



Test NG

Class 1

Agenda

What is framework and why do we need it?

What is TestNG ? Why do We Require TestNG In Selenium? Advantages of TestNG?

TestNG Annotations

First TestNG script

TestNG Report

What is Framework?

A testing framework is a set of guidelines or rules used for creating and designing test cases. These guidelines could include coding standards, test-data handling methods, object repositories, processes for storing test results, or information on how to access external resources.

Framework is a collection of concepts you know in an organised manner with some proper rules, validation, proper exception and error handling which can be easily reuse, manage and scale.

In a framework we create set of multiple reusable components like
methods to launch browser, methods for clicking, sending keys, selecting, switching between frames etc.

We can create methods for reading writing from external data sources, validation, reporting etc and can place it in such a way that you and other projects can also use it. It saves efforts, time and of course money of company.

Characteristics of a framework

- Framework should be developed in such a way that any body can plug and use it. They should be able to start scripting with minimal or no setup.
- It must not be project dependent.
- It should be reusable and easy to manage and maintain.
- Flows should be clear.
- It should be simple so that an average automation tester could understand and use it.
- It should not contain scripting codes.
- It should be easily extendable i.e. enhancing.
- It should be platform independent.

Benefits of Test Automation Framework

Utilizing a framework for automated testing will increase a team's test speed and efficiency, improve test accuracy, and will reduce test maintenance costs as well as lower risks. They are essential to an efficient automated testing process for a few key reasons:

- Improved test efficiency
- Lower maintenance costs
- Minimal manual intervention
- Maximum test coverage
- Reusability of code

TestNG

Introduction

TestNG is a unit test framework designed for testing needs, TestNG is designed to cover all categories of tests: unit, functional, end-to-end, integration, etc.

It is inspired from JUnit and Nunit by adding new functionalities, which made TestNG more powerful than other unit test frameworks.

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

TestNG gives ability to write more flexible and powerful tests with help of easy annotations, grouping, sequencing & parametrizing.

TestNG provides a lot of advantage like parallel execution, Test Dependencies, Set Execution Order, Divide the tests in groups like smoke, sanity, regression and execute the group tests only.

The best feature of TestNG over Junit is TestNG report. TestNG generates the HTML report for Test Execution.

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 (You can Set the Execution Order) more easily
- Parallel testing is possible
- Generates Logs
- Data Parameterization is possible

Benefits of TestNG

- TestNG comes with nice Reporting features
- We can configure dependent test methods in TestNG means TestTwo() is dependent to TestOne().
- We can also configure that if earlier test method (TestOne()) fails during execution then dependent software test method(TestTwo()) has to be executed or not.
- There is not required to extend any class in TestNG.
- We can configure our test suite using test methods, classes, packages, groups, etc.. in single testng.xml file.
- TestNG is supported by many tools and plugins like Eclipse, Maven, IDEA, etc..
- TestNG has built in HTML report and XML report generation facility. It has also built in logging facility.

Writing test case using TestNG

Test case writing from TestNG include following steps:

- Write the Business Logic of Test
- Insert TestNG annotations in your Test
- Modify TestNG.XML file. Add your Test information like - groups, classes, methods
- Execute TestNG

What is annotation in TestNG ?

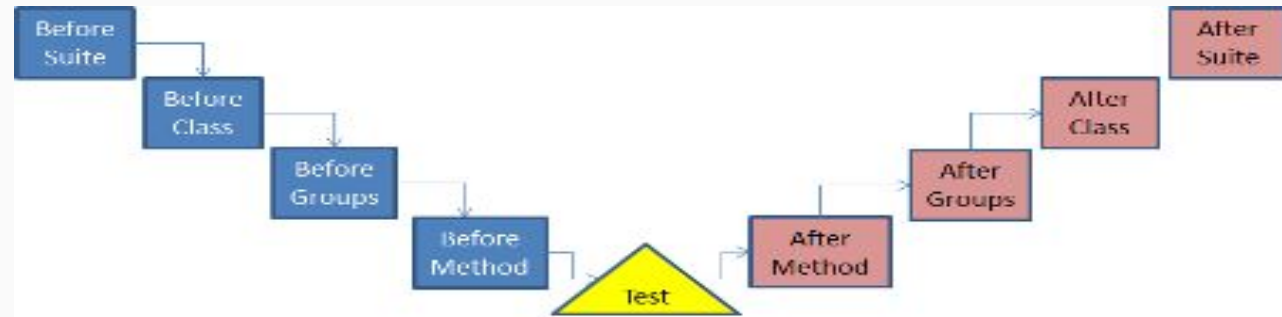
To execute TestNG script we don't have to write the separate class. We can use simple java class but here we will not write public static void main(String []args) because we are not going to execute this from JVM.

