**Infosys**

**TestNG Framework in Selenium**

**Jan, 2018**

**INFOSYS BPM**

**Bengaluru**

|  |  |  |  |
| --- | --- | --- | --- |
| Document No. | 1 | Ver. Rev. : | 1 |
| Authorized by: | Infosys | Signature/:  Date: | Drishti\_Srivastava/ Satheeskumar\_j |

**COPYRIGHT NOTICE**

©2018 Infosys Limited, Bangalore, India. All rights reserved. Infosys believes the information in this document is accurate as of its publication date; such information is subject to change without notice. Infosys acknowledges the proprietary rights of other companies to the trademarks, product names and such other intellectual property rights mentioned in this document. Except as expressly permitted, neither this document nor any part of it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, printing, photocopying, recording or otherwise, without the prior permission of Infosys Limited and/or any named intellectual property rights holders under this document.

Infosys Limited

Hosur Road

Electronic City, 3rd Cross

Bangalore 560 100

India.

Telephone: (91) (80)28520 261-270

Fax: (91) (80) 8520 362

Website: <http://www.infosys.com>

**Author(s): Drishti Srivastava, Satheeskumar J**

**Date written (MM/DD/YY): 01/29/18**

**Target readers: Automation** **Testers**

**Keywords:** **Selenium, TestNG, Data Source, DataProvider, Factory, XML, Reports**

Contents

[1 Introduction to TestNG 6](#_Toc505162506)

[1.1 What is TestNG? 6](#_Toc505162507)

[1.2 Features of TestNG 6](#_Toc505162508)

[1.3 Functionalities of TestNG 6](#_Toc505162509)

[1.4 Why TestNG? 6](#_Toc505162510)

[1.5 Benefits of TestNG 6](#_Toc505162511)

[2. Commonly used Terminologies used in TestNG 7](#_Toc505162512)

[2.1 TestNG “Groups” 7](#_Toc505162513)

[2.2 TestNG – Dependencies 8](#_Toc505162514)

[2.3 Priority 8](#_Toc505162515)

[3. Annotations in TestNG: - 9](#_Toc505162516)

[4. Data Source in TestNG: - 11](#_Toc505162517)

[4.1 @DataProvider Annotation: - 11](#_Toc505162518)

[4.2 @Factory Annotation: - 12](#_Toc505162519)

[5. Execution through Xml in TestNG: - 14](#_Toc505162520)

[5.1 How to create Xml file: - 14](#_Toc505162521)

[6 TestNG Reporting: - 18](#_Toc505162522)

[Why do we need reporting? - 18](#_Toc505162523)

[Generating HTML Reports 19](#_Toc505162524)

[7. Installation of TestNG: - 21](#_Toc505162525)

# 1 Introduction to TestNG

TestNG is a testing framework used for testing web based applications. Its features include use of annotations in the code, use of xml files to configure tests and Report generation. The various categories of testing like Unit testing, Integration testing, Functional testing can be done using this framework.

## 1.1 What is TestNG?

TestNG is an open source frame work which will help you in building test cases. It also helps us to achieve the following objectives in an automation project.

* To batch run the test cases or suites
* To define the test execution order
* To skip the test cases in between and continue with rest of the cases
* To read data from external source like XLS, XML etc.
* To generate test reports with details of screen shots, Logs.

## 1.2 Features of TestNG

* Multiple Before and after annotation options.
* We can use @DataProvider for data-driven testing
* Supported by different tools and plug-ins for ex: Eclipse, Maven, Ant etc...)
* Different reporting formats

## 1.3 Functionalities of TestNG

* We can write a Test Case
* We can write a Test Suite
* We can write Exception Tests
* Writing Parameterized Test Cases
* Prioritizing Tests
* Ignoring Tests

## 1.4 Why TestNG?

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

It is an open source automated testing framework; where NG of TestNG means Next Generation.

TestNG is similar to JUnit but it is much more powerful than JUnit but still it’s inspired by JUnit. It is designed to be better than JUnit, especially when testing integrated classes. Pay special thanks to Cedric Beust who is the creator of TestNG.

TestNG eliminates most of the limitations of the older framework and gives the developer the ability to write more flexible and powerful tests with help of easy annotations, grouping, sequencing & parametrizing.

## 1.5 Benefits of TestNG

There are number of benefits but from Selenium perspective, major advantages of TestNG are:

* It gives the ability to produce HTML Reports of execution
* Annotations made testers life easy
* Test cases can be Grouped & Prioritized more easily
* Parallel testing is possible
* Data Parameterization is possible
* Supports Data driven Testing using Data providers.
* Readily supports integration with other tools and plug ins like Eclipse IDE, build tools Ant, Maven etc...

