Tech Mahindra

TestNG

Let's Recap

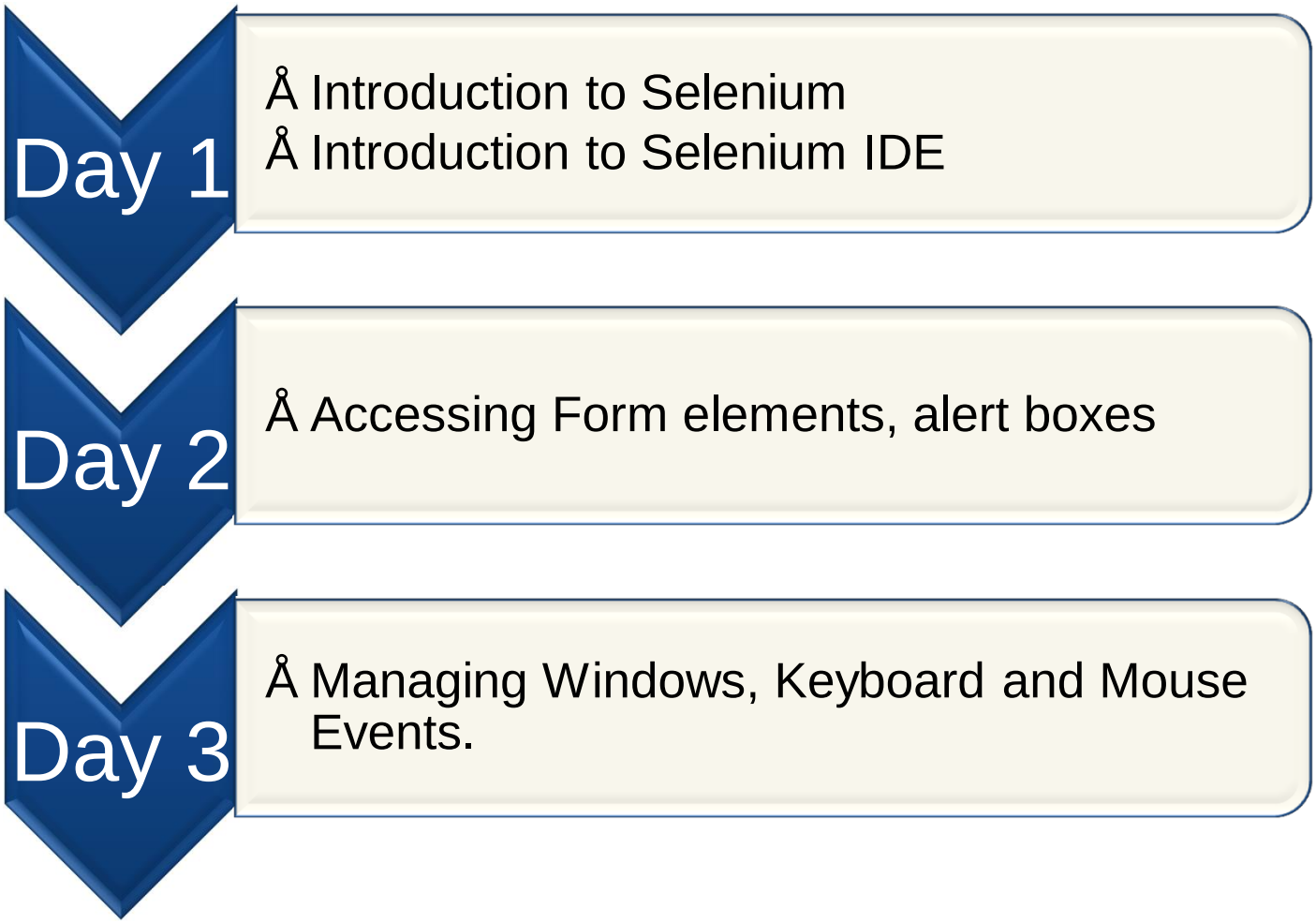
- Different methods of cssSelector and xpath to locate element.
- Different option with in the CSS selector mode for locating elements.
- Different xpath options to locate elements - Absolute abd relative path method.
- Implementations to understand the cssSelector()...and xpath()..
- Locate form elements and the actions associated with the form elements..
- Locater tree..
- Synchronization - Conditional and Unconditional..
- Implement implicit and explicit wait..
- Handling the windows...windowHnadler..
- Handling popups - Alert interface, switchTo()/alert()--WebDriver..

2

Agenda

- Introduction of TestNG
- TestNG Features
- JUnit 4 Vs TestNG
- Setup TestNG with Eclipse
- Writing Tests using TestNG
- Using Annotations
- Generating reports with Test NG

Web Driver : Concepts Summary



TestNG : Introduction

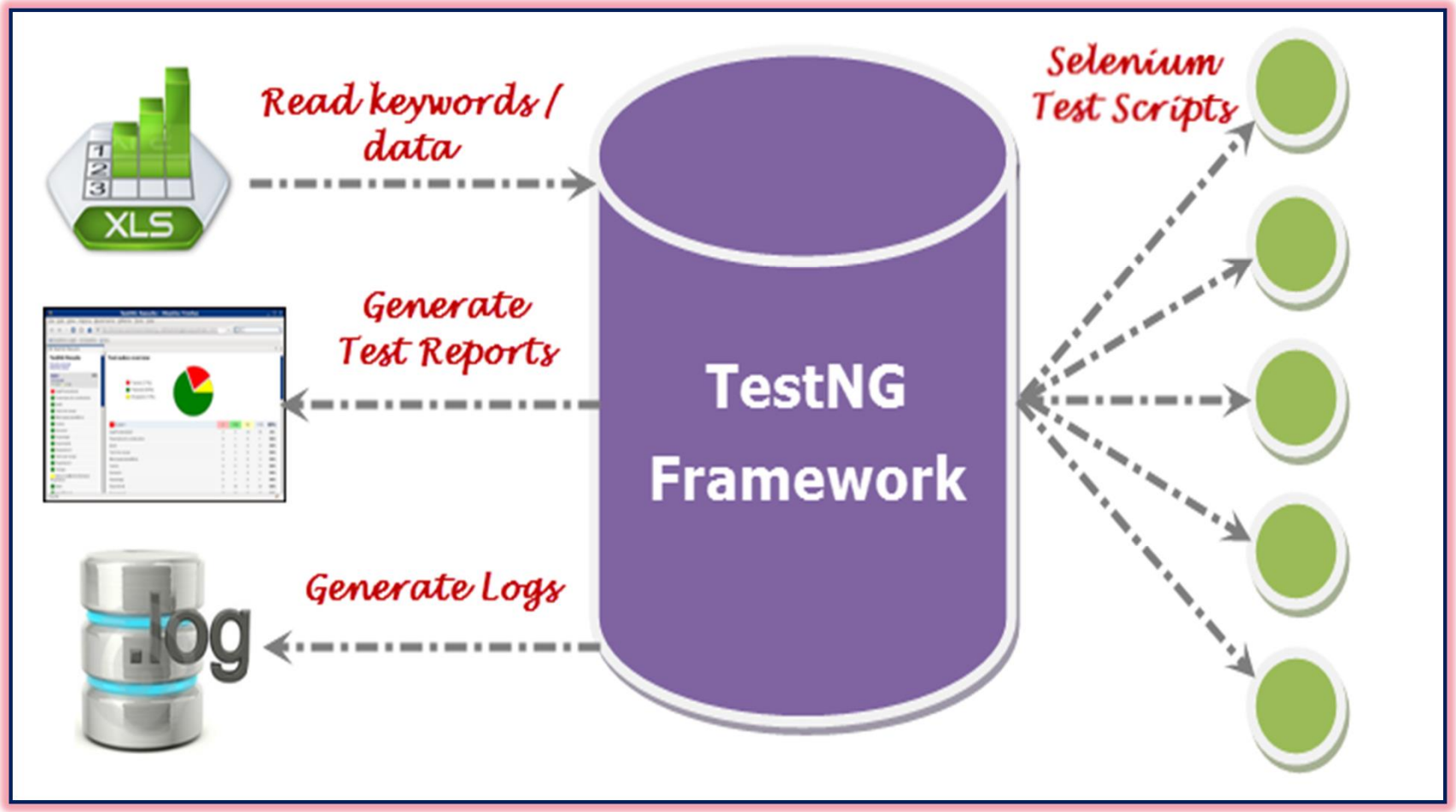
- Test **N**ext **G**eneration.
- Alternative unit testing tool for Java.
- Authored by Cédric Beust.
- First version released in 2004.
- First version has a lot of out-of-the-box features which was missing in Junit.
- *A test automation framework inspired by JUnit (in Java) and NUnit (in C#).*
- It can be used for *unit, functional, integration, and end-to-end testing*.
- It mainly uses Java annotations to configure and write test methods.