TestNG works with Annotations and annotation can be represented by @ symbol

An annotation is a tag or meta-data that provides additional information about class, interface, or method in TestNG.

Annotations are used to keep the structure of the test classes; these annotations invoke the methods according to the invocation time.

Annotations in TestNG



Test class based annotations

Annotation	Description
@BeforeClass	The annotated method will be executed Before any of the test method of a test class.
@AfterClass	The annotated method will be executed After any of the test method of a test class.
@BeforeGroups	The annotated method will be executed Before any of the test method belongs to specified group.
@AfterGroups	The annotated method will be executed After any of the test method belongs to specified group.
@BeforeMethod	The annotated method will be executed Before each test method.
@AfterMethod	The annotated method will be executed After each test method.
@Test	Marks a class or a method as a test method. If used at class level, all the public methods of a class will be considered as a test method.

Test class based annotations

Annotation	Description
@Parameters	This annotation is used to pass parameters to a test method. These parameter values are provided using the testng.xml configuration file at run time.
@DataProvider	Marks a method as a data providing method for a test method. The said method has to return an Object double array (Object[][]) as data.
@Factory	Marks a annotated method as a factory that returns an array of class objects (Object[]). These class objects will then be used as test classes by TestNG. This is used to run a set of test cases with different values. When you wants to execute specific group of test cases with different values, you need to use @Factory annotation
@Listeners	Applied on a test class. Defines an array of test listeners classes extending org.testng.ITestNGListener. Helps in tracking the execution status and logging purpose.

Testng.xml based annotations

Annotation	Description
@BeforeSuite	The annotated method will be executed Before any tests declared inside the TestNG suite.
@AfterSuite	The annotated method will be executed After any tests declared inside the TestNG suite.
@BeforeTest	The annotated method will be executed Before test section declared inside the TestNG suite.
@AfterTest	The annotated method will be executed After test section declared inside the TestNG suite.

@Test Annotation in TestNG

The most important annotation of TestNG is Test annotation. This annotation marks a method or class as TestNG test.

@Test annotation makes a method to run as the base for the TestNG annotations, method annotated with @test will hold the test logic of the business and assertions.

Every @Test annotated methods should correspond to a manual test case, sometimes some teams may map every @test with a requirement.

