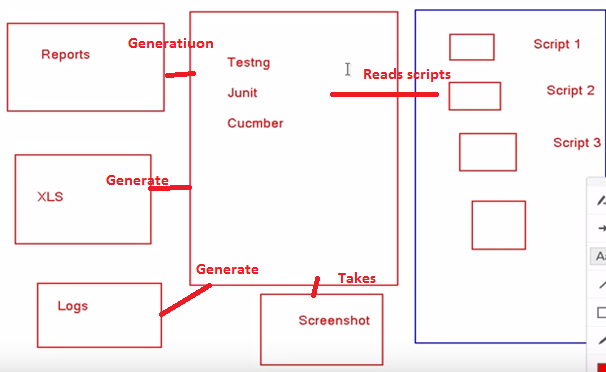
* TestNG/JUnit/Cucumber reads the scripts and generates Reports, xls, logs and also takes screenshots.

### ****Overall Picture****



### ****Features of TestNG****

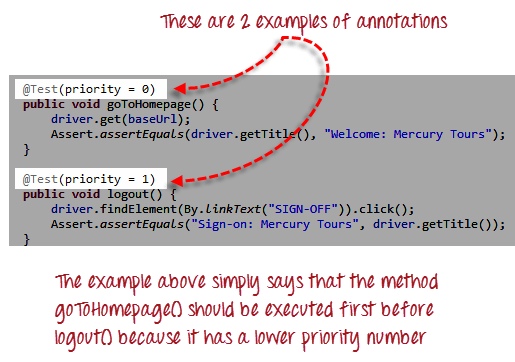
* Support for **annotations**
* Support for **parameterization**
* Advance execution methodology that do not require test suites to be created
* Support for Data Driven Testing using **Data providers**
* Enables user to **set execution priorities** for the test methods
* Supports threat safe environment when executing multiple threads
* Readily supports integration with various tools and plug-ins like build tools (**Ant, Maven** etc.), Integrated Development Environment (Eclipse).
* Facilitates user with effective means of Report Generation using **ReportNG**.

### ****Advantages of TestNG over JUnit****

1. Annotations are easier to understand
2. Test cases can be grouped more easily
3. Parallel testing is possible
4. TestNG has built in HTML report and XML report generation facility. It has also built in  logging facility

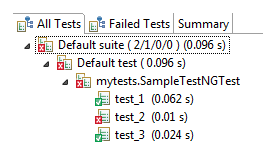
* **Annotations in TestNG are lines of code that can control how the method below them will be executed**.

They are always preceded by the @ symbol.

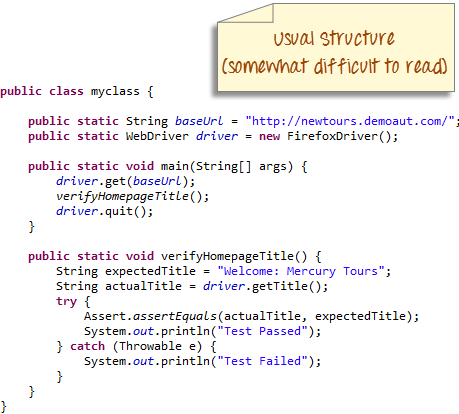


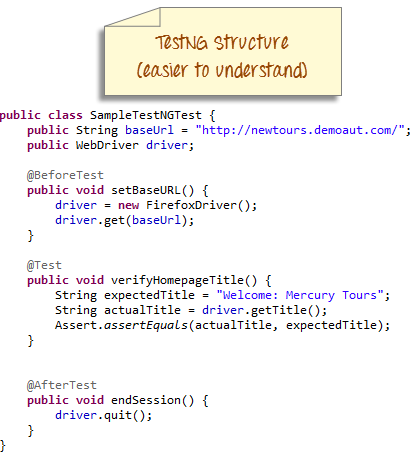
### ****Why do we need TestNG in Selenium****

1. Web Driver has no native mechanism for **generating reports**. Hence TestNG can generate reports based on our Selenium test results.