The creator of TestNG

The creator of TestNG is Cedric Beust, a well-known name in the field of Java programming, a member of the EJB 3 expert group, and creator of other popular open source projects like EJBGen and Doclipse. TestNG is distributed under the terms of Apache Software License and can be downloaded from its Web site (see Resources for a link to this and to Cedric's site).

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

TestNG : Workflow



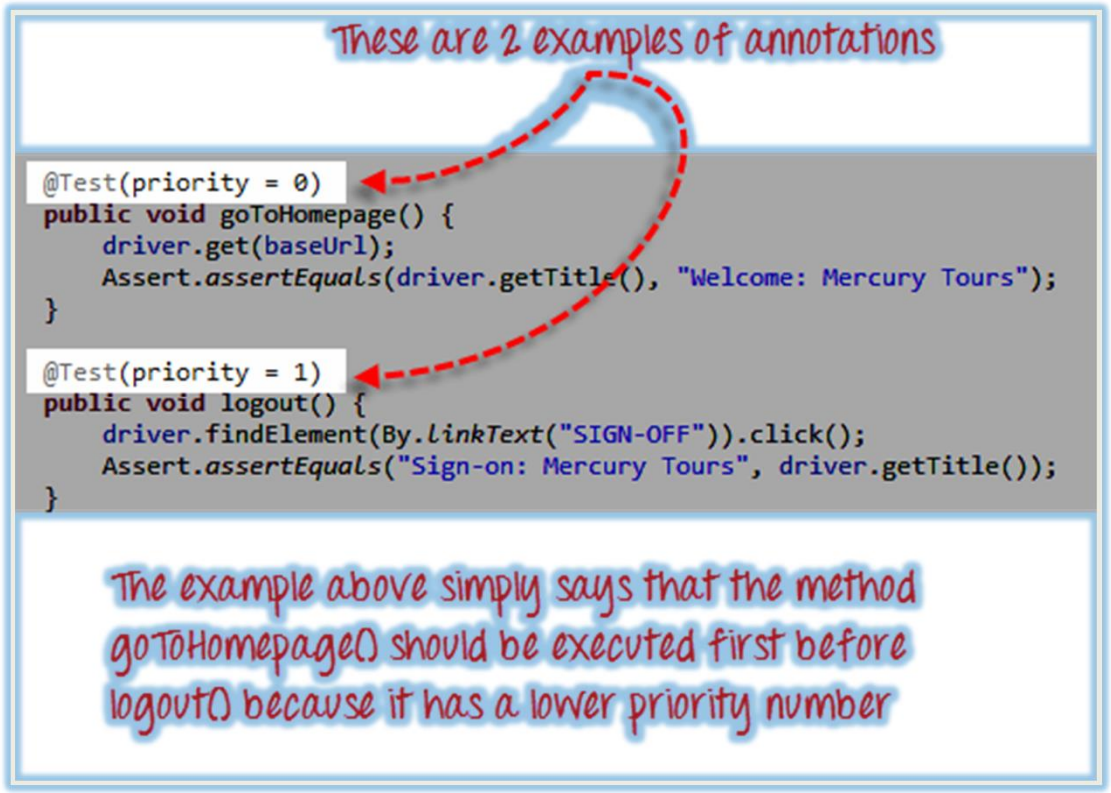
TestNG : Features/Advantages

TestNG Features/ Advantages	
	Supports Annotations.
	Flexible test configuration.
	Support for data-driven testing (with @DataProvider).
	Support for parameters.
	Powerful execution model (no more Test Suite).
	Supported by a variety of tools and plug-ins (Eclipse, IDEA, Maven etc..).
	Dependent methods for application server testing.
	Run tests in arbitrarily big thread pools with various policies available.

TestNG : Advantages Over JUnit

- There are three major advantages of TestNG over JUnit:
  - *Annotations are easier to understand.*
  - *Test cases can be grouped more easily.*
  - *Parallel testing is possible.*

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

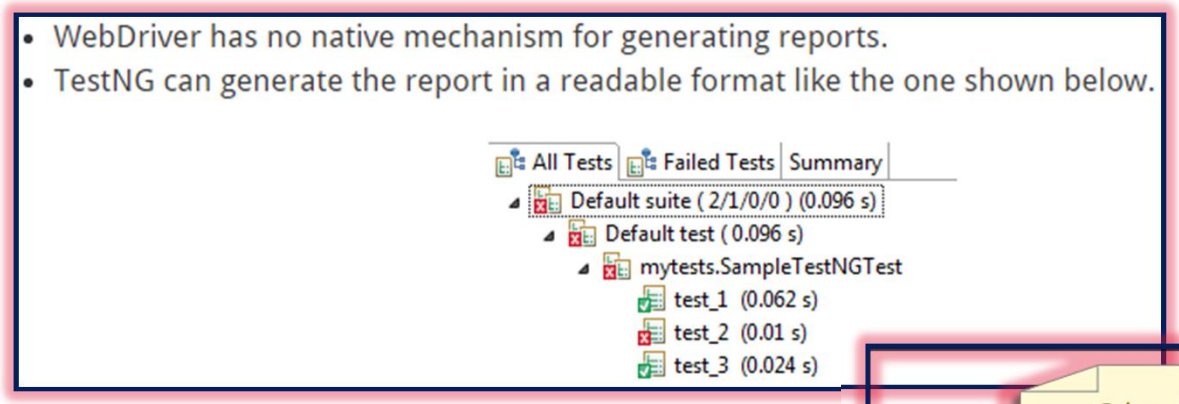


TestNG : Advantages Over JUnit

- TestNG supports group concept, while JUnit does not support this.
- Parameter Testing is possible in TestNG, but it cannot be done in JUnit.
- TestNG provides special annotations, such as @Before/AfterSuite and @Before/AfterTest.
- In TestNG, it is possible to change the method name. JUnit, however, does not allow this.
- TestNG supports the ódependenceômethod. This is not available in JUnit.