Task

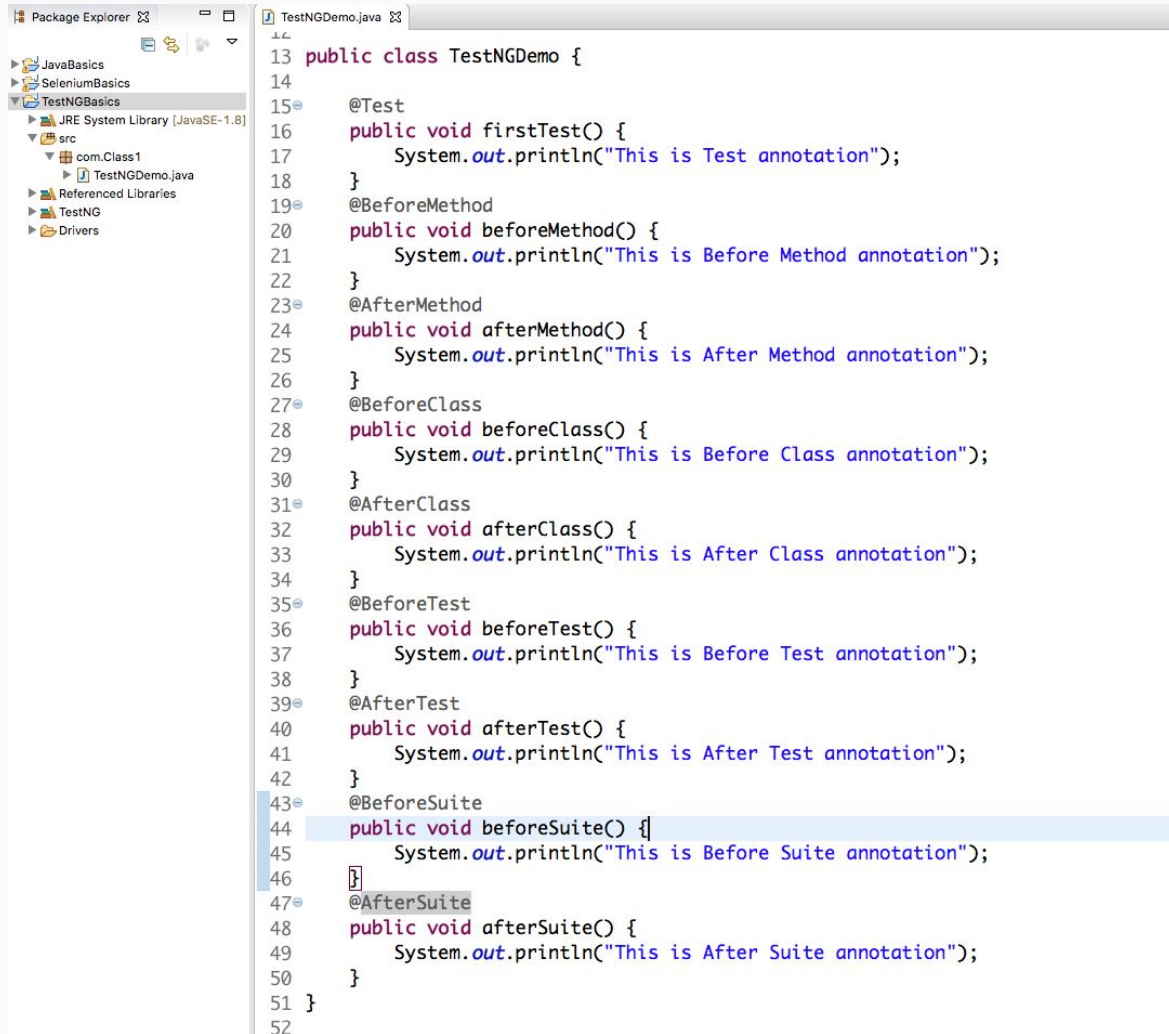
1. Create three test case with unique @test methods names.
2. Execute the test case
3. Post a screenshot of your report

Execution order of TestNG Annotations

TestNG annotations execute in a predefined order. It's very easy to find out the execution sequence of TestNG annotations:

- BeforeSuite
- BeforeTest
- BeforeClass
- BeforeGroups
- BeforeMethod
- Test
- AfterMethod
- AfterGroups
- AfterClass
- AfterTest
- AfterSuite

