**TestNG Introduction**

***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 Test****NG****means****N****ext****G****eneration*. 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.  
 

**Benefits of TestNG**

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

1. It gives the ability to produce ***HTML Reports*** of execution
2. ***Annotations***made testers life easy
3. Test cases can be ***Grouped & Prioritized*** more easily
4. ***Parallel***testing is possible
5. Generates ***Logs***
6. Data ***Parameterization***is possible

**Test Case Writing**

Writing a test in TestNG is quite simple and basically involves following steps:

**Step 1** – Write the business logic of the test

**Step 2** – Insert TestNG annotations in the code

**Step 3** – Add the information about your test (e.g. the class names, methods names, groups names etc…) in a testng.xml file

**Step 4** – Run TestNG

**Annotations in TestNG**

**@BeforeSuite**: The annotated method will be run before all tests in this suite have run.

**@AfterSuite**: The annotated method will be run after all tests in this suite have run.

**@BeforeTest**: The annotated method will be run before any test method belonging to the classes inside the tag is run.

**@AfterTest**: The annotated method will be run after all the test methods belonging to the classes inside the tag have run.

**@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**: The annotated method will be run before the first test method in the current class is invoked.

**@AfterClass**: The annotated method will be run after all the test methods in the current class have been run.

**@BeforeMethod**: The annotated method will be run before each test method.

**@AfterMethod**: The annotated method will be run after each test method.

**@Test**: The annotated method is a part of a test case.  
 

**Benefits of using Annotations**

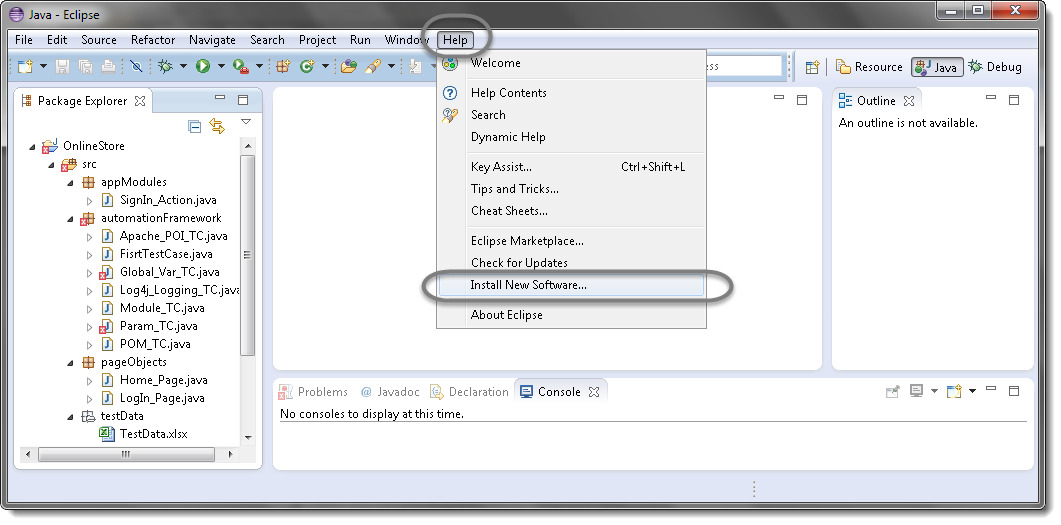
1. It identifies the methods it is interested in by looking up annotations. Hence method names are not restricted to any pattern or format.
2. We can pass additional parameters to annotations.
3. Annotations are strongly typed, so the compiler will flag any mistakes right away.
4. Test classes no longer need to extend anything (such as Test Case, for JUnit 3).