1. There is no more need for a **static main method** in our tests.



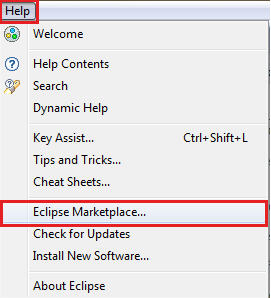


1. Uncaught exceptions are automatically handled by TestNG without terminating the test prematurely. These exceptions are reported as failed steps in the report.

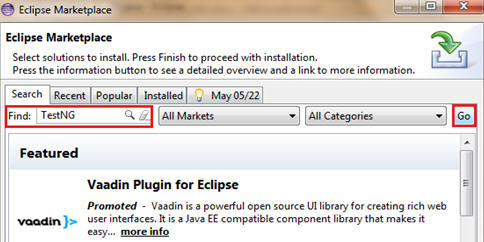
### ****TestNG Installation in Eclipse****

**Follow the below steps to TestNG Download and installation on eclipse:**

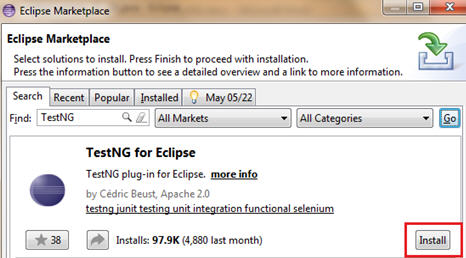
**Step 1:**Launch eclipse IDE -> Click on the Help option within the menu -> Select “Eclipse Marketplace.” option within the dropdown.

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-TestNG-tutorial-1.jpg)

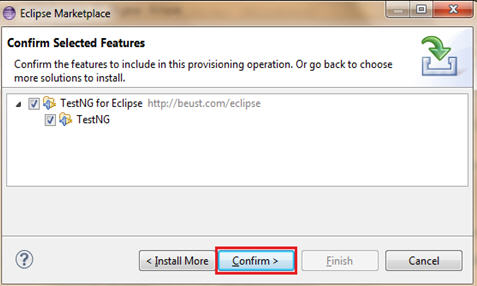
**Step 2:** Enter the keyword “TestNG” in the search textbox and click on “Go” button as shown below.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-TestNG-tutorial-2.jpg)

**Step 3:** As soon as the user clicks on the “Go” button, the results matching to the search string would be displayed. Now user can click on the Install button to install TestNG.

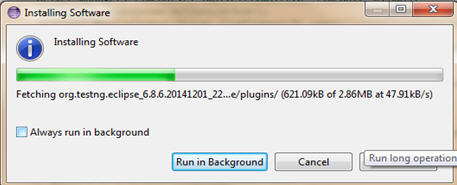
[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-TestNG-tutorial-3.jpg)

**Step 4:** As soon as the user clicks on the Install button, the user is prompted with a window to confirm the installation. Click on “Confirm” button.

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-TestNG-tutorial-4.jpg)

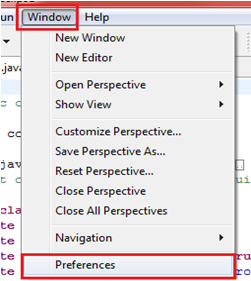
**Step 5:** In the next step, the application would prompt you to accept the license and then click on the “Finish” button.

**Step 6:** The installation is initiated now and the progress can be seen as following:

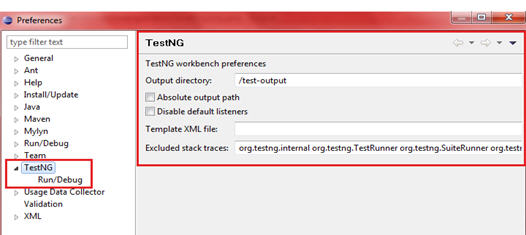
[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-TestNG-tutorial-5.jpg)

We are advised to restart our eclipse so as to reflect the changes made.