Execution order of TestNG Annotations



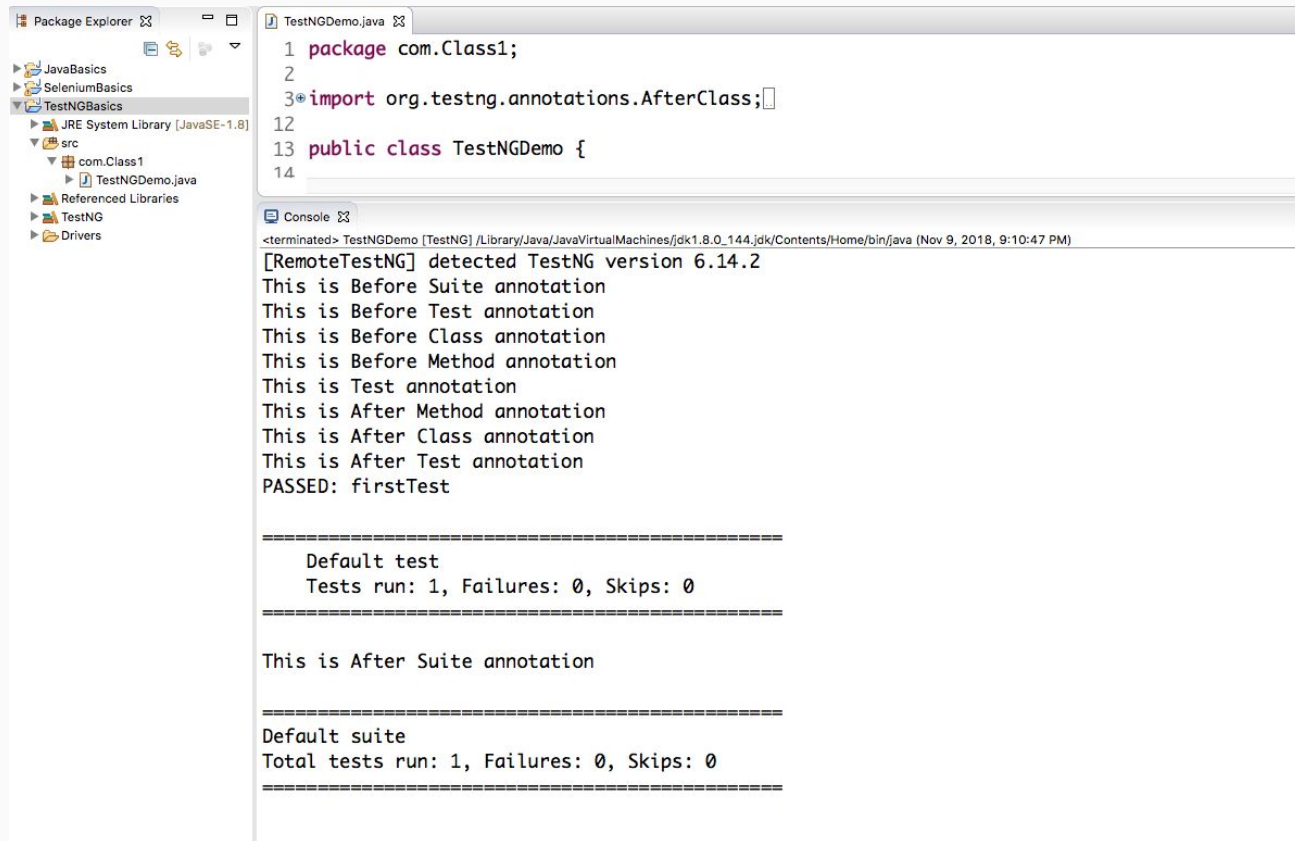
The screenshot displays the Package Explorer on the left and the TestNGDemo.java file on the right. The Package Explorer shows the following structure:

- JavaBasics
- SeleniumBasics
- TestNGBasics
 - JRE System Library [JavaSE-1.8]
 - src
 - com.Class1
 - TestNGDemo.java
 - Referenced Libraries
 - TestNG
 - Drivers

The TestNGDemo.java file contains the following code:

```
13 public class TestNGDemo {
14
15     @Test
16     public void firstTest() {
17         System.out.println("This is Test annotation");
18     }
19     @BeforeMethod
20     public void beforeMethod() {
21         System.out.println("This is Before Method annotation");
22     }
23     @AfterMethod
24     public void afterMethod() {
25         System.out.println("This is After Method annotation");
26     }
27     @BeforeClass
28     public void beforeClass() {
29         System.out.println("This is Before Class annotation");
30     }
31     @AfterClass
32     public void afterClass() {
33         System.out.println("This is After Class annotation");
34     }
35     @BeforeTest
36     public void beforeTest() {
37         System.out.println("This is Before Test annotation");
38     }
39     @AfterTest
40     public void afterTest() {
41         System.out.println("This is After Test annotation");
42     }
43     @BeforeSuite
44     public void beforeSuite() {
45         System.out.println("This is Before Suite annotation");
46     }
47     @AfterSuite
48     public void afterSuite() {
49         System.out.println("This is After Suite annotation");
50     }
51 }
52
```

Execution order of TestNG Annotations



The screenshot displays an IDE interface. On the left, the Package Explorer shows a project structure with 'TestNGDemo.java' selected. The main editor shows the source code of 'TestNGDemo.java', which includes a package declaration, an import for 'org.testng.annotations.AfterClass', and a public class 'TestNGDemo' with a 'firstTest' method. The Console window on the right shows the execution output, detailing the sequence of TestNG annotations and the final test results.

```
TestNGDemo.java
1 package com.Class1;
2
3 import org.testng.annotations.AfterClass;
12
13 public class TestNGDemo {
14
```