# 2. Commonly used Terminologies used in TestNG

* Group Test
* Dependency Test
* Priority

## 2.1 TestNG “Groups”

* Each test method can be tagged with any # of groups.
* @Test // represents no groups
* @Test (groups = “groupTest1”)
* @Test (groups = { “gp1”, “gp2”, ... })
* A group therefore contains any # of test methods.
* Can also annotate test class this way; test method groups are union of any explicit test method groups, and any of its test class.
* Groups can also be externally defined (TestNG xml configuration file).
* A group is identified by a unique string (don’t use white space).
* There are no pre-defined group names.
* E.g., “fast”, “dbTest”, “uiTest”, “unitTesting”, “intTesting”, “broken. knownReason”,
* “check-in”, “front-end”, “funct. payments”

**Example:**

@Test(groups = "integration")

**public** **class** ServiceTest **extends** AbstractTestNGSpringContextTests{

@Test(groups = "broken")

**public** **void** xyz() **throws** Exception{

….}

@Test

**public** **void** abc () **throws** Exception{

…..}

}

Here the class ServiceTest is annotated with group name integration, so all the method in the class xyz() and abc() are under group integration. Here the method xyz() is annotated with group name broken, so this method is under two groups integration and broken.

**Exclusion Groups**

We can exclude groups using include or exclude keywords in the xml file. Example –

|  |
| --- |
| <test name="Simple example"> |

|  |
| --- |
| <groups> |

|  |
| --- |
| <run> |

|  |
| --- |
| <include name="checkintest"/> |

|  |
| --- |
| <exclude name="broken"/> |

|  |
| --- |
| </run> |

|  |
| --- |
| </groups> |

|  |
| --- |
|  |

|  |
| --- |
| <classes> |

|  |
| --- |
| <class name="example1.Test1"/> |

|  |
| --- |
| </classes> |

|  |
| --- |
| </test> |

## 2.2 TestNG – Dependencies

* TestNG supports the declaration of explicit dependencies between test methods.
* It is used to Invoke methods of a Test case in a particular order .
* When we need to share some data and state between test methods then TestNG dependency is helpful.
  + @Test(dependsOnMethods = {“method1”, “method2”, ...})
  + @Test (dependsOnGroups = {“group1”, “group2”, ...})

**Example:**

@Test(priority=1)

public void SigninTest () {

Throw new SkipException (“Skipping this test”);

}

@Test(priority=2,dependensOnMethods={“SigninTest”})

public void SignoutTest(){

// **In this example, we are skipping the signin test hence signout will be failed so we introduce the dependency concept between test cases**

}

## 2.3 Priority

**@Test(priority=0)**

**public void goToHomepage()**

**{**

**driver.get(baseURL);**

**Assert.assertEquals(driver.getTitle(), “Welcome to the new world”);**

**}**

**@Test(priority=1)**

**public void logout()**

**{**

**driver.findElement(By.linkTest(“LOG-OFF”)).click();**

**Assert.assertEquals(“LOG-ON: New World”, driver.getTitle());**

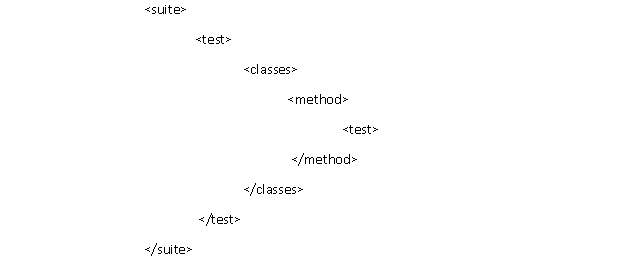
**}**

Priority 0 is the highest priority, so method of priority 0 will be executed first, then priority 1 method will be executed.

# 3. Annotations in TestNG: -

**TestNG annotations** are lines of code that are inserted in the program/ business logic to control how the methods below are to be run.

Before we begin with the different annotations available in the TestNG framework, below is the small idea on the TestNG hierarchy or annotation levels in TestNG.



The above diagram indicates that @Test is the smallest annotation here, @Method will be executed first, before and after the execution of @Test. The same way @Class will be executed first, before and after the execution of @Method and so on.

Below is the quick overview of the widely used annotations: -

**@Test: -** The method annotated with @Test will be the part of test case.

**@BeforeMethod: -** A method which is marked with this annotation will be executed before every *@test* annotated method.