Upon restart, user can verify the TestNG installation by navigating to “Preferences” from “Window” option in the menu bar. Refer the following figure for the same.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-TestNG-tutorial-6.jpg)

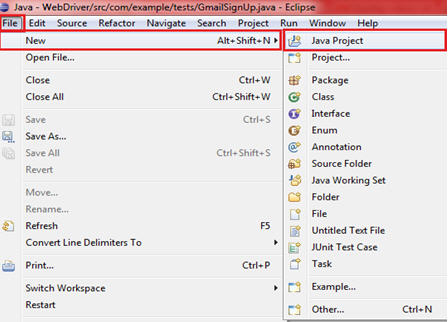
(Click on image to view enlarged)

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-TestNG-tutorial-7.jpg)

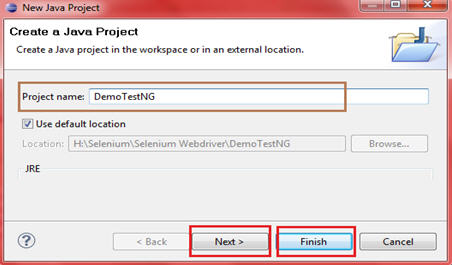
### ****Creation of Sample TestNG project****

Let us begin with the creation of TestNG project in eclipse IDE.

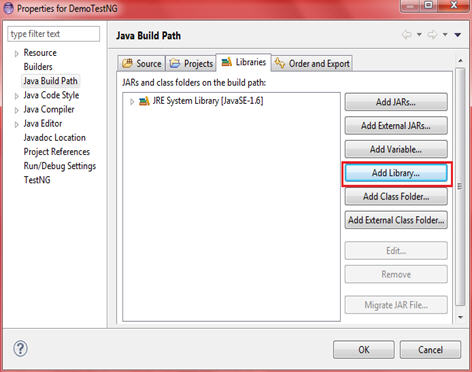
**Step 1:** Click on the File option within the menu -> Click on New -> Select Java Project.

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-TestNG-tutorial-8.jpg)

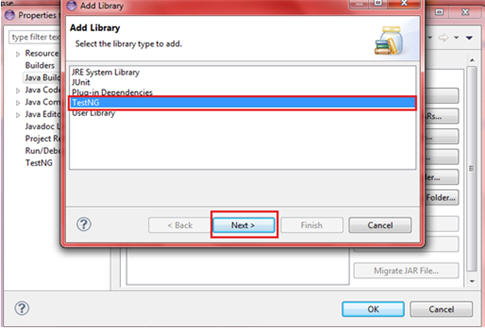
**Step 2:** Enter the project name as “**DemoTestNG**” and click on “Next” button and Click Finish.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-TestNG-tutorial-9.jpg)

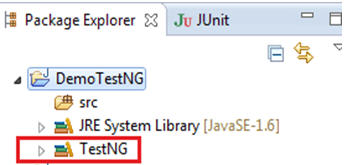
**Step 3:**  Click on “Add library” as shown below.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-TestNG-tutorial-10.jpg)

**Step 4:**  Select TestNG.

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-TestNG-tutorial-11.jpg)

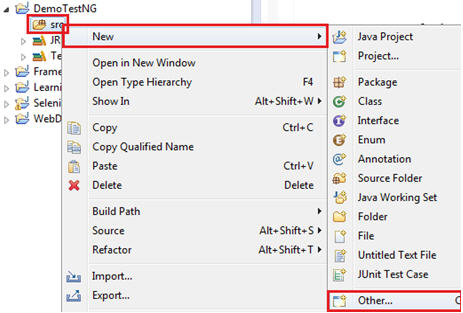
The TestNG is now added to the Java project and the required libraries can be seen in the package explorer upon expanding the project.