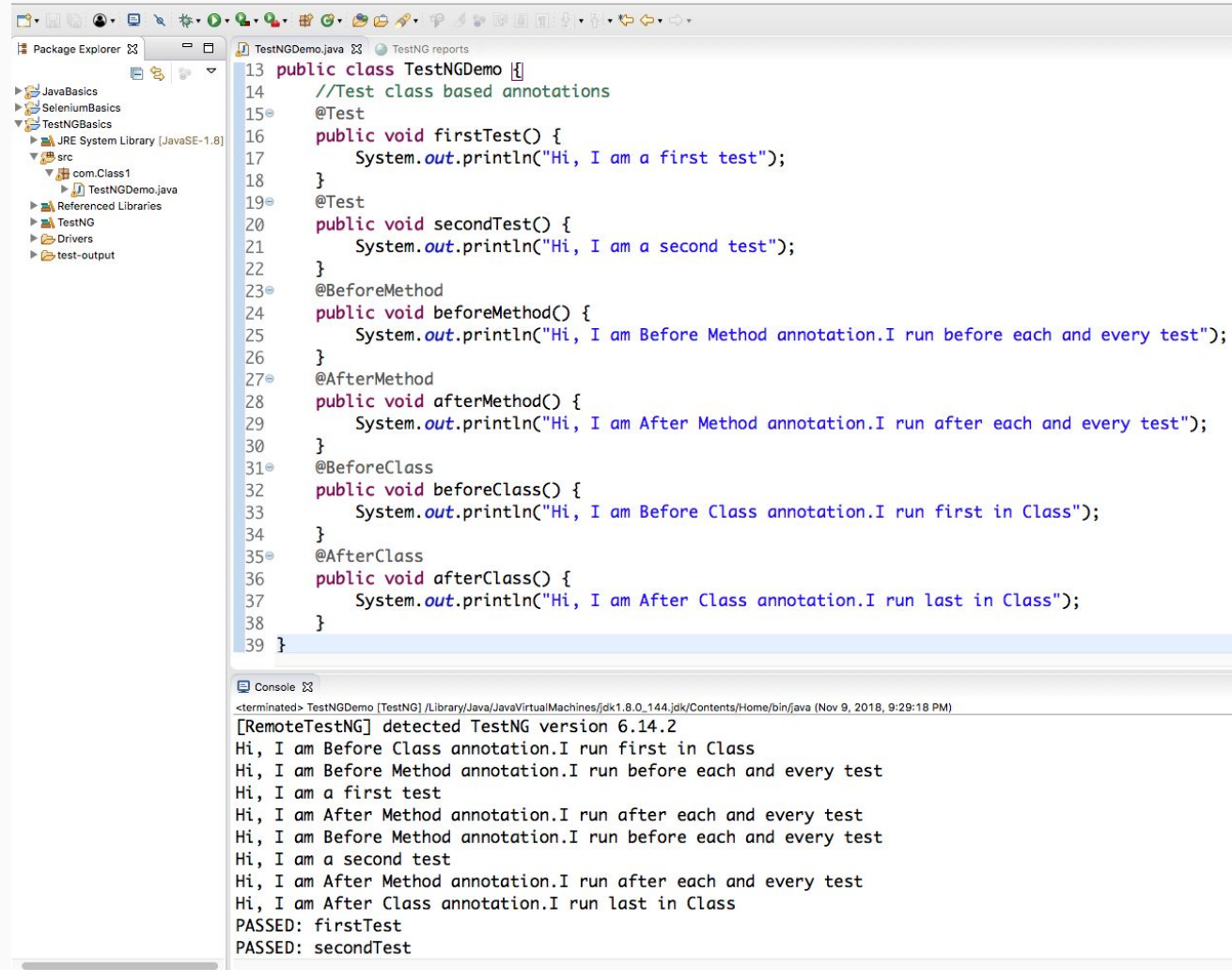
```
<terminated> TestNGDemo [TestNG] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Nov 9, 2018, 9:10:47 PM)
[RemoteTestNG] detected TestNG version 6.14.2
This is Before Suite annotation
This is Before Test annotation
This is Before Class annotation
This is Before Method annotation
This is Test annotation
This is After Method annotation
This is After Class annotation
This is After Test annotation
PASSED: firstTest

=====
Default test
Tests run: 1, Failures: 0, Skips: 0
=====

This is After Suite annotation

=====
Default suite
Total tests run: 1, Failures: 0, Skips: 0
=====
```