**Install TestNG in Eclipse**

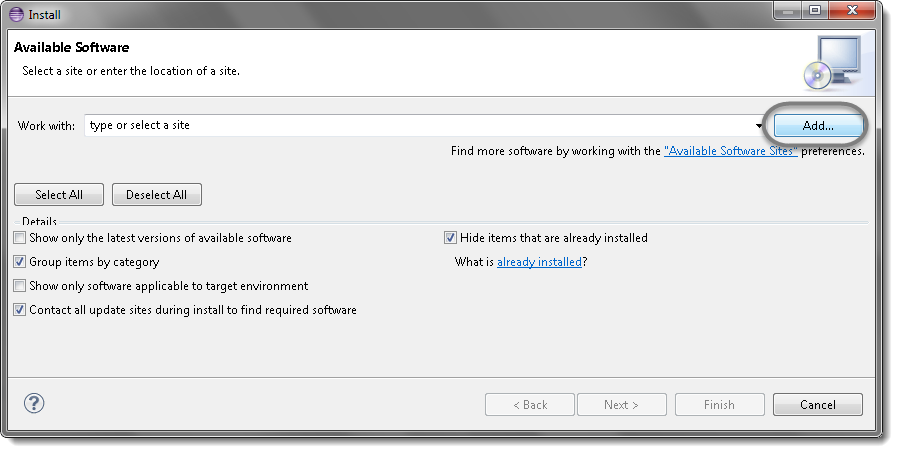
It is easy to install TestNG, as it comes as a plugin for Eclipse IDE. Prerequisite for installing TestNG is your Internet connection should be up & running during installation of this plugin and Eclipse IDE should be installed in your computer. Please see[***Download and Install Eclipse***](http://toolsqa.wpengine.com/selenium-webdriver/download-and-start-eclipse/) to setup Eclipse to you system.

**Steps to follow:**

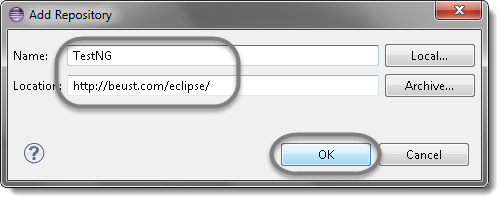
1) Launch the Eclipse IDE and from Help menu, click “**Install New Software**”.



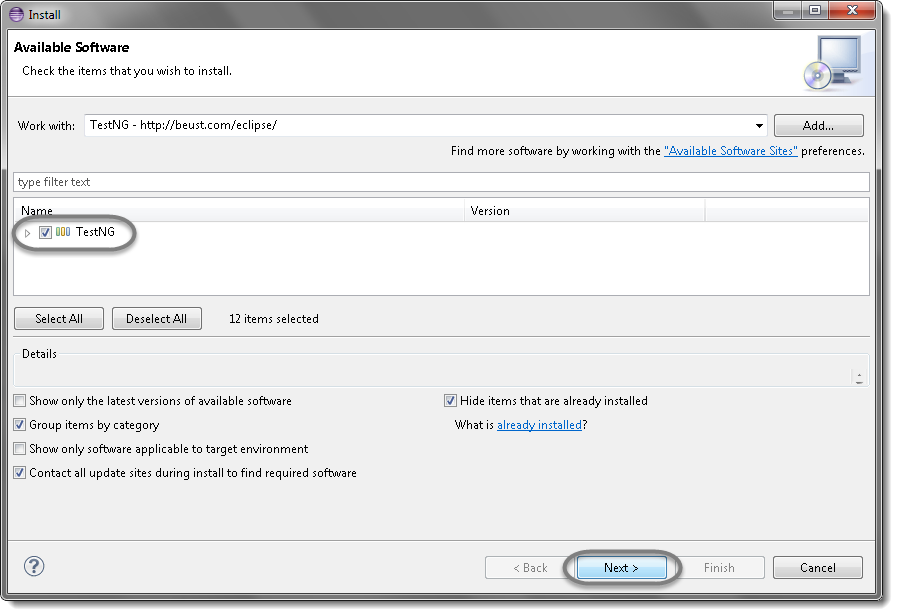
2) You will see a dialog window, click “**Add**” button.