**@AfterMethod: -** A method which is marked with this annotation will be executed after every *@test* annotated method.

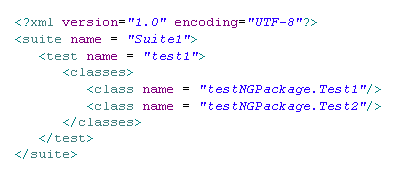
**@BeforeClass: -** A method which is marked with this annotation will be executed before the first @Test method and it runs once per class

**@AfterClass: -** A method which is marked with this annotation will be executed after all the test methods in the current class have been run

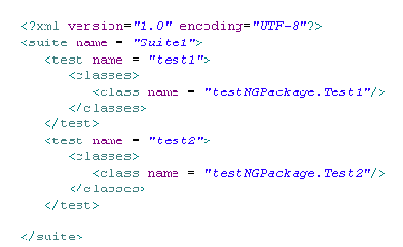
**@BeforeTest: -** A method which is marked with this annotation will be executed before **first @Test** annotated method.

**The usage of @BeforeTest annotation has been explained in detailed below with example**: -

* If there are 2 @BeforeTest annotated method in 2 different classes respectively under the same <test> tag, then both the methods annotated as @BeforeTest will be executed first before **first @Test** annotated method. The xml structure for the same is mentioned below: -



* If there are 2 @BeforeTest annotated method in 2 different classes respectively under the different <test> tag, then the first method annotated as @BeforeTest will be executed first before **first @Test** annotated method under the first <test> tag and the second method annotated as @BeforeTest will be executed first before **first @Test** annotated method under the second <test> tag. The xml structure for the same is mentioned below: -



**@AfterTest: -** A method which is marked with this annotation will be executed when **all** *@Test* annotated methods complete the execution of those classes which are inside *<test>* tag in *testng.xml* file.

**@BeforeSuite: -** A method which is marked with this annotation will run **only once before** all tests in the suite have run.

**@AfterSuite: -** A method which is marked with this annotation will run **once after** execution of all tests in the suite have run.

**@Dataprovider**: - The data provider is another annotation which supports data-driven testing. You can use it to handle a broad range of complex parameters like the following.

* Java objects.
* Objects read from a database.
* Data from Excel or property file etc.

**@Factory: -** @Factory allows tests to be created at runtime depending on certain data-sets or conditions

**Note**: Detailed explanation on @Dataprovider and @Factory is explained in further section.

# 4. Data Source in TestNG: -

A key benefit of automating functional testing is the ability to test large volumes of data on the system quickly. But you must be able to manipulate the data sets, perform calculations, and quickly create hundreds of test iterations and permutations with minimal effort. Test Automation Frameworks must have capability to integrate with spreadsheets and provide powerful calculation features.

**In TestNG we use @Factory and @DataProvider for executing the script with different sets of data, by providing the data through excel or properties file or database etc.**

## 4.1 @DataProvider Annotation: -

When you need to pass complex parameters or parameters that need to be created from Java (objects read from a property file or an Excel file, etc…), in such cases parameters can be passed using Dataproviders.

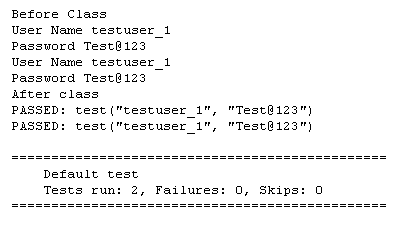
**Key features of DataProvider**: -

* A Data Provider is a method annotated with @DataProvider.
* A Data Provider returns an array of objects.
* The test method will be executed using the same instance of the test class to which the test method belongs.

**Below is the example of DataProvider: -**



On executing the above code, the output will be displayed as follows: -



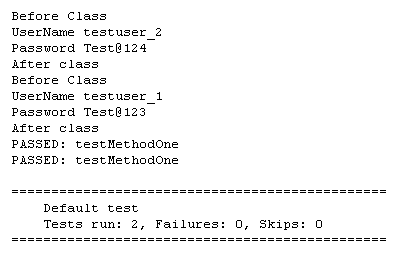
## 4.2 @Factory Annotation: -

Can be used to execute all the test methods present inside a test class, using separate instance of the same class.  That means running the class in parametrized way by providing different inputs to class constructor. A factory will execute all the test methods present inside a test class using a separate instance of the respective class.

**Below is the example of DataFactory: -**



On executing the above code, the output will be displayed as follows: -



**Combination of@ Dataprovider and @Factory for reading data through excel**: -

Combination of data providers and factory annotations in TestNG can be used to reiterate the same test class with different test data. This will enable users to use the same class seamlessly without duplicating the test class code.