Execution order of TestNG Annotations



The screenshot displays an IDE with a Package Explorer on the left and a code editor on the right. The Package Explorer shows a project structure with 'TestNGDemo.java' selected. The code editor shows the following Java code:

```
13 public class TestNGDemo {
14     //Test class based annotations
15     @Test
16     public void firstTest() {
17         System.out.println("Hi, I am a first test");
18     }
19     @Test
20     public void secondTest() {
21         System.out.println("Hi, I am a second test");
22     }
23     @BeforeMethod
24     public void beforeMethod() {
25         System.out.println("Hi, I am Before Method annotation.I run before each and every test");
26     }
27     @AfterMethod
28     public void afterMethod() {
29         System.out.println("Hi, I am After Method annotation.I run after each and every test");
30     }
31     @BeforeClass
32     public void beforeClass() {
33         System.out.println("Hi, I am Before Class annotation.I run first in Class");
34     }
35     @AfterClass
36     public void afterClass() {
37         System.out.println("Hi, I am After Class annotation.I run last in Class");
38     }
39 }
```

Below the code editor, the Console window shows the execution output:

```
<terminated> TestNGDemo [TestNG] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Nov 9, 2018, 9:29:18 PM)
[RemoteTestNG] detected TestNG version 6.14.2
Hi, I am Before Class annotation.I run first in Class
Hi, I am Before Method annotation.I run before each and every test
Hi, I am a first test
Hi, I am After Method annotation.I run after each and every test
Hi, I am Before Method annotation.I run before each and every test
Hi, I am a second test
Hi, I am After Method annotation.I run after each and every test
Hi, I am After Class annotation.I run last in Class
PASSED: firstTest
PASSED: secondTest
```

Task

Create the test case for each one of the testing annotation.

Test
BeforeMethod
AfterMethod
BeforeClass
AfterClass
BeforeTest
AfterTest
BeforeSuite
AfterSuite

And Share your screenshot of your report

TestNG Report

Selenium Webdriver along with TestNG allows user to generate reports. Generating Report for test automation execution is essential part of the automation.

Execution reports play a crucial role in the formation of a sophisticated automation framework.

Reports helps you to identify the status of the test case (Pass/Fail/Skip).