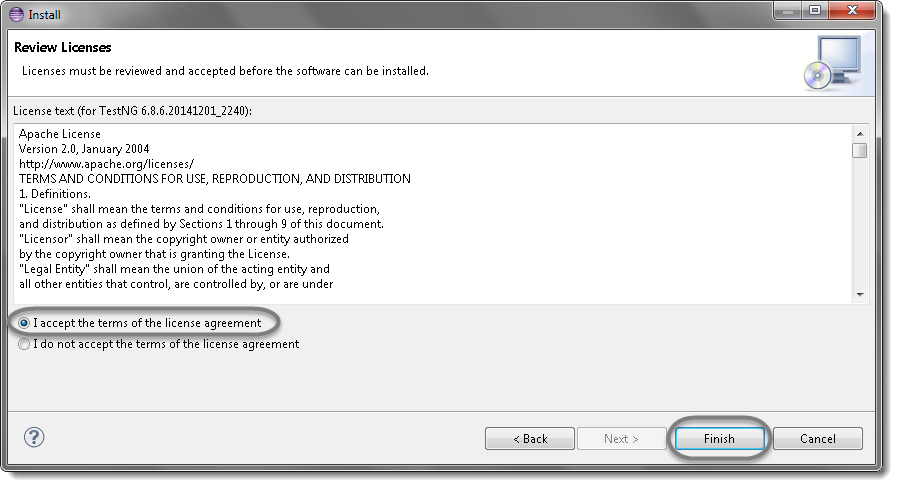
3) Type name as you wish, lets take “**TestNG**” and type “**http://beust.com/eclipse/**” as location. Click OK.



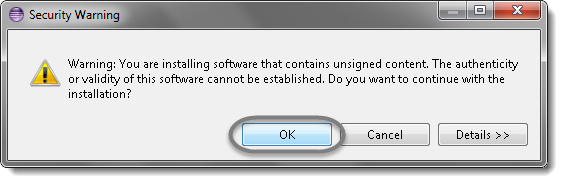
4) You come back to the previous window but this time you must see TestNG option in the available software list. Just Click TestNG and press “**Next**” button.



5) Click “**I accept the terms of the license agreement**” then click **Finish**.

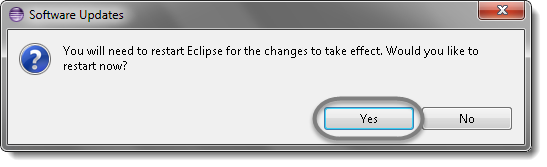


6) You may or may not encounter a Security warning, if in case you do just click **OK**.



