Perform Testing on Your Application with TestNG

A Day in the Life of a Full Stack Developer

Joe, the Full Stack Developer for Abq Inc., has been working hard. He has delivered most of the functions of a fully dynamic web application. They were also approved by the Quality Assurance team. However, the company is still facing bandwidth issues in basic automation testing due to a resource crunch.

During sprint planning, Joe has been assigned to perform high-level testing of the features that are incorporated in the application with the help of TestNG and enhance it with Jenkins.

In this lesson, we will learn how to solve this real-world scenario and help Joe effectively complete his task.

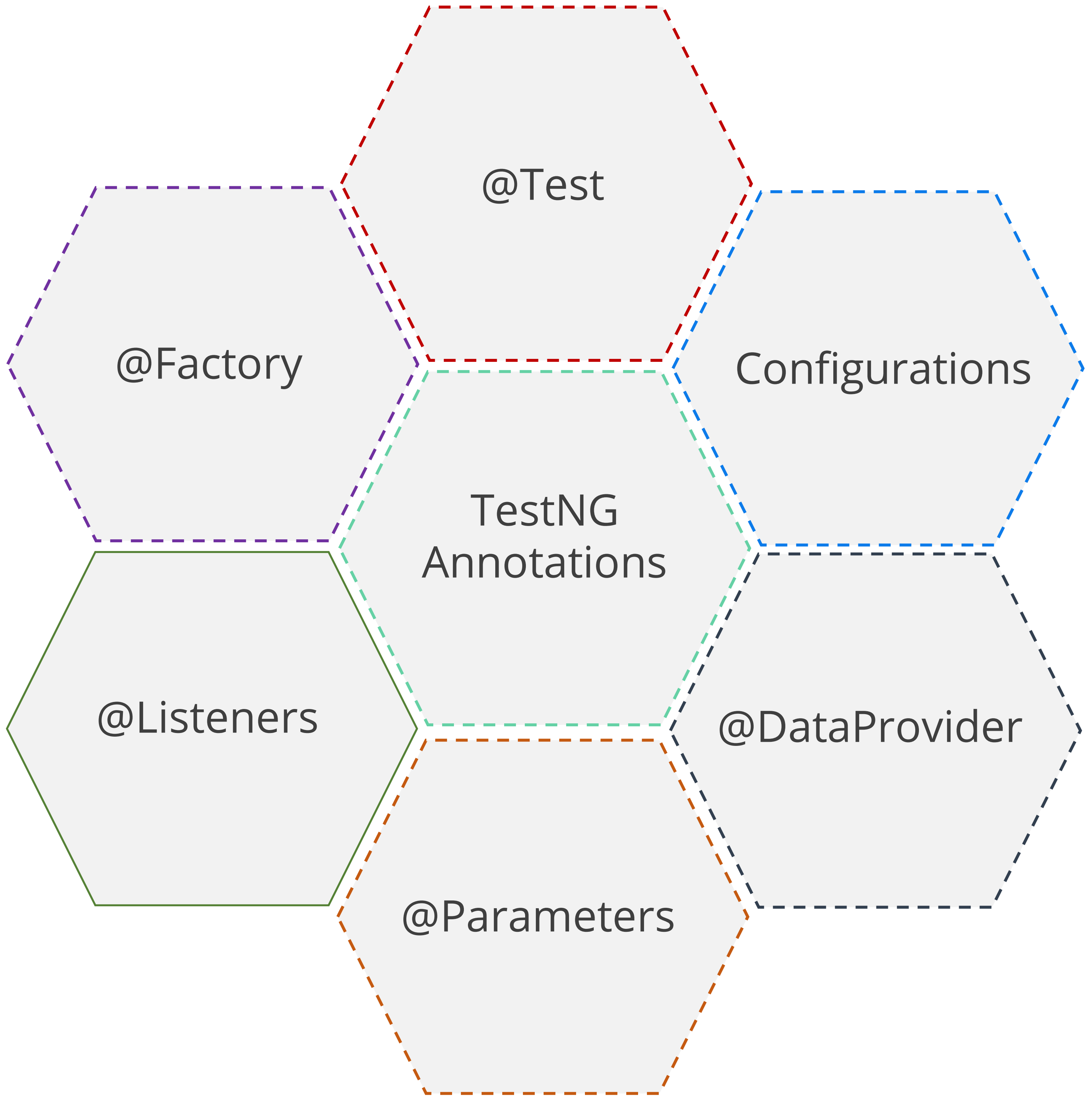
Learning Objectives

By the end of this lesson, you will be able to:

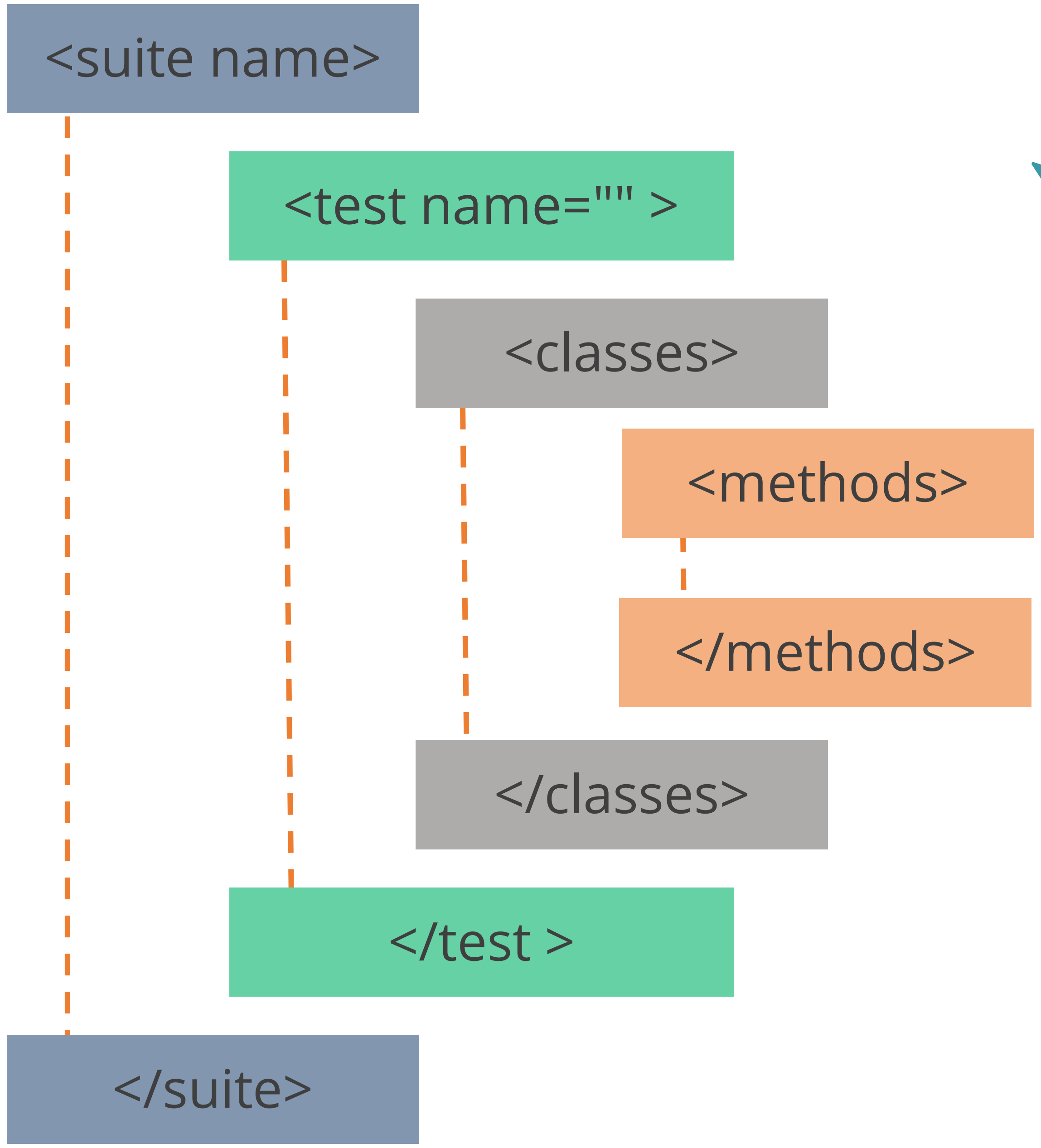
- Write test cases for applications using TestNG
- Integrate Jenkins with Selenium
- Fetch test reports in web page and spreadsheet

Testing with Selenium Using TestNG

Various Annotations in TestNG



TestNG Configuration File