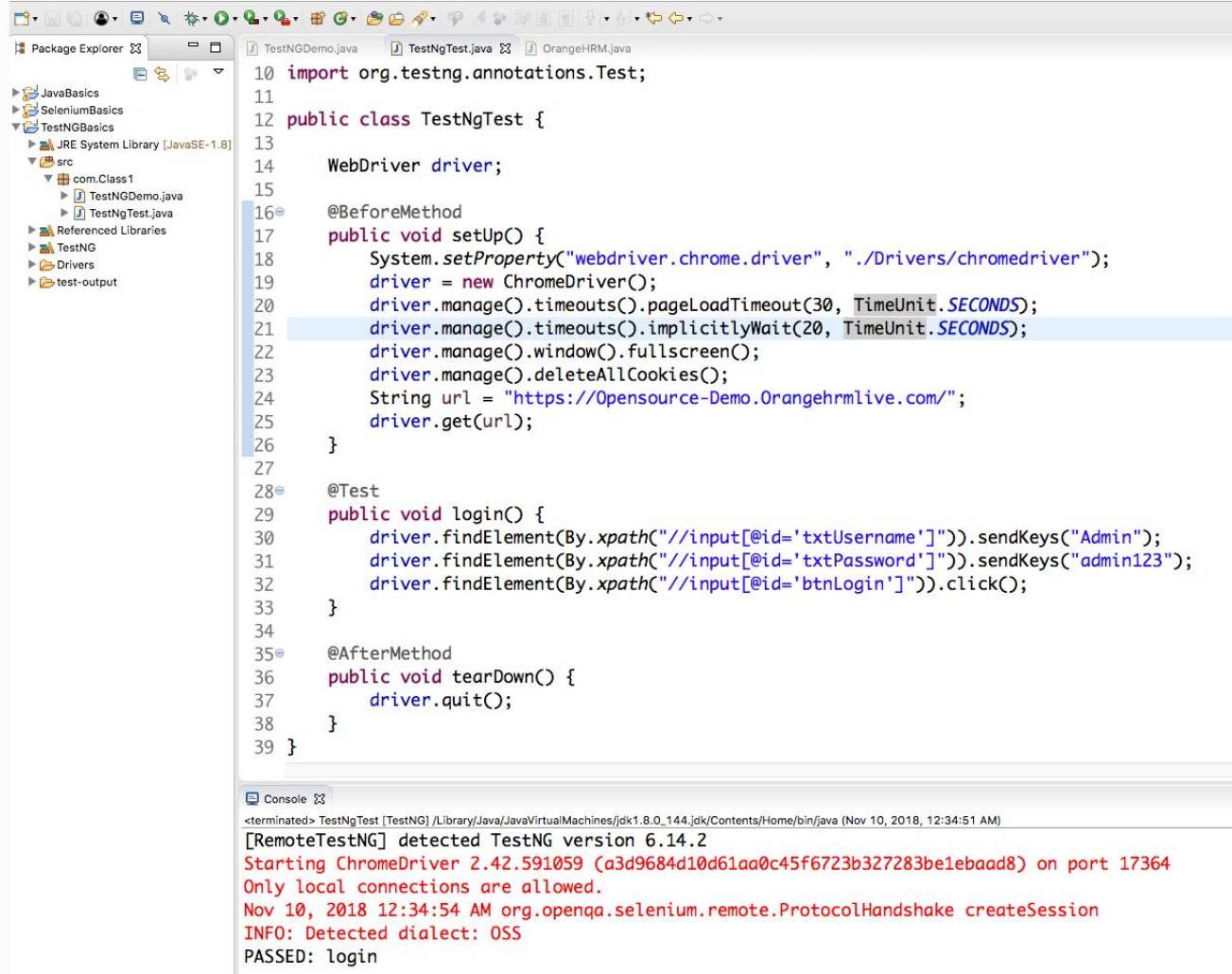
Using reports we calculate time taken by each test case.

TestNG generates the HTML report which will be stored in "test-Output" folder in our project.

Every time we execute the TestNG test class or suite this report will refresh himself and display the latest result to you.

We can share automation reports with the team and clients as well to share the status of testing progress etc.

First TestNG-WebDrive r Test Case



```
10 import org.testng.annotations.Test;
11
12 public class TestNgTest {
13
14     WebDriver driver;
15
16     @BeforeMethod
17     public void setUp() {
18         System.setProperty("webdriver.chrome.driver", "./Drivers/chromedriver");
19         driver = new ChromeDriver();
20         driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS);
21         driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
22         driver.manage().window().fullscreen();
23         driver.manage().deleteAllCookies();
24         String url = "https://Opensource-Demo.Orangehrmlive.com/";
25         driver.get(url);
26     }
27
28     @Test
29     public void login() {
30         driver.findElement(By.xpath("//input[@id='txtUsername']")).sendKeys("Admin");
31         driver.findElement(By.xpath("//input[@id='txtPassword']")).sendKeys("admin123");
32         driver.findElement(By.xpath("//input[@id='btnLogin']")).click();
33     }
34
35     @AfterMethod
36     public void tearDown() {
37         driver.quit();
38     }
39 }
```

<terminated> TestNgTest [TestNG] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Nov 10, 2018, 12:34:51 AM)

[RemoteTestNG] detected TestNG version 6.14.2

Starting ChromeDriver 2.42.591059 (a3d9684d10d61aa0c45f6723b327283be1ebaad8) on port 17364

Only local connections are allowed.

Nov 10, 2018 12:34:54 AM org.openqa.selenium.remote.ProtocolHandshake createSession

INFO: Detected dialect: OSS

PASSED: login

Task

TC 1: Swag Lab Title and Login Verification

@BeforeMethod should contain navigation to the URL and title verification

@Test should contain steps to login and “Product” word verification

@AfterMethod should logOut from the application and close the browser