TestNG : Why ?

- *TestNG can generate reports based on our Selenium test results.*



- *TestNG simplifies the way the tests are coded.*

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

TestNG structure
(easier to understand)

```
public class SampleTestNGTest {
    public String baseUrl = "http://newtours.demoaut.com/";
    public WebDriver driver;

    @BeforeTest
    public void setBaseURL() {
        driver = new FirefoxDriver();
        driver.get(baseUrl);
    }

    @Test
    public void verifyHomepageTitle() {
        String expectedTitle = "Welcome: Mercury Tours";
        String actualTitle = driver.getTitle();
        Assert.assertEquals(actualTitle, expectedTitle);
    }


    @AfterTest
    public void endSession() {
        driver.quit();
    }
}
```

JUnit 4 Vs TestNG : Comparison

- The main annotation differences between JUnit4 and TestNG are :
  - In JUnit 4, we have to declare ***"@BeforeClass"*** and ***"@AfterClass"*** method as static method.
  - TestNG is more flexible in method declaration, it does not have this constraints.
  - 3 additional setUp/tearDown level : suite and group ***(@Before/AfterSuite***, ***@Before/AfterTest, @Before/AfterGroup***).

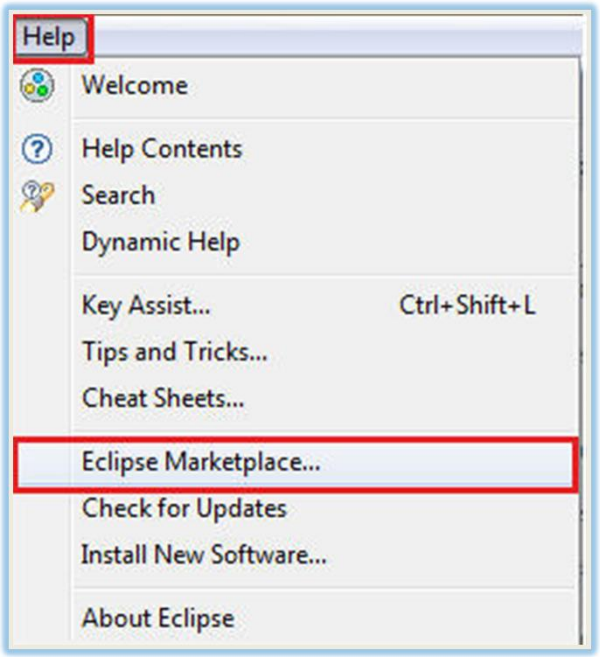
Functionality - JUnit 4 vs TestNG

	Annotation Support	Exception Test	Ignore Test	Timeout Test	Suite Test	Group Test	Parameterized (primitive value)	Parameterized (object)	Dependency Test
TestNG	✓	✓	✓	✓	✓	✓	✓	✓	✓
JUnit 4	✓	✓	✓	✓	✓	✗	✓	✗	✗

JUnit 4 Vs TestNG : Features Comparison

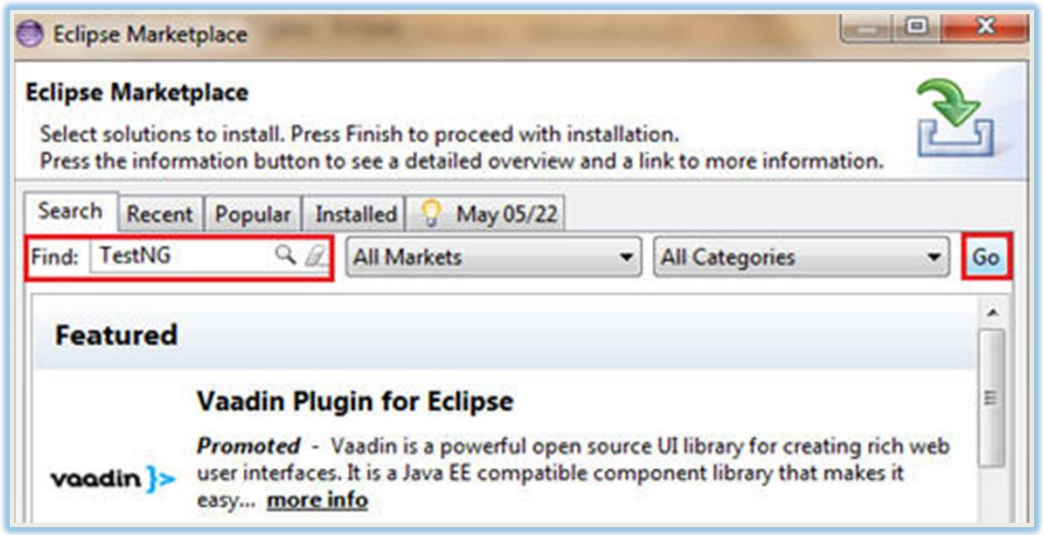
FEATURE	JUNIT	TESTNG
Purpose	General unit testing	Focus on Integration testing for Enterprise projects
IDE support	yes	Yes
Maven support	yes	Yes
setup/teardown for test	@Before / @After	@BeforeMethod / @AfterMethod
setup/teardown for class	@Before / @After	@BeforeClass / @AfterClass
setup/teardown for suite	no	@BeforeSuite / @AfterSuite
setup/teardown for test groups	no	@BeforeGroups/ @AfterGroups
setup/teardown for test	in annotations	In annotations and/or XML file
Parameterised tests	Yes, but in a limited way	Yes
Test groups	Yes with Categories (new feature)	Yes
Test for Exceptions	Yes	Yes
Timeouts in tests	Yes	Yes
Test order	Non-Deterministic or alphabetical	Can be defined in detail with dependencies
Dynamic test input	No	Yes with DataProviders
Can run tests of the other library	No	Yes, TestNG can run JUnit tests
Assumptions before running a test	Yes	No
Dependency injection for tests	No	Yes, with Google Guice
Ignore/disable test	Yes	Yes
Parallel testing	No	Yes
Test listeners	No	Yes
Test reporters	No	Yes