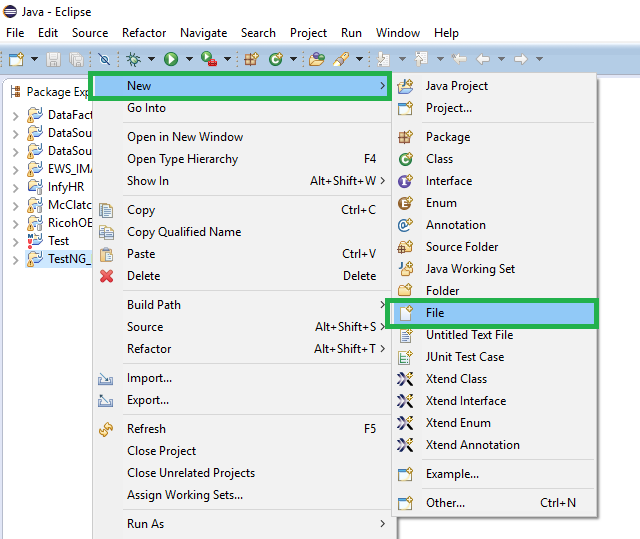
# 5. Execution through Xml in TestNG: -

* In testng.xml file we can specify multiple name (s) which needs to be executed. In a project there may be many classes, but we want to execute only the selected classes. So with the help of xml, it becomes easy for us to select the classes which we want to execute.
* We can pass class names of multiple packages also. If say suppose, we want to execute two classes in one package and other class from some other package.
* With the help of Xml execution, we can generate reports which is user friendly and easy to understand as well as the reports can be sent via email to multiple recipient as well.
* The main advantage of the report generated via xml execution is that it **saves the effort of updating the status in the excel** for each test cases.

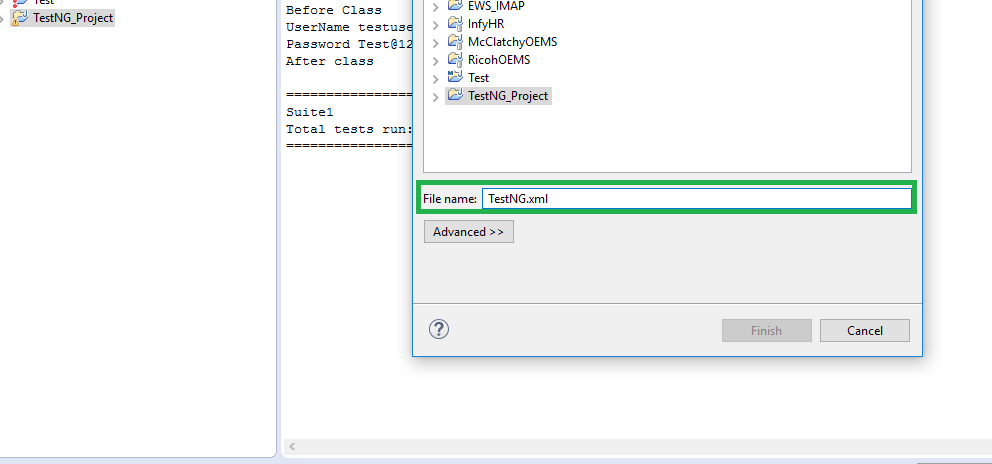
## 5.1 How to create Xml file: -

**Step 1: Create TestNG xml file**

* Right click on Project folder, go to **New** and select ‘**File** ‘as shown in below image



* In New file wizard, add file name as ‘**TestNG.xml** ‘as shown in below given image and click on Finish button.



* It will add TestNG xml file under your project folder

**Step 2: Write xml code:**

* Add below given code in your TestNG xml file.

<suite name = *"Suite1"*>

<test name = *"test1"*>

<classes>

<class name = *"packageName.ClassName"*/>

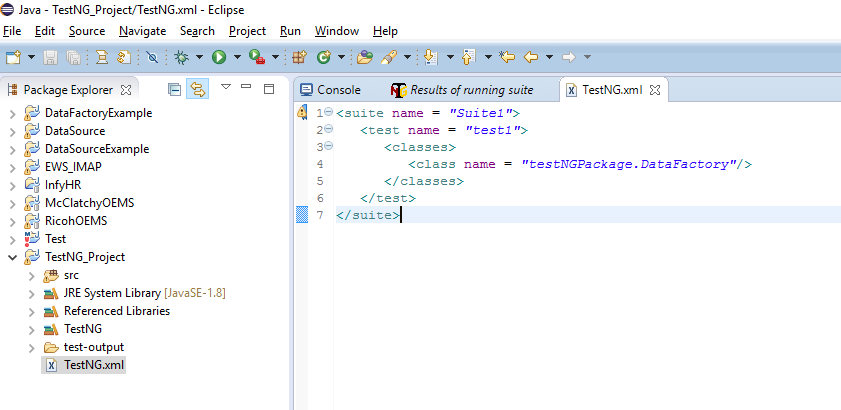
</classes>

</test>

</suite>

**Note:** You can choose any name for your Test Suite & Test Name as per your need.

* After giving appropriate names, now your TestNG xml file will looks like this:

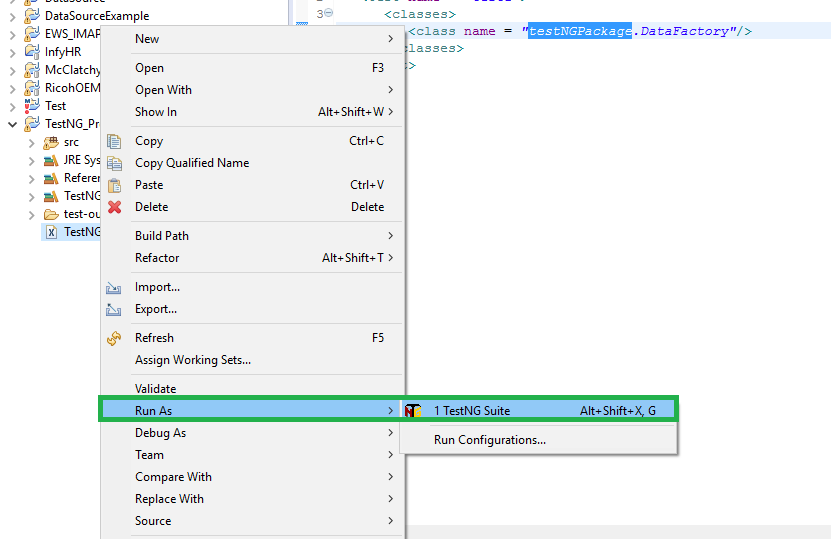


Very first tag is the Suite tag*<suite>*, under that it is the Test tag*<test>* and then the Class tag*<classes>*. You can give any name to the *suite* and the *test* but you need to provide the correct name to the *<classes>* tag which is a combination of your **Package** name and **Test Class** name.

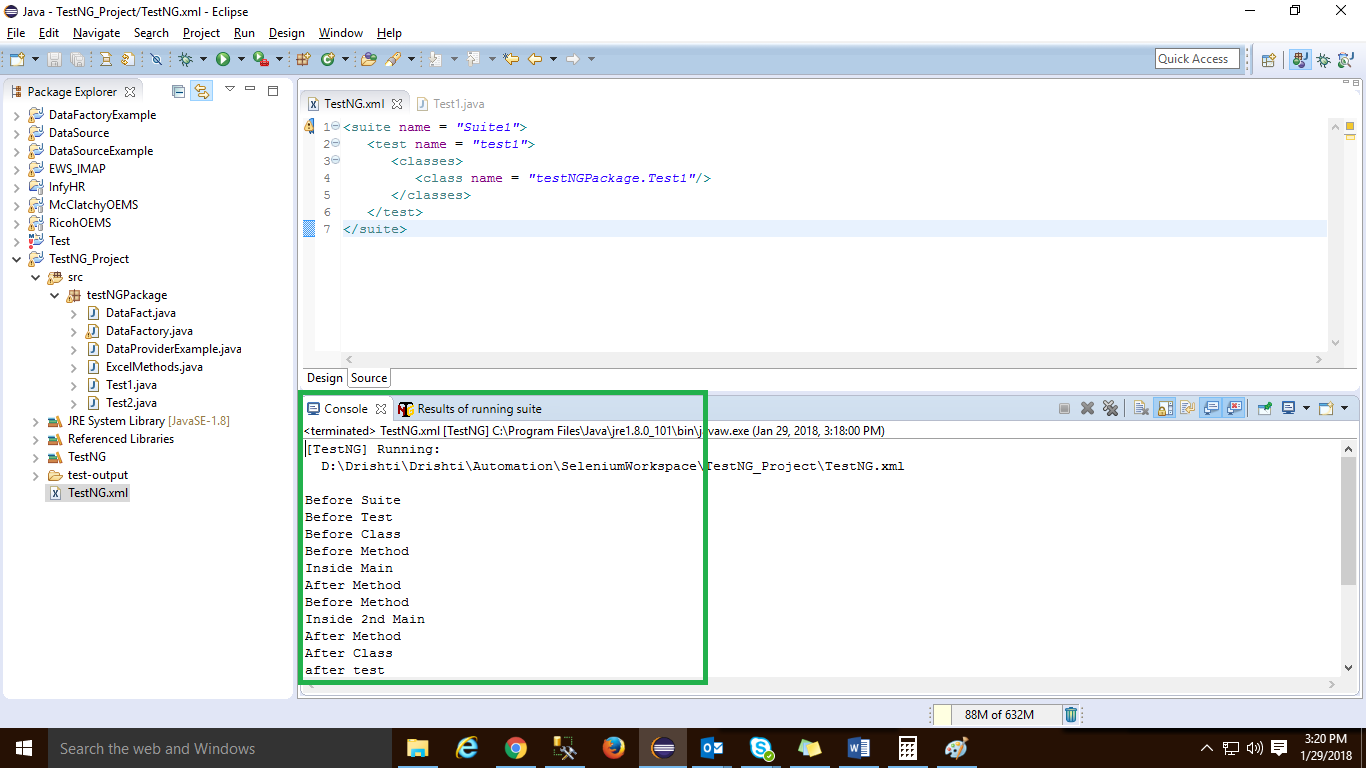
e.g. **Package Name** is *“testNGPackage”*, **Test Class Name** is *“DataFactory”*. So the C**lass Name** should be *testNGPackage. DataFactory*