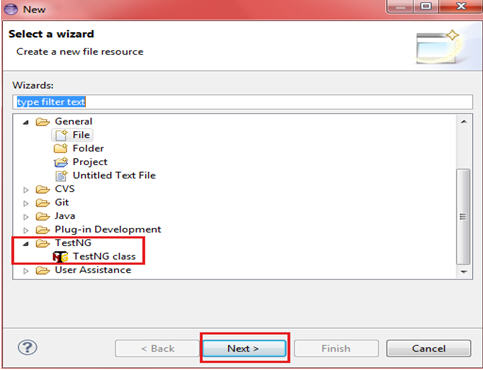
[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-TestNG-tutorial-12.jpg)

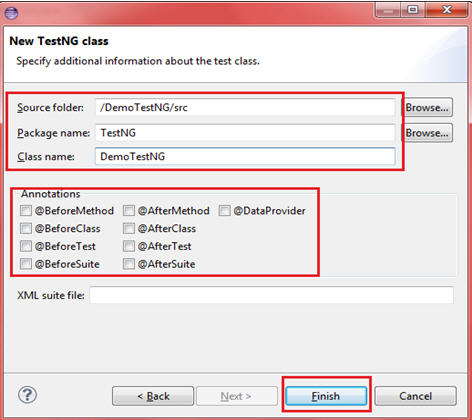
Add all the downloaded Selenium libraries and jars.

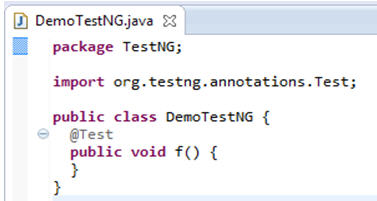
### ****Creating TestNG class****

**Step 1:** Expand the “DemoTestNG” project and go to “src” folder. Right click on the “src” package and navigate to New -> Other.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-TestNG-tutorial-13.jpg)

**Step 2:**  click “TestNG” class option [](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-TestNG-tutorial-14.jpg)

**Step 3:** Specify the Source folder, package name and the TestNG class name and click on the Finish button.[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-TestNG-tutorial-15.jpg)

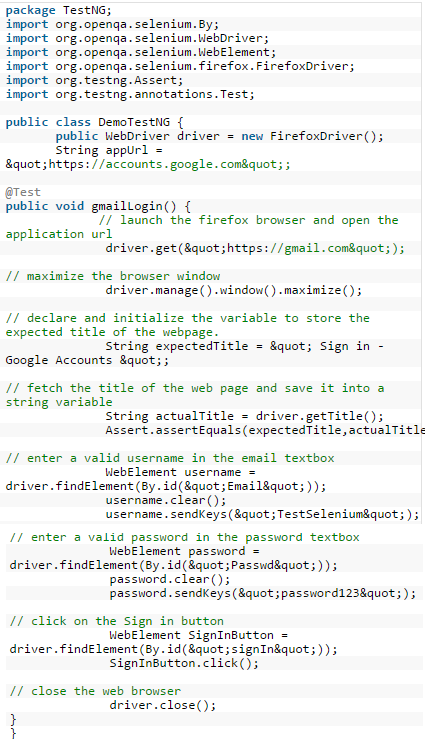
[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-TestNG-tutorial-16.jpg)

Now that we have created the basic foundation for the TestNG test script, let us now inject the actual test code. We are using the same code we used in the previous session.

**Scenario:**

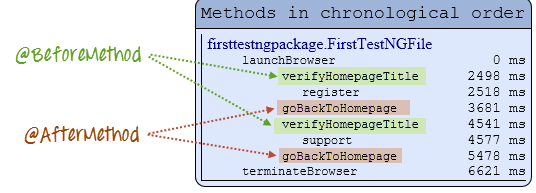
* Launch the browser and open “gmail.com”.
* Verify the title of the page and print the verification result.
* Enter the username and Password.
* Click on the Sign in button.
* Close the web browser.

**Code:**



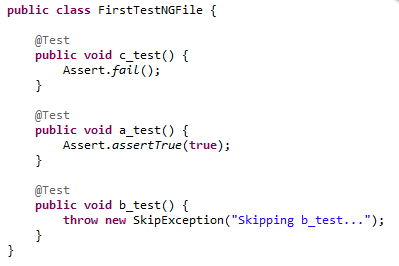
**Summary of TestNG Annotations**

* **@BeforeSuite:** The annotated method will be run before all tests in this suite have run.
* **@AfterSuite:** This method will be run after all tests in this suite have run.
* **@BeforeTest:** This method will be run before any test method belonging to the classes inside the tag is run i.e prior to the first test case in the TestNG file.
* **@AfterTest:** This method will be run after all the test methods belonging to the classes inside the tag have run i.e after all test cases in the TestNG file are executed.
* **@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.
* **@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.
* **@BeforeClass:** This method will be run before the first test method in the current class is invoked.
* **@AfterClass:** This method will be run after all the test methods in the current class have been run.
* **@BeforeMethod:** This method will be run before each test method.
* **@AfterMethod:** This method will be run after each test method.
* **@Test:** This method is a part of a test case.