TestNG : Configuring with Eclipse

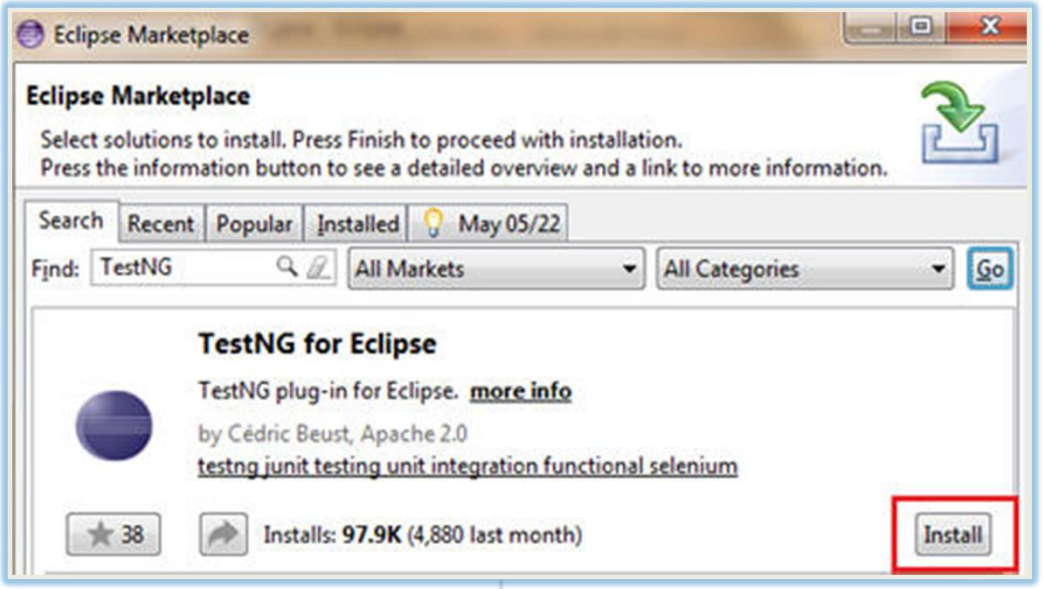


1

2

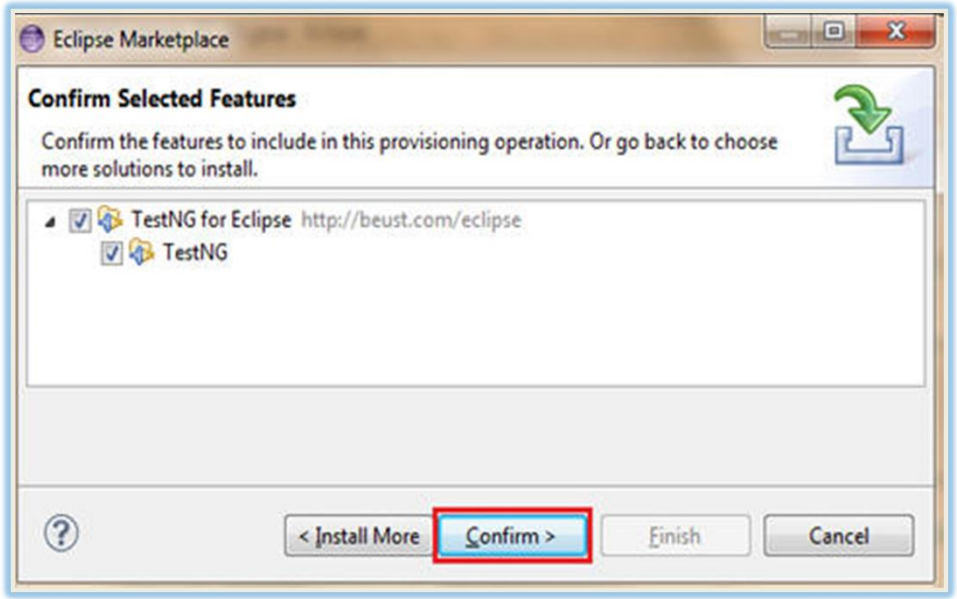


3

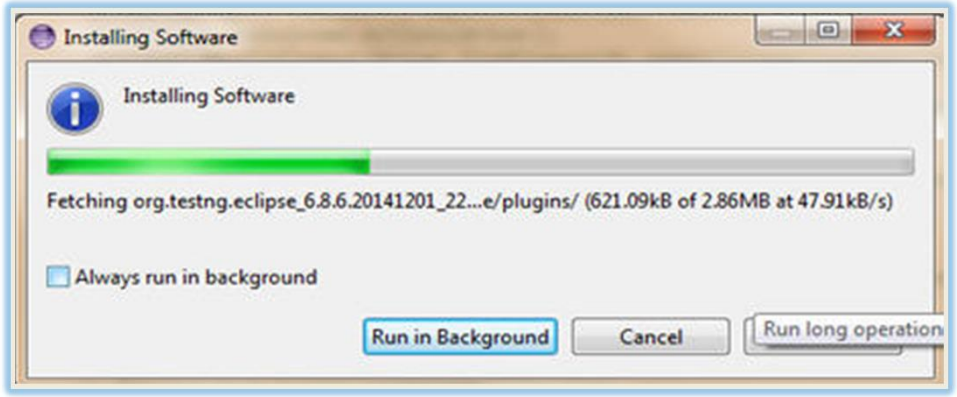


TestNG : Configuring with Eclipse Cont...

