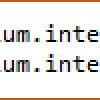
#### [Selenium C# Webdriver Tutorial: NUnit Example](https://www.guru99.com/selenium-csharp-tutorial.html)

###### Selenium

[](https://www.guru99.com/listeners-selenium-webdriver.html)

#### [TestNG Listeners in Selenium: ITestListener & ITestResult Example](https://www.guru99.com/listeners-selenium-webdriver.html)

###### Selenium

[](https://www.guru99.com/verify-tooltip-selenium-webdriver.html)

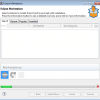
#### [How to Verify Tooltip using Selenium WebDriver](https://www.guru99.com/verify-tooltip-selenium-webdriver.html)

###### Selenium

[](https://www.guru99.com/scroll-up-down-selenium-webdriver.html)

#### [How to Scroll Down or UP a Page in Selenium Webdriver](https://www.guru99.com/scroll-up-down-selenium-webdriver.html)

###### Selenium

[](https://www.guru99.com/selenium-python.html)

#### [Selenium Webdriver using Python: Tutorial with Example](https://www.guru99.com/selenium-python.html)

###### Selenium

[](https://www.guru99.com/selenium-github.html)

#### [Github Integration with Selenium: Complete Tutorial](https://www.guru99.com/selenium-github.html)

### Selenium Tutorials

* [24) Introduction to TestNG Groups](https://www.guru99.com/introduction-testng-groups.html)
* [25) Test Case Priority in TestNG](https://www.guru99.com/test-case-priority-testng.html)
* [26) Sessions,Parallel & Dependency](https://www.guru99.com/sessions-parallel-run-and-dependency-in-selenium.html)
* [27) Execute multiple test suites](https://www.guru99.com/testng-execute-multiple-test-suites.html)
* [28) TestNG Listeners](https://www.guru99.com/listeners-selenium-webdriver.html)
* [29) TestNG Tutorial](https://www.guru99.com/testng-tutorial.html)
* [30) TestNG Report Generation](https://www.guru99.com/testng-report.html)
* [31) PDF, Emails and Screenshot](https://www.guru99.com/pdf-emails-and-screenshot-of-test-reports-in-selenium.html)
* [32) POM & Page Factory](https://www.guru99.com/page-object-model-pom-page-factory-in-selenium-ultimate-guide.html)
* [33) Parameterization](https://www.guru99.com/parameterization-using-xml-and-dataproviders-selenium.html)
* [34) Excel in Selenium](https://www.guru99.com/all-about-excel-in-selenium-poi-jxl.html)
* [35) Handling Date Time Picker](https://www.guru99.com/handling-date-time-picker-using-selenium.html)
* [36) Selenium Grid](https://www.guru99.com/introduction-to-selenium-grid.html)
* [37) Maven & Jenkins](https://www.guru99.com/maven-jenkins-with-selenium-complete-tutorial.html)
* [38) Create Framework Key & Hybrid](https://www.guru99.com/creating-keyword-hybrid-frameworks-with-selenium.html)
* [39) Database Testing](https://www.guru99.com/database-testing-using-selenium-step-by-step-guide.html)
* [40) Handling Iframes](https://www.guru99.com/handling-iframes-selenium.html)
* [41) Cross Browser Testing](https://www.guru99.com/cross-browser-testing-using-selenium.html)
* [42) Take Screenshot in Selenium](https://www.guru99.com/take-screenshot-selenium-webdriver.html)
* [43) Log4j and LogExpert](https://www.guru99.com/tutorial-on-log4j-and-logexpert-with-selenium.html)
* [44) HTMLUnit Driver & PhantomJS](https://www.guru99.com/selenium-with-htmlunit-driver-phantomjs.html)
* [45) Robot API](https://www.guru99.com/using-robot-api-selenium.html)
* [46) AutoIT with Selenium](https://www.guru99.com/use-autoit-selenium.html)
* [47) SSL Certificate Error Handling](https://www.guru99.com/ssl-certificate-error-handling-selenium.html)
* [48) Handling Ajax call](https://www.guru99.com/handling-ajax-call-selenium-webdriver.html)
* [49) Execute JavaScript based code](https://www.guru99.com/execute-javascript-selenium-webdriver.html)
* [50) Using Selenium with Python](https://www.guru99.com/selenium-python.html)

### About

[About US](https://www.guru99.com/about-us.html)  
[Advertise with Us](https://www.guru99.com/advertise-us.html)  
[Write For Us](https://www.guru99.com/become-an-instructor.html)  
[Contact US](https://www.guru99.com/contact-us.html)

### Career Suggestion

[SAP Career Suggestion Tool](https://www.guru99.com/best-sap-module.html)  
[Software Testing as a Career](https://www.guru99.com/software-testing-career-complete-guide.html)  
[Certificates](https://www.guru99.com/certificate-it-professional.html)

### Interesting

[Books to Read!](https://www.guru99.com/books.html)  
[Suggest a Tutorial](javascript:void(%20window.open(%20'https://form.jotform.me/71523024124442',%20'blank',%20'scrollbars=yes,%20toolbar=no,%20width=700,%20height=700'%20)%20))  
[Blog](https://www.guru99.com/blog/)  
[Quiz](https://www.guru99.com/tests.html)  
[eBook](https://www.guru99.com/ebook-pdf.html)

### Execute online

[Execute Java Online](https://www.guru99.com/try-java-editor.html)  
[Execute Javascript](https://www.guru99.com/execute-javascript-online.html)  
[Execute HTML](https://www.guru99.com/execute-html-online.html)  
[Execute Python](https://www.guru99.com/execute-python-online.html)

© Copyright - Guru99 2019 [Privacy Policy](https://www.guru99.com/privacy-policy.html)