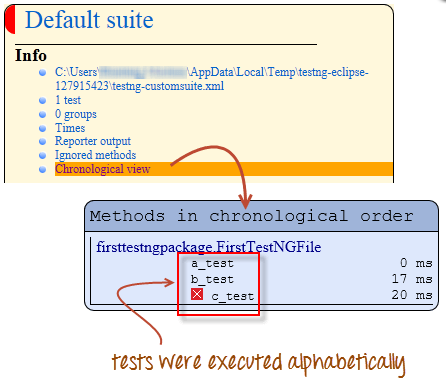


**Multiple Test Cases**

* We can use multiple @Test annotations in a single TestNG file. By default, methods annotated by @Test are executed alphabetically. See the code below. Though the methods **c\_test, a\_test, and b\_test** are not arranged alphabetically in the code, they will be executed as such.

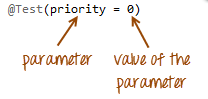


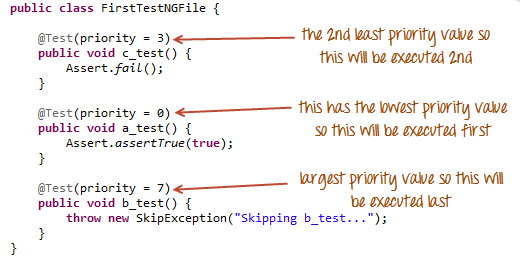
* Run this code and on the generated index.html page, click "Chronological view".



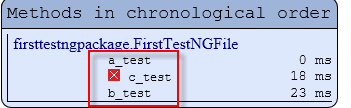
**Parameters**

* If you want the methods to be executed in a different order, use the parameter "priority".





* The TestNG HTML report will confirm that the methods were executed based on the ascending value of priority.



## Assertion

* An assertion is a Boolean expression at a specific point in a program which will be **true** unless there is a bug in the program.
* Asserts helps us to verify the conditions of the test and decide whether test has failed or passed. A test is considered successful ONLY if it is completed without throwing any exception.

## Benefits of Assertions:

* It is used to detect subtle errors which might go unnoticed.
* It is used to detect errors sooner after they occur.
* Make a statement about the effects of the code that is guaranteed to be true.

## Limitations of Assertion

* Reporting an error when it does not exist.
* Can take time to execute if it contains errors and occupies memory as well.