4

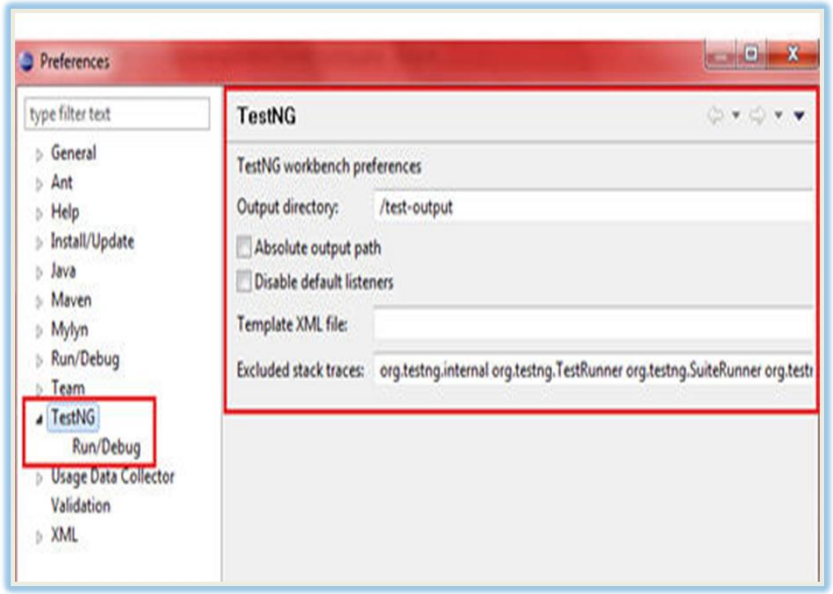


5



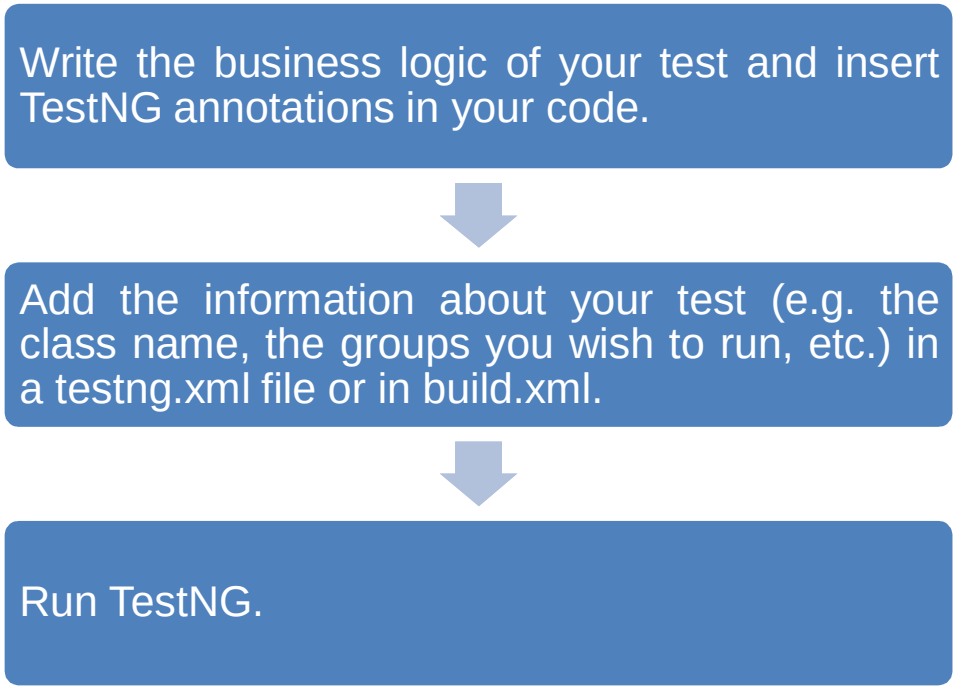
Upon restart, user can verify the TestNG installation by navigating to "Preferences" from "Window" option in the menu bar.

6



TestNG : Quick Start

- A TestNG test class is a plain old Java object; you don't need to extend any special class or use any naming convention for test methods:
- You just use the annotation ***@Test*** to signal to the framework the methods of the class that are tests.
- Writing a test in TestNG basically involves the following steps:

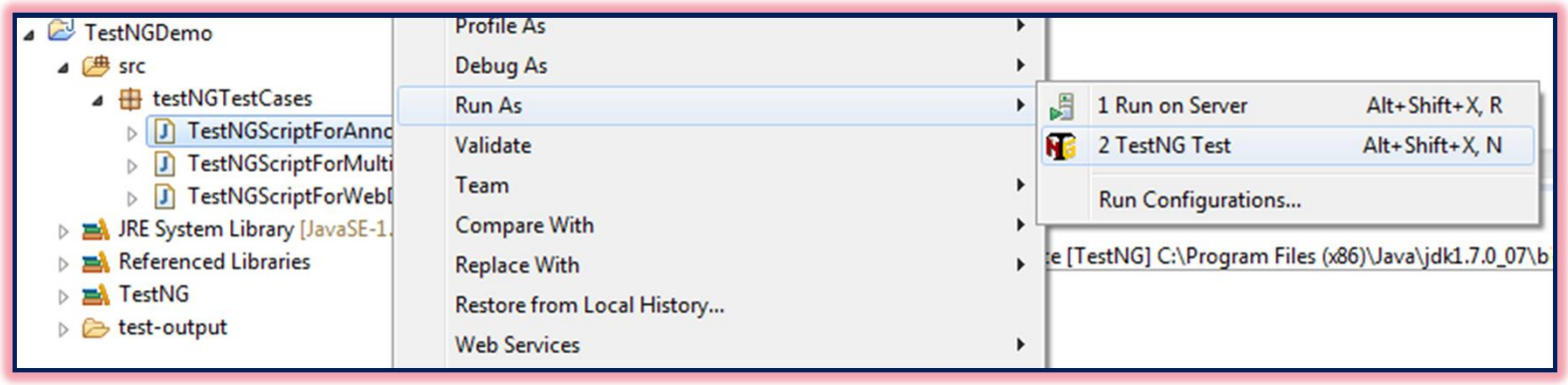


TestNG : Quick Start

```
package firsttestngpackage;

import org.openqa.selenium.*;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.Assert;
import org.testng.annotations.*;

public class FirstTestNGFile {
    public String baseUrl = "http://newtours.demoaut.com/";
    public WebDriver driver = new FirefoxDriver();

    @Test
    public void verifyHomepageTitle() {
        driver.get(baseUrl);
        String expectedTitle = "Welcome: Mercury Tours";
        String actualTitle = driver.getTitle();
        Assert.assertEquals(actualTitle, expectedTitle);
        driver.quit();
    }
}
```

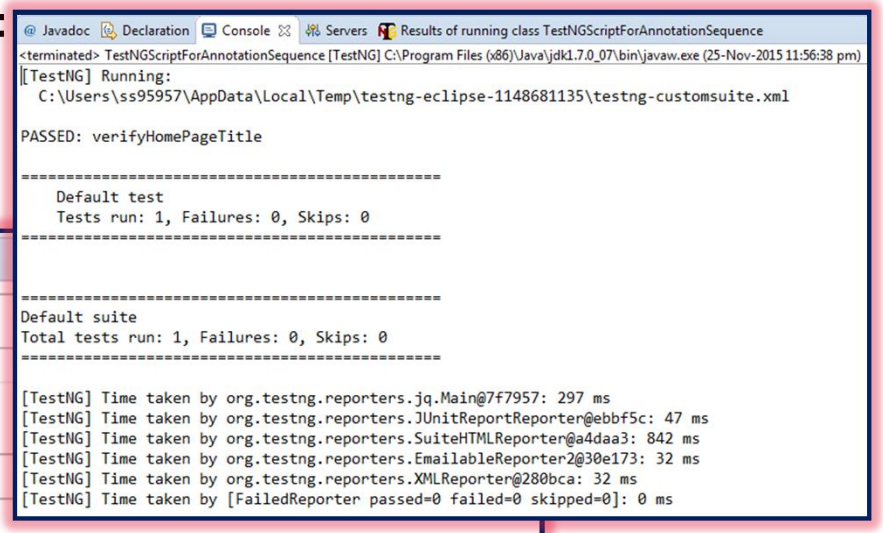
- TestNG does not require you to have a main() method.
- Methods need not be static.
- We used the @Test annotation.
- *@Test is used to tell that the method under it is a test case.*
- We used the Assert class. *The Assert class is used to conduct verification operations in TestNG*.
- To use it, we need to import the org.testng.Assert package.