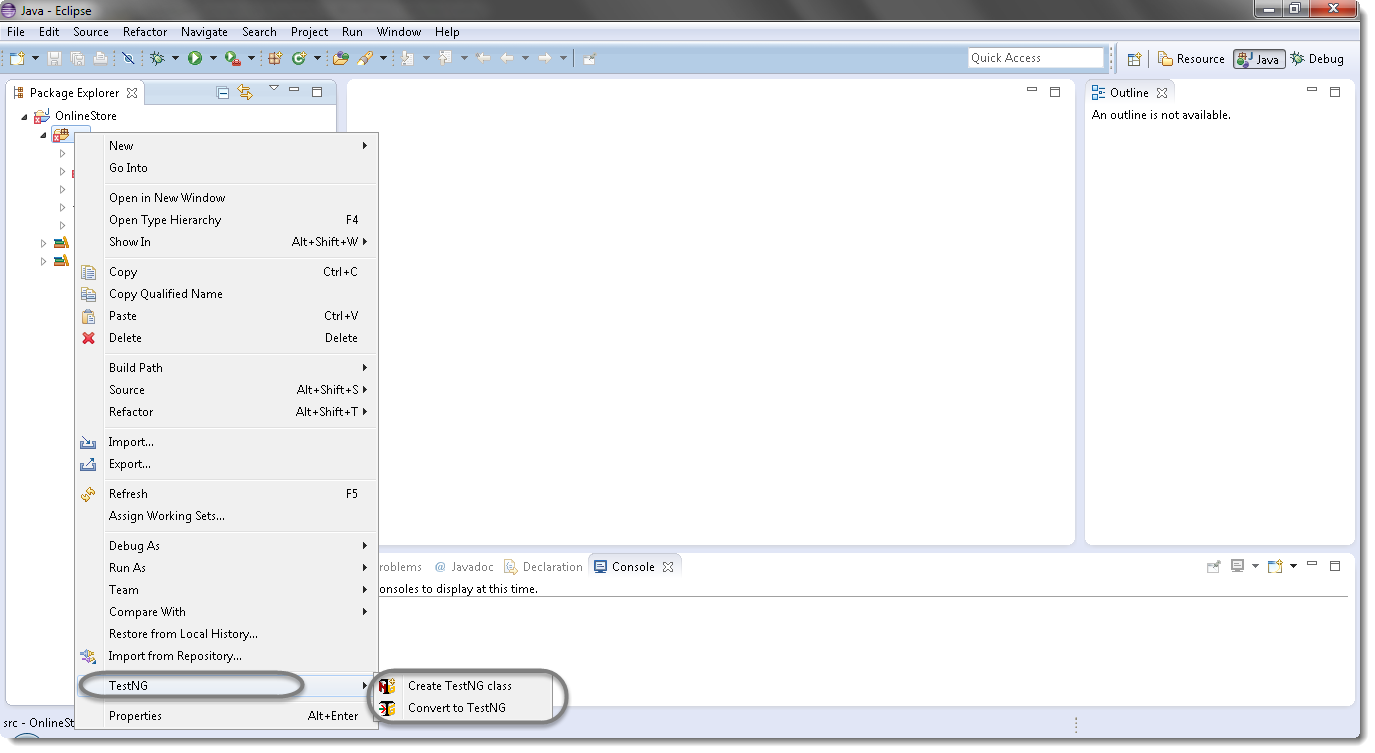
7) Click **Next** again on the succeeding dialog box until it prompts you to Restart the Eclipse.

8) You are all done now, just Click **Yes**.



9) Proceed with your workplace.

10) After restart, verify if TestNG was indeed successfully installed. Right click on you project and see if **TestNG**is displayed in the opened menu.