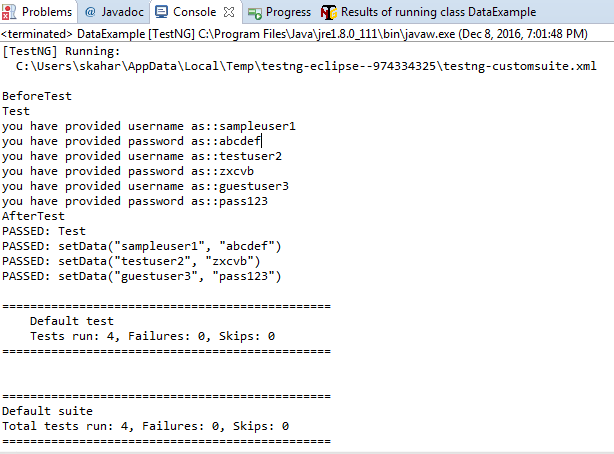
## Type of Assert statements

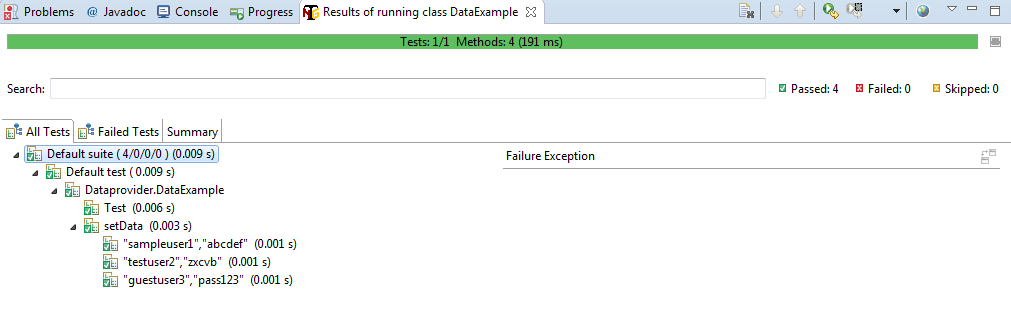
* **assertEqual(String actual,String expected) :-** It takes two string arguments and checks whether both are equal, if not it will fail the test.
* **assertEqual(String actual,String expected, String message) :-** It takes three string arguments and checks whether both are equal, if not it will fail the test and throws the message which we provide.
* **assertEquals(boolean actual,boolean expected) :-** It takes two boolean arguments and checks whether both are equal, if not it will fail the test.
* **assertEquals(java.util.Collection actual, java.util.Collection expected, java.lang.String message) :-** Takes two collection objects and verifies both collections contain the same elements and with the same order. if not it will fail the test with the given message.
* **Assert.assertTrue(condition) :-** It takes one boolean arguments and checks that a condition is true, If it isn't, an AssertionError is thrown.
* **Assert.assertTrue(condition, message) :-** It takes one boolean argument and String message. It Asserts that a condition is true. If it isn't, an AssertionError, with the given message, is thrown.
* **Assert.assertFalse(condition) :-** It takes one boolean arguments and checks that a condition is false, If it isn't, an AssertionError is thrown.
* **Assert.assertFalse(condition, message) :-** It takes one boolean argument and String message. It Asserts that a condition is false. If it isn't, an AssertionError, with the given message, is thrown.

## DataProvider in TestNG

* 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.
* The name of this data provider. If it's not supplied, the name of this data provider will automatically be set to the name of the method.
* Example: passing three different usernames and passwords
* 

**Output:**



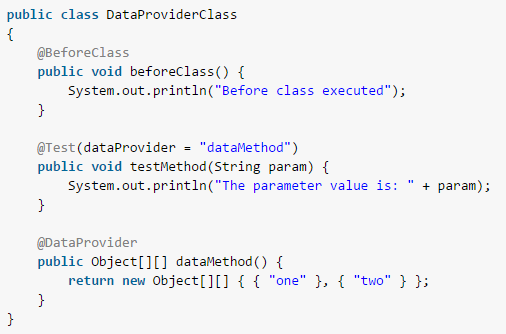


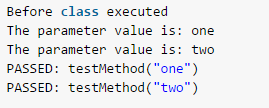
## Difference between @Factory and @DataProvider

* **DataProvider**: A test method that uses DataProvider will be executed a multiple number of times based on the data provided by the DataProvider. The test method will be executed using the same instance of the test class to which the test method belongs.
* dataProvider is used to provide parameters to a test. If you provide dataProvider to a test, the test will be run taking different sets of value each time. This is useful for a scenario like where you want to login into a site with different sets of username and password each time.
* **Factory:** A factory will execute all the test methods present inside a test class using a separate instance of the respective class.
* TestNG factory is used to create instances of test classes dynamically. This is useful if you want to run the test class any number of times. For example, if you have a test to login into a site and you want to run this test multiple times, then its easy to use TestNG factory where you create multiple instances of test class and run the tests (may be to test any memory leak issues).