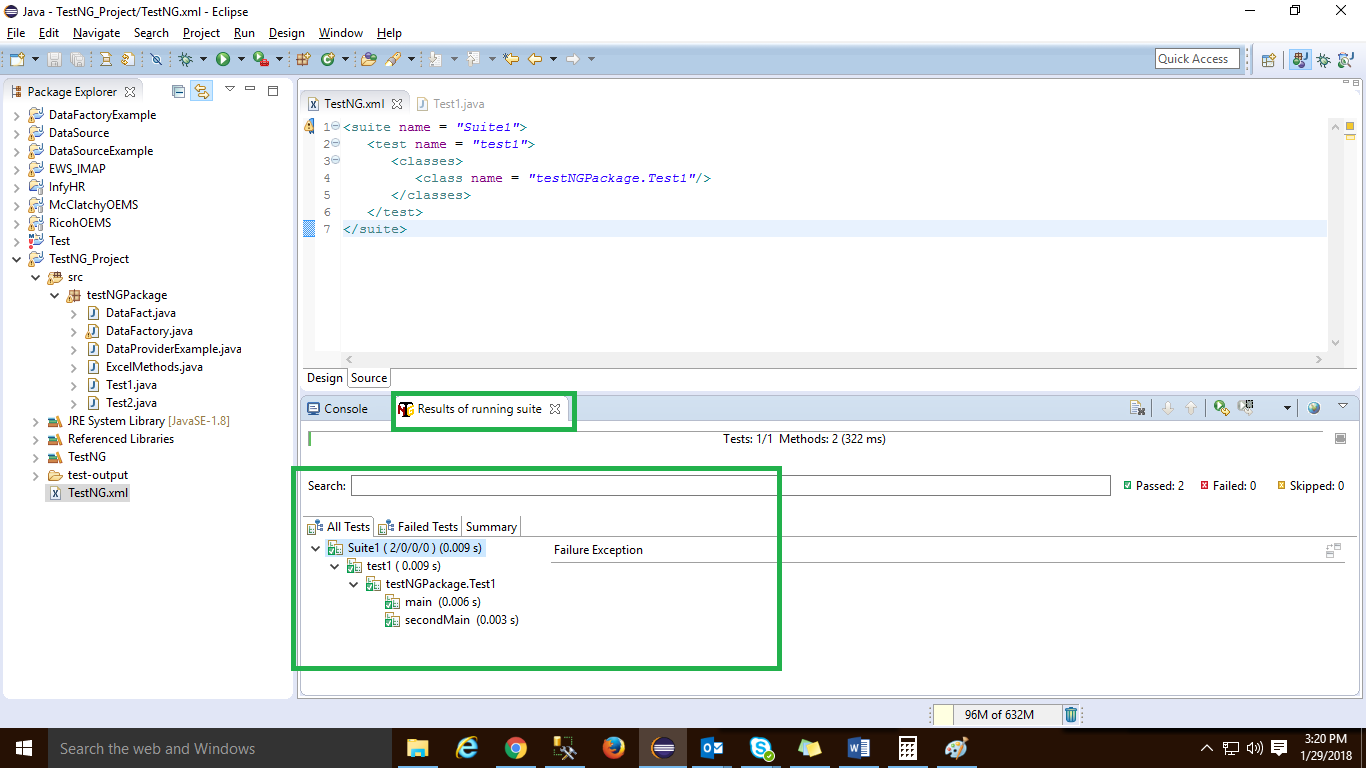
**Step 3: Execute a TestNG xml**

Now let’s run the xml. Run the test by right click on the TestNG.xml file and select **Run as > TestNG Suite.**

**

Once the execution is done, you could view test result under the TestNG console.