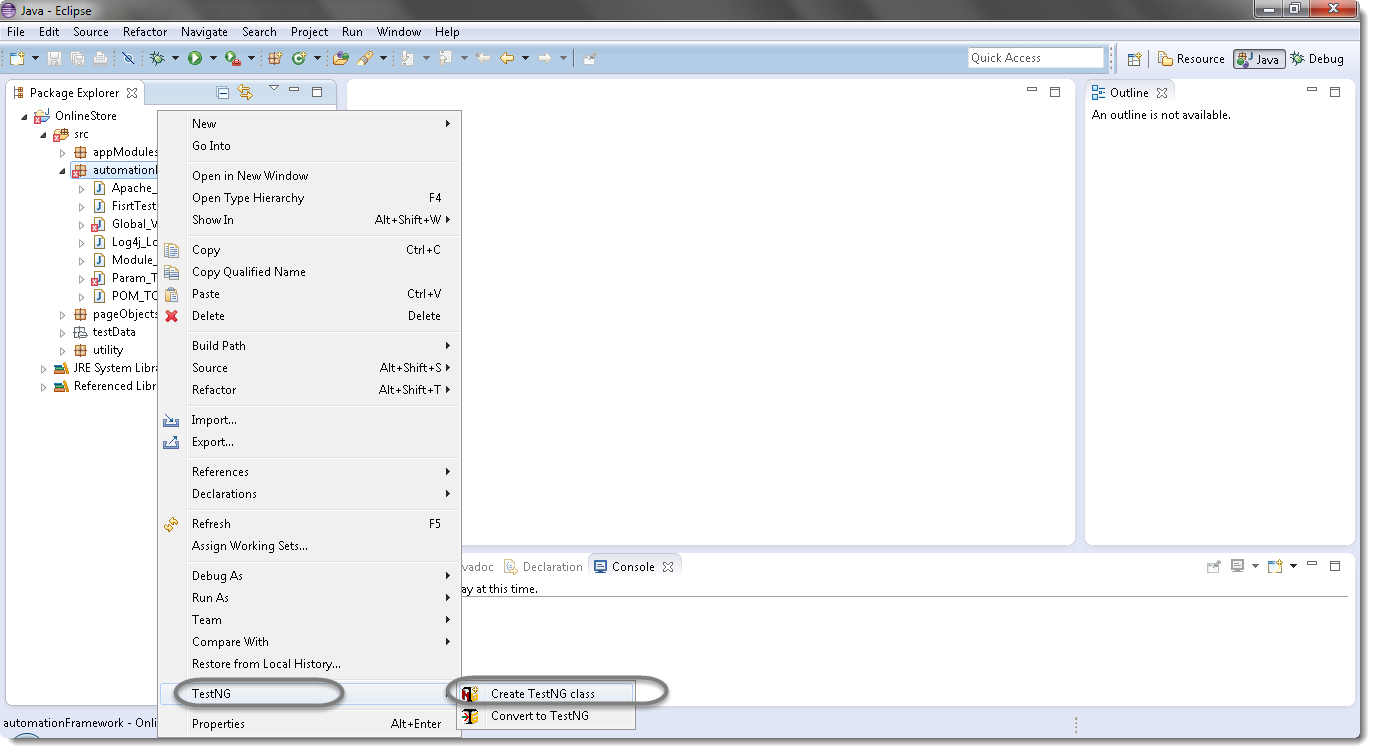
**First Test case with TestNG**

**Steps to follow:**

1) Press **Ctrl+N** , select “**TestNG Class**” under **TestNG**category and click **Next**.

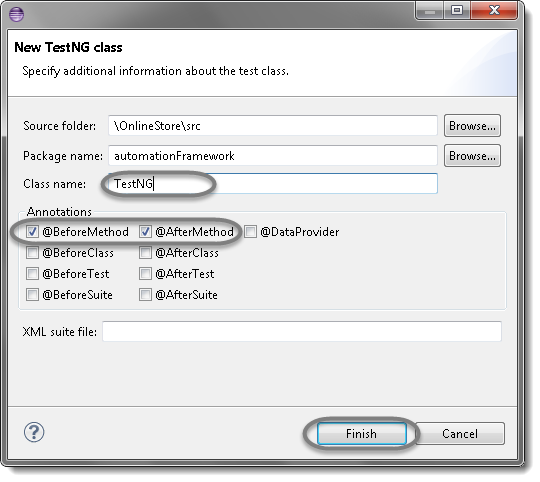
**Or**

Right click on Test Case folder, go to **TestNG**and select “**TestNG Class**“.

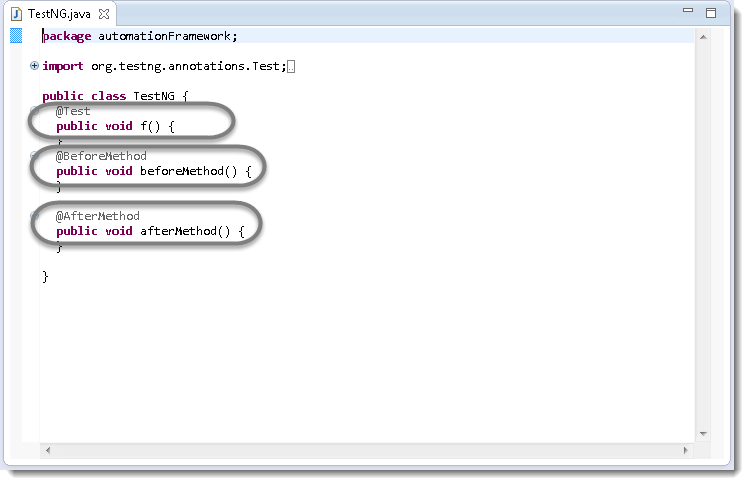


2) If your project is set up and you have selected the Test Case folder before creating TestNG class then the source folder and the package name will be prepopullated on the form. Set class name as ‘**TestNG**‘.