**@DataProvider Example**

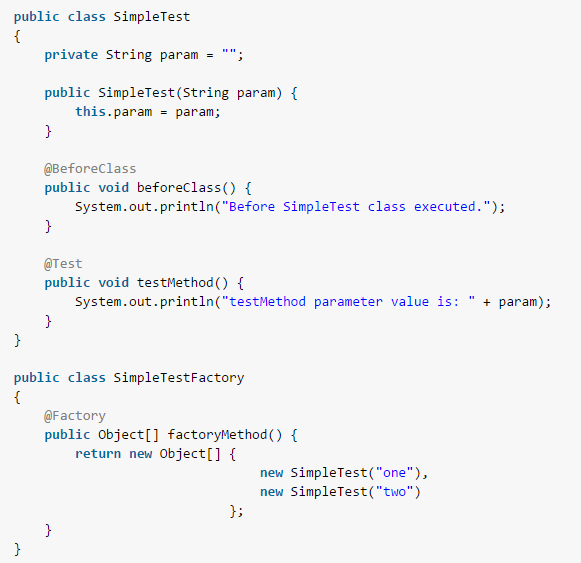
* The below class contains the testMethod and beforeClass methods. testMethod takes a String argument and the value of the argument is provided by the DataProvider method, dataMethod. The beforeClass method prints a message onto the console when executed, and the same is the case with testMethod. testMethod prints the argument passed onto it to the console when executed.

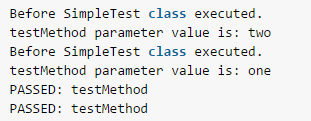




**@Factory Example**

* The below class contains the testMethod and beforeClass methods. The constructor of the test class takes a String argument value. Both beforeClass and testMethod print a message onto console.

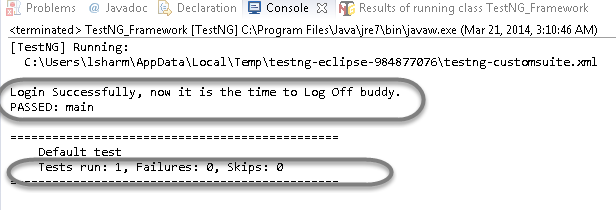


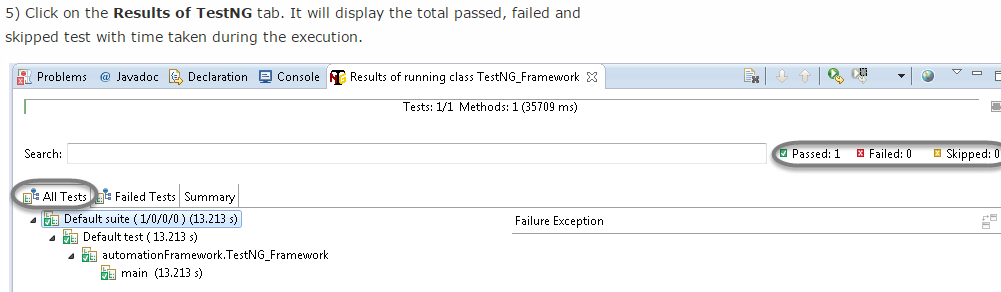


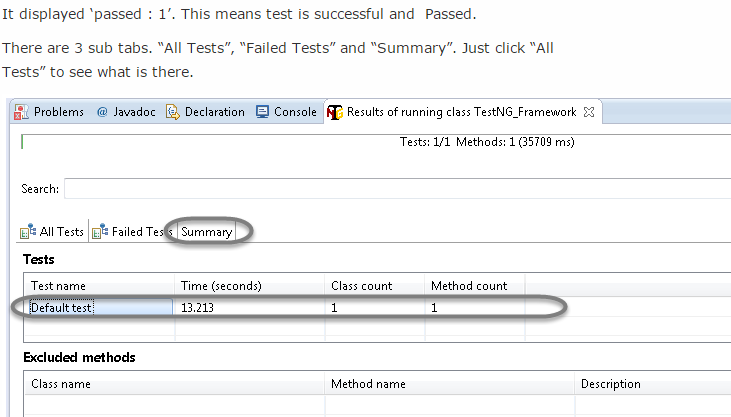
## TestNG Reporting

**Report type 1- Console output.**

Run the test by right click on the test case script and select **Run As** > **TestNG Test**.







**Report type 2- HTML report**