**

**

**Important features to remember for execution through Xml: -**

* We need to specify the class names along with packages in between the classes tags.
* We can pass class names of multiple packages also.  If say suppose, we want to execute two classes in one package and other class from some other package. Below mentioned example illustrate the same: -

<suite name = *"Suite1"*>

<test name = *"test1"*>

<classes>

<class name = *"testNGPackage.Test1"*/>

<class name = *"testNGPackage.Test2"*/>

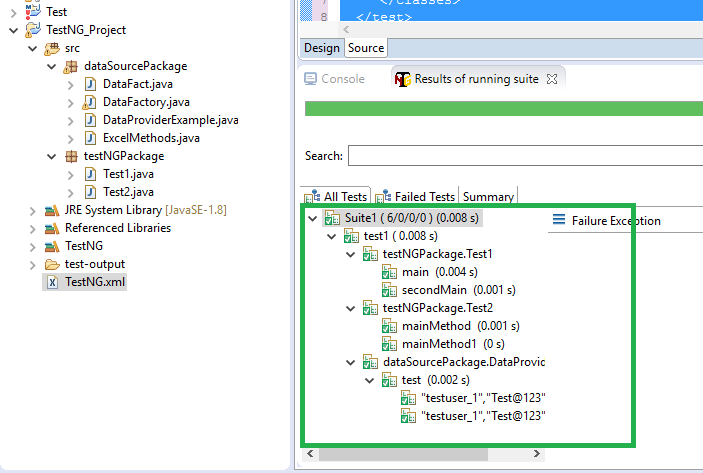
<class name = *"dataSourcePackage.DataProviderExample"*/>

</classes>

</test>

</suite>

The output will be generated as follow: -



* All the classes specified in the xml will get executed which have TestNG annotations.

# 6 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

1. **Index.html**
2. **emailable-report.html**

## 6.1 Why do we need reporting? -

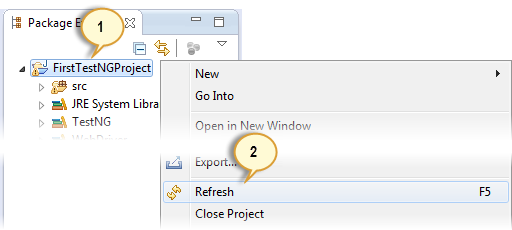
When we are using Selenium or any other automation tool, we are performing operations on the web application. But our purpose of automation is not just to exercise the Application Under Test. We, as an automation tester are supposed to test the application, find bugs and report it to the development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

**Index.html: -**

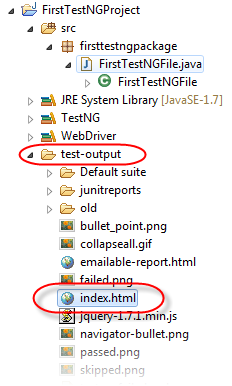
**Generating HTML Reports**

TestNG has the ability to generate reports in HTML format.

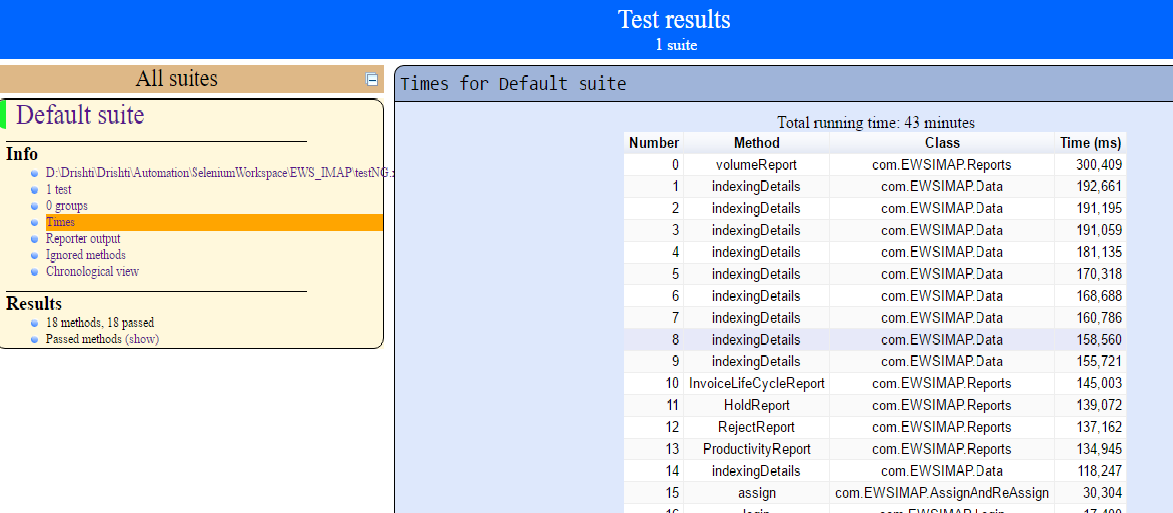
**Step 1:**After running our FirstTestNGFile that we created in the previous section, right-click the project name (FirstTestNGProject) in the Project Explorer window then click on the "Refresh" option.