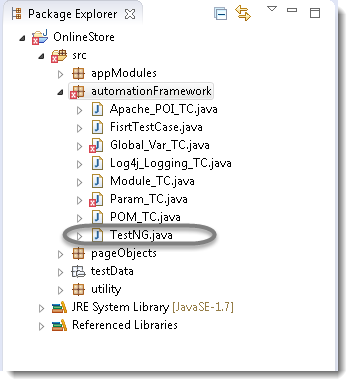
Under Annotations, check “**@BeforeMethod**”, “**@AfterMethod**” and click **Finish**. That’s it.



3) Now it will display the newly created TestNg class under the Test Case package(folder). TestNG class will look like the image below with displaying three empty methods. One method f() by default and before & after method, as selected during the creation of the class.



4) Project explorer will look like this with TestNG class.



Now it is the time to write the first TestNG test case.

5) Let’s take an example of [**First Test Case**](http://toolsqa.wpengine.com/selenium-webdriver/first-test-case/) and divide the test case in to three parts .

**@BeforeMethod** : Launch Firefox and direct it to the Base URL

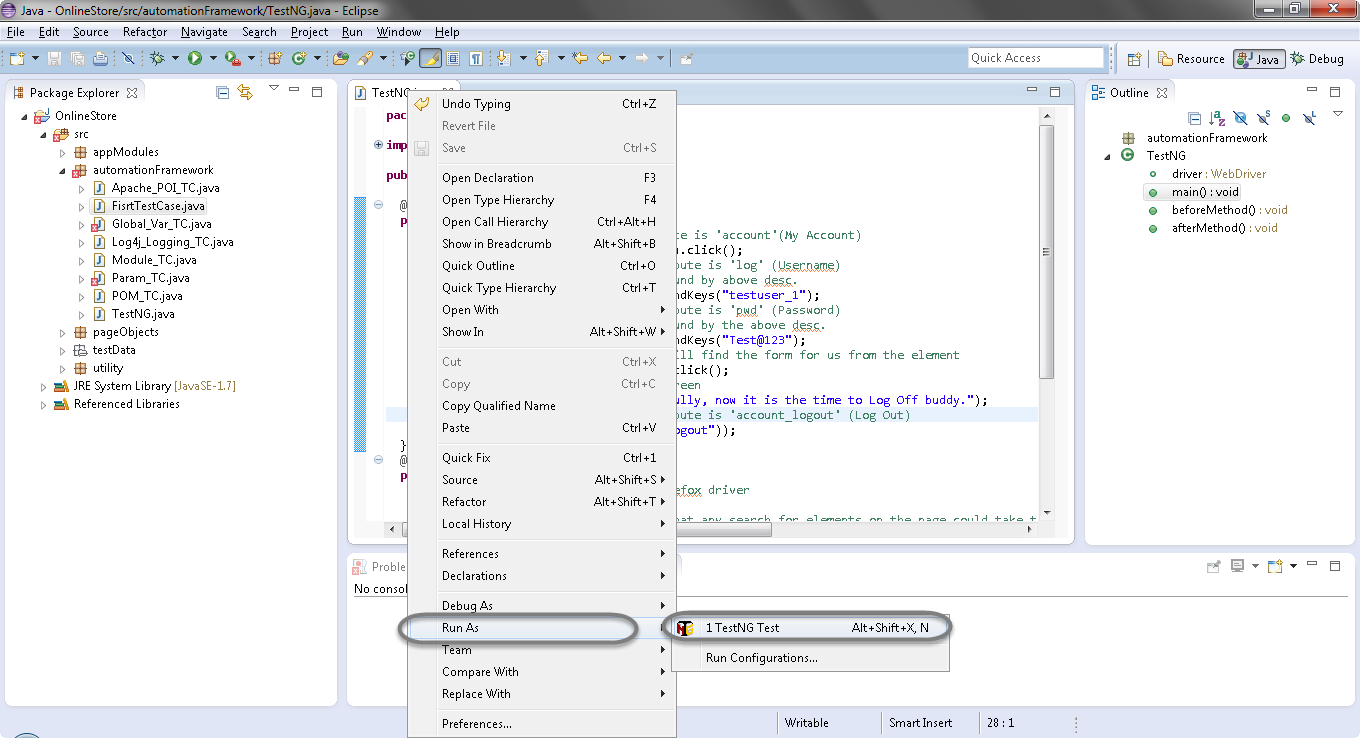
**@Test** : Enter Username & Password to Login, Print console message and Log out

**@AfterMethod** : Close Firefox browser



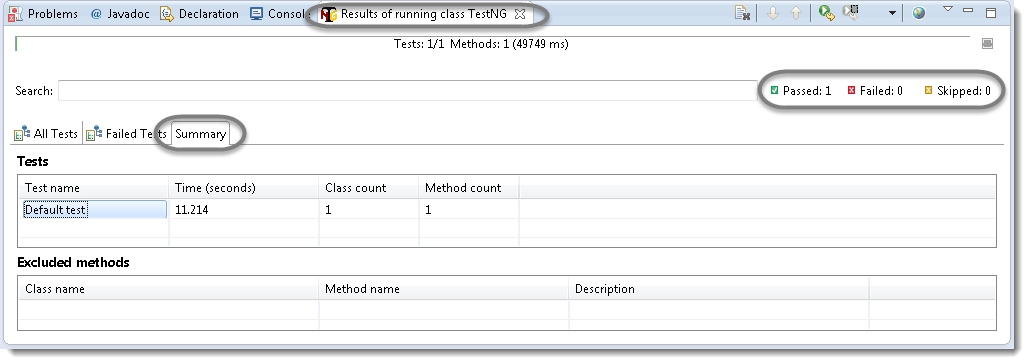
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83 | package automationFramework;    import java.util.concurrent.TimeUnit;    import org.openqa.selenium.By;    import org.openqa.selenium.WebDriver;    import org.openqa.selenium.firefox.FirefoxDriver;    import org.testng.annotations.Test;    import org.testng.annotations.BeforeMethod;    import org.testng.annotations.AfterMethod;    public class TestNG {    public WebDriver driver;      @Test      public void main() {    // Find the element that's ID attribute is 'account'(My Account)          driver.findElement(By.id("account")).click();          // Find the element that's ID attribute is 'log' (Username)          // Enter Username on the element found by above desc.          driver.findElement(By.id("log")).sendKeys("testuser\_1");          // Find the element that's ID attribute is 'pwd' (Password)          // Enter Password on the element found by the above desc.          driver.findElement(By.id("pwd")).sendKeys("Test@123");          // Now submit the form. WebDriver will find the form for us from the element          driver.findElement(By.id("login")).click();          // Print a Log In message to the screen          System.out.println(" Login Successfully, now it is the time to Log Off buddy.");          // Find the element that's ID attribute is 'account\_logout' (Log Out)          driver.findElement(By.id("account\_logout"));      }      @BeforeMethod      public void beforeMethod() {      // Create a new instance of the Firefox driver          driver = new FirefoxDriver();          //Put a Implicit wait, this means that any search for elements on the page could take the time the implicit wait is set for before throwing exception          driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);          //Launch the Online Store Website          driver.get("http://www.onlinestore.toolsqa.wpengine.com");      }      @AfterMethod      public void afterMethod() {      // Close the driver          driver.quit();      }    } |

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



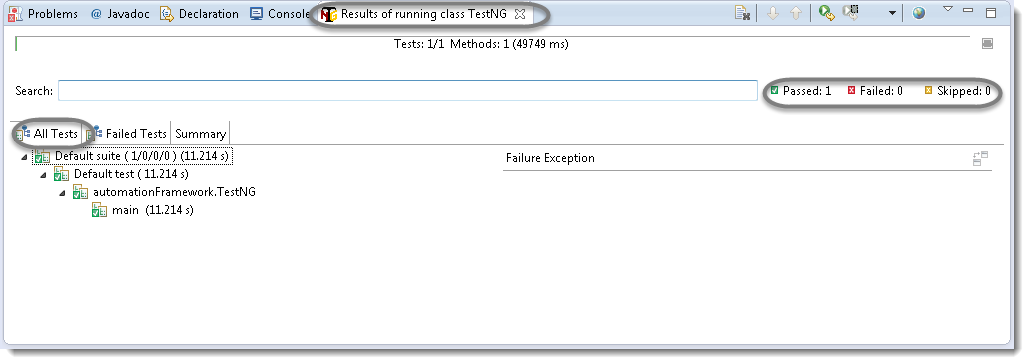
**Results of running the Testng Test Case**

7) Give it few minutes to complete the execution, once it is finished the results will look like this in the **TestNg Result**window.