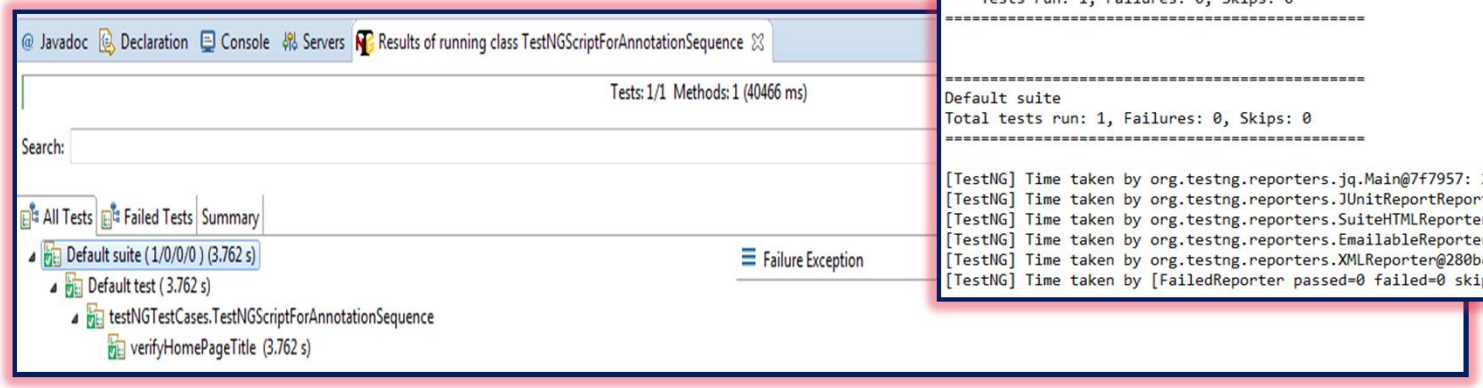
TestNG : Executing Script



TestNG result is displayed into two windows:

- Console Window.
- TestNG Result Window.

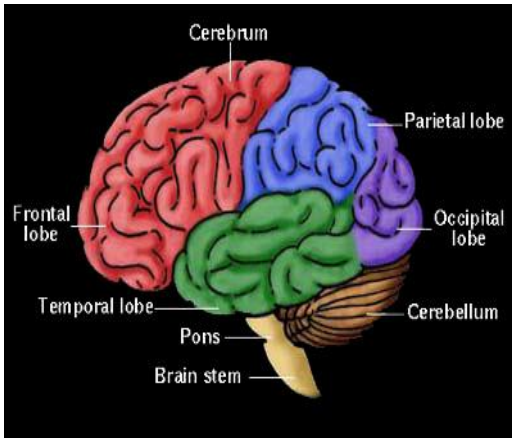




TestNG : testng.xml

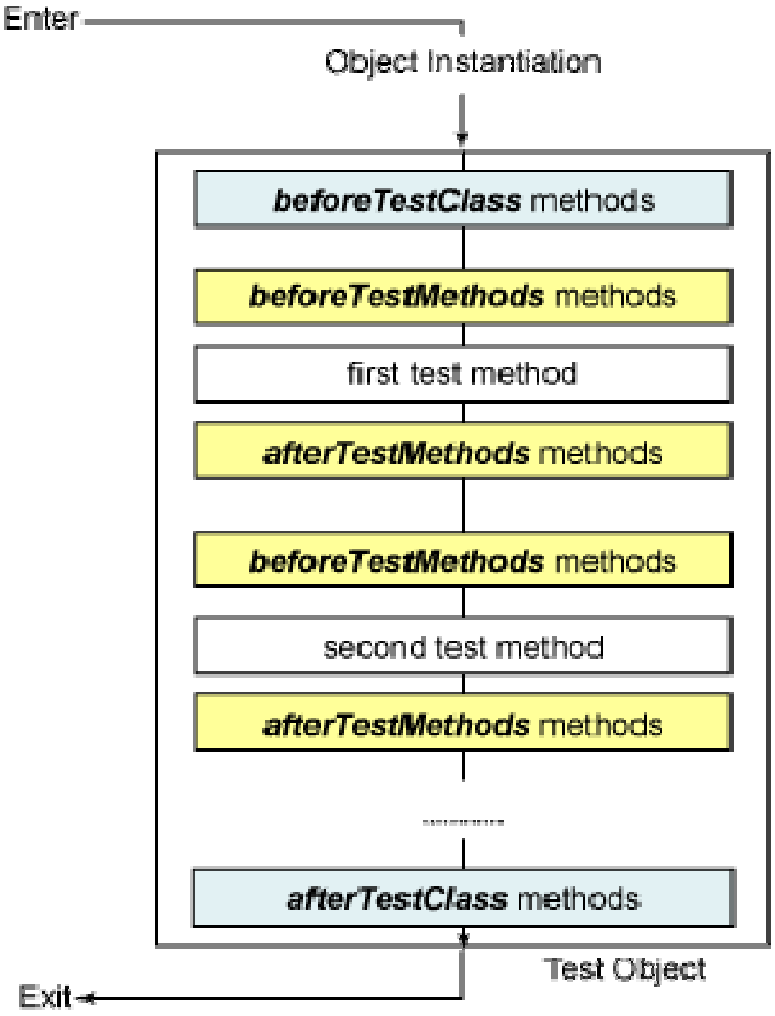
Where all the runtime information goes:

- The test methods, classes, packages
- Which groups should be run (include-groups)
- Which groups should not be run (exclude-groups)
- Define additional groups (groups of groups)
- Whether the tests should be run in parallel
- Parameters
- JUnit mode



Note: testng.xml is optional (can also use ant, command line)

TestNG : Test case Life Cycle



beforeTestClass methods are executed after the instantiation of the class but before any test method has been run.

afterTestClass methods are executed after every test method in a class has been run.

beforeTestMethod methods are executed before the execution of any test method in a class.

afterTestMethod methods are executed after the execution of every test method in a class.

TestNG Annotations

- With using TestNG as a testing framework, there is no need to use ñmainòmethod. The execution will be handled by the TestNG (testing framework).
- **@Test is used to tell that the method under it is a test case**.
- *TestNG identifies the methods it is interested in, by looking up annotations*. Hence, method names are not restricted to any pattern or format.
- We can pass additional parameters to annotations.
- Annotations are strongly typed, so the compiler will flag any mistakes right away.
- Test classes no longer need to extend anything (such as TestCase, for JUnit 3).
- **The placement of the annotation blocks can be interchanged without affecting the chronological order by which they will be executed**.

TestNG Annotations contd..

```
public class FirstTestNGFile {
    public String baseUrl = "http://newtours.demoaut.com/";
    public WebDriver driver = new FirefoxDriver();

    @AfterTest
    public void terminateBrowser() {
        driver.quit();
    }

    @BeforeTest
    public void launchBrowser() {
        driver.get(baseUrl);
    }

    @Test
    public void verifyHomepageTitle() {
        String expectedTitle = "Welcome: Mercury Tours";
        String actualTitle = driver.getTitle();
        Assert.assertEquals(actualTitle, expectedTitle);
    }
}
```

Jumbled

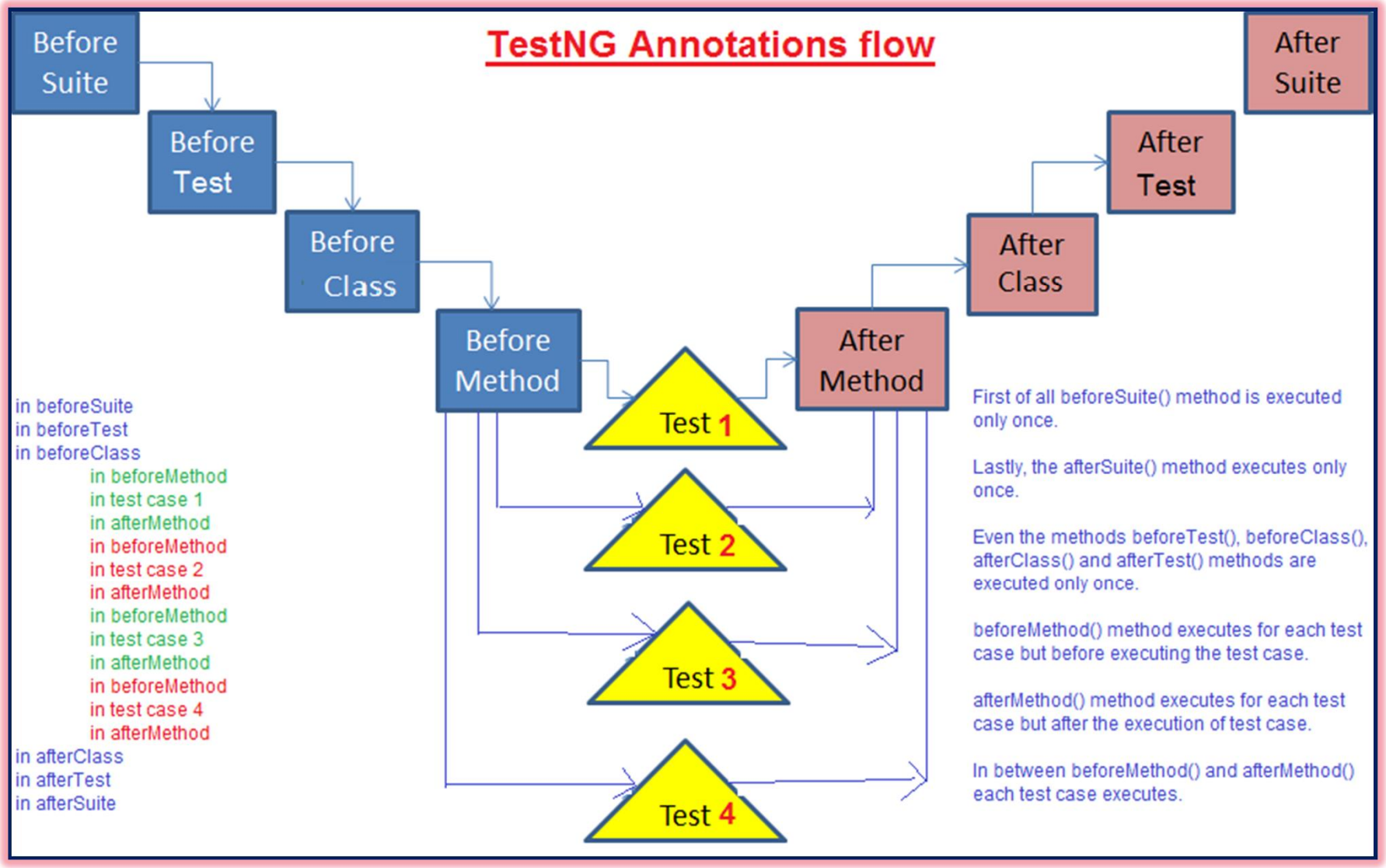
Methods in chronological order

firsttestngpackage.FirstTestNGFile

launchBrowser	0 ms
verifyHomepageTitle	4641 ms
terminateBrowser	4664 ms

TestNG : Annotations

TestNG : Annotations Flow



TestNG : Annotations Hierarchy

- It says that @Test is the smallest annotation.
- @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.

```
<suite>
        <test>
                <classes>
                        <method>
                                <test>
                        </method>
                </classes>
        </test>
</suite>
```

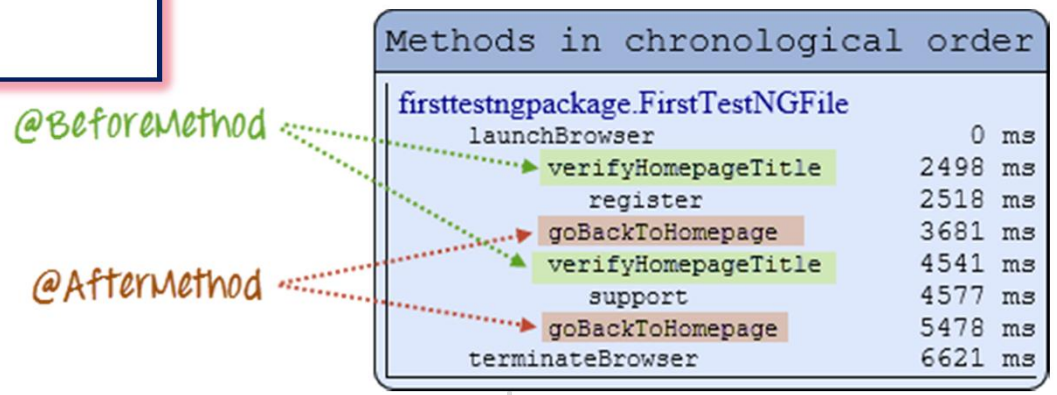
TestNG : Annotations Support

Feature	JUnit 4	TestNG
test annotation	@Test	@Test
run before all tests in this suite have run	—	@BeforeSuite
run after all tests in this suite have run	—	@AfterSuite
run before the test	—	@BeforeTest
run after the test	—	@AfterTest
run before the first test method that belongs to any of these groups is invoked	—	@BeforeGroups
run after the last test method that belongs to any of these groups is invoked	—	@AfterGroups
run before the first test method in the current class is invoked	@BeforeClass	@BeforeClass
run after all the test methods in the current class have been run	@AfterClass	@AfterClass
run before each test method	@Before	@BeforeMethod
run after each test method	@After	@AfterMethod
ignore test	@ignore	@Test(enbale=false)
expected exception	@Test(expected = ArithmeticException.class)	@Test(expectedExceptions = ArithmeticException.class)
timeout	@Test(timeout = 1000)	@Test(timeout = 1000)

TestNG : Annotations Sample

```
public class FirstTestNGFile {

    public String baseUrl = "http://newtours.demoaut.com/";
    public WebDriver driver = new FirefoxDriver();
    public String expected = null;
    public String actual = null;

    @BeforeTest
    public void launchBrowser() {
        driver.get(baseUrl);
    }

    @BeforeMethod
    public void verifyHomepageTitle() {
        expected = "Welcome: Mercury Tours";
        actual = driver.getTitle();
        Assert.assertEquals(actual, expected);
    }

    @Test(priority = 0)
    public void register(){
        driver.findElement(By.linkText("REGISTER")).click();
        expected = "Register: Mercury Tours";
        actual = driver.getTitle();
        Assert.assertEquals(actual, expected);
    }
```

```
    @Test(priority = 1)
    public void support() {
        driver.findElement(By.linkText("SUPPORT")).click();
        expected = "Under Construction: Mercury Tours";
        actual = driver.getTitle();
        Assert.assertEquals(actual, expected);
    }

    @AfterMethod
    public void goBackToHomepage() {
        driver.findElement(By.linkText("Home")).click();
    }

    @AfterTest
    public void terminateBrowser() {
        driver.quit();
    }
}
```



TestNG : Annotations Summary

@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.

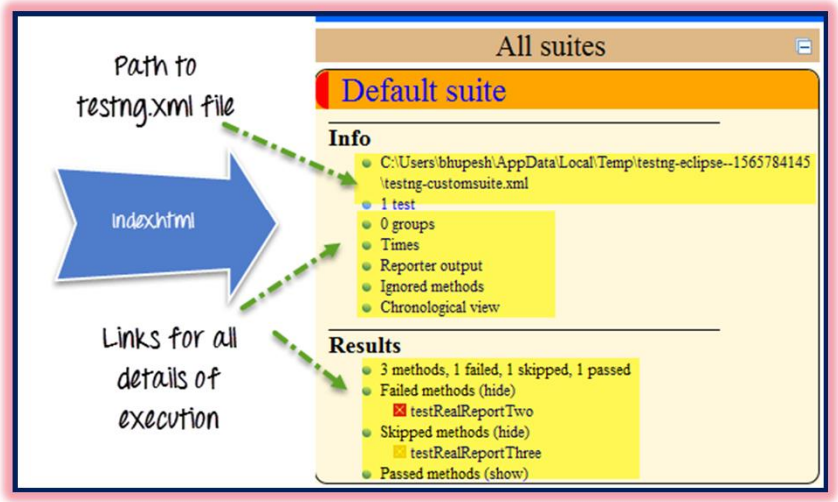
@BeforeMethod: The annotated method will be run before each test method.

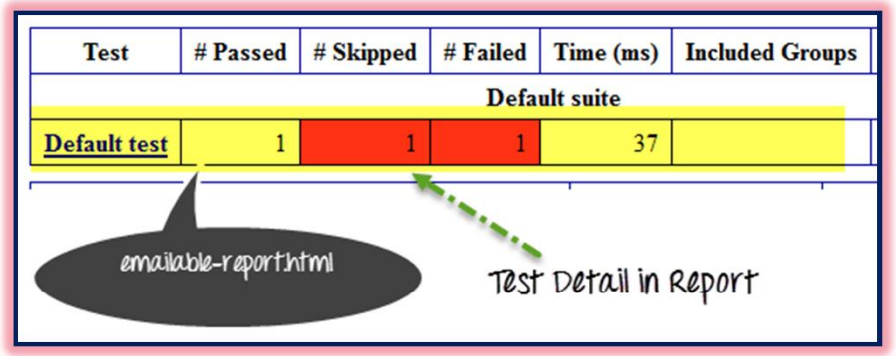
@AfterMethod: The annotated method will be run after each test method.

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

TestNG : Reporting

- TestNG library provides a very handy reporting feature.
- After execution, TestNG 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 error, groups, time ,reporter logs, testng xml files.
  - **emailable-report.html:** This is the summarize report of the current test execution which contains test case message in green (for pass test cases) and red(for failed test cases) highlight.





- TestNG is an advance framework designed in a way to leverage the benefits by both the developers and testers.
- TestNG is an open source framework which is distributed under the Apache software License and is readily available for download.
- TestNG is freely available and can be easily installed in the Eclipse IDE using Eclipse Market.
- @Test is one of the annotations provided by TestNG. This annotation lets the program execution to know that method annotated as @Test is a test method.
- To be able to use different TestNG annotations, we need to import the package ñ**import** org.testng.annotations.*ò
- There is no need of main() method while creating test scripts using TestNG.
- To be able to use different assertions, we are required to import ñ**import** org.testng.Assertò

Thank you

Saradhi.Seshagiri@TechMahindra.com

Disclaimer

Tech Mahindra Limited, herein referred to as TechM provide a wide array of presentations and reports, with the contributions of various professionals. These presentations and reports are for informational purposes and private circulation only and do not constitute an offer to buy or sell any securities mentioned therein. They do not purport to be a complete description of the markets conditions or developments referred to in the material. While utmost care has been taken in preparing the above, we claim no responsibility for their accuracy. We shall not be liable for any direct or indirect losses arising from the use thereof and the viewers are requested to use the information contained herein at their own risk. These presentations and reports should not be reproduced, re-circulated, published in any media, website or otherwise, in any form or manner, in part or as a whole, without the express consent in writing of TechM or its subsidiaries. Any unauthorized use, disclosure or public dissemination of information contained herein is prohibited. Unless specifically noted, TechM is not responsible for the content of these presentations and/or the opinions of the presenters. Individual situations and local practices and standards may vary, so viewers and others utilizing information contained within a presentation are free to adopt differing standards and approaches as they see fit. You may not repackage or sell the presentation. Products and names mentioned in materials or presentations are the property of their respective owners and the mention of them does not constitute an endorsement by TechM. Information contained in a presentation hosted or promoted by TechM is provided "as is" without warranty of any kind, either expressed or implied, including any warranty of merchantability or fitness for a particular purpose. TechM assumes no liability or responsibility for the contents of a presentation or the opinions expressed by the presenters. All expressions of opinion are subject to change without notice.

Version History

Version History				
Version No	**Date**	**Created/ Changed by**	**Changes made**	**Reviewed by**
1	27-Jan-16	Saradhi Seshagiri	- Conversion to 2016 template - Adding the new contents	Prakash Goteti

Tech Mahindra