[](https://cdn.guru99.com/images/step_1-0019.png)

**Step 2:**Notice that a "test-output" folder was created. Expand it and look for an index.html file. This HTML file is a report of the results of the most recent test run.

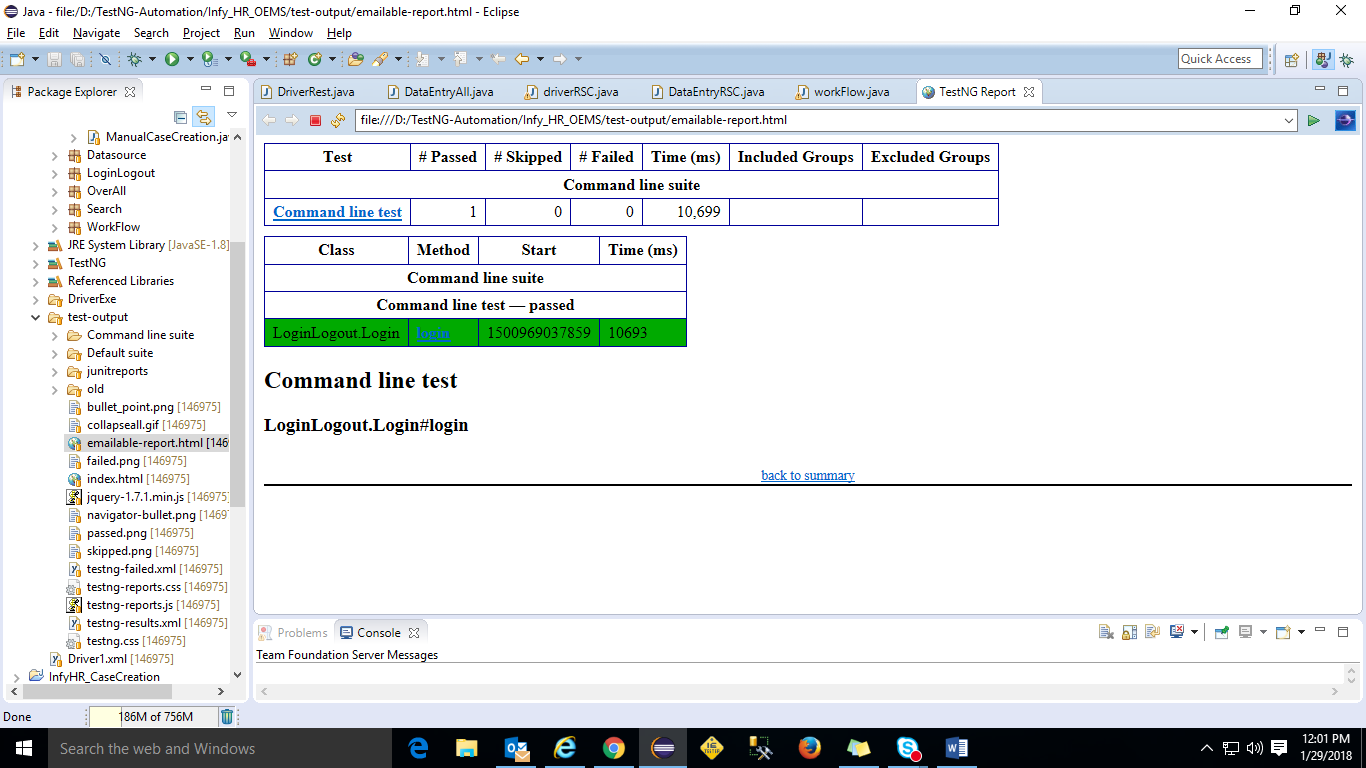
[](https://cdn.guru99.com/images/test-output___index-0022.png)

**Step 3:**Double-click on that index.html file to open it within Eclipse's built-in web browser. You can refresh this page any time after you rerun your test by simply pressing F5 just like in ordinary web browsers.



**Emailable report: -**

Open ‘**emailable-report.html** ‘, as this is a html report open it with browser.



# 7. Installation of TestNG: -

****

**Reference(s)**

* [**https://howtodoinjava.com/testng/testng-difference-between-factory-and-dataprovider/**](https://howtodoinjava.com/testng/testng-difference-between-factory-and-dataprovider/)
* [**https://www.softwaretestingmaterial.com/testng-tutorial/**](https://www.softwaretestingmaterial.com/testng-tutorial/)
* [**http://toolsqa.com/selenium-webdriver/**](http://toolsqa.com/selenium-webdriver/)
* [**http://testng.org/doc/**](http://testng.org/doc/)