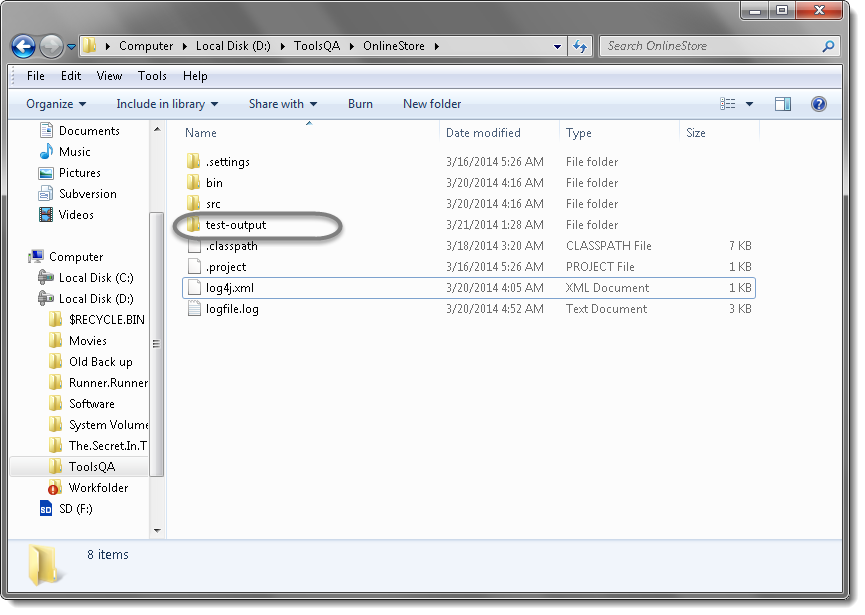
It displayed ‘passed : 1’. This means test is successful and  Passed.

There are 3 sub tabs. “All Tests”, “Failed Tests” and “Summary”. Just click “All Tests” to see what is there.

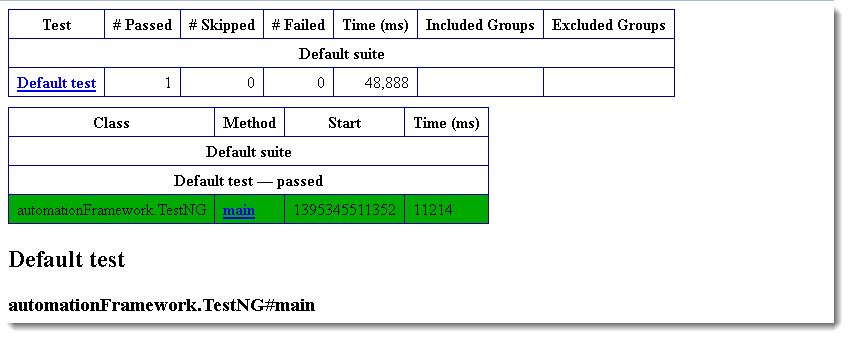


As you see, there is information of which test cases are executed and their duration. Take look to other tabs. Better than Junit right?

8) TestNG also produce HTML reports. To access those reports go to the**Project**directory and open **test-output** folder.



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



10) TestNG also produce ‘**index.html**‘ report and it resides in the same **test-output** folder. This reports gives the link to all the different component of the TestNG reports like **Groups** & **Reporter Output**. On clicking these will display detailed descriptions of execution. In the advance chapter of TestNG we will go though each of the TestNG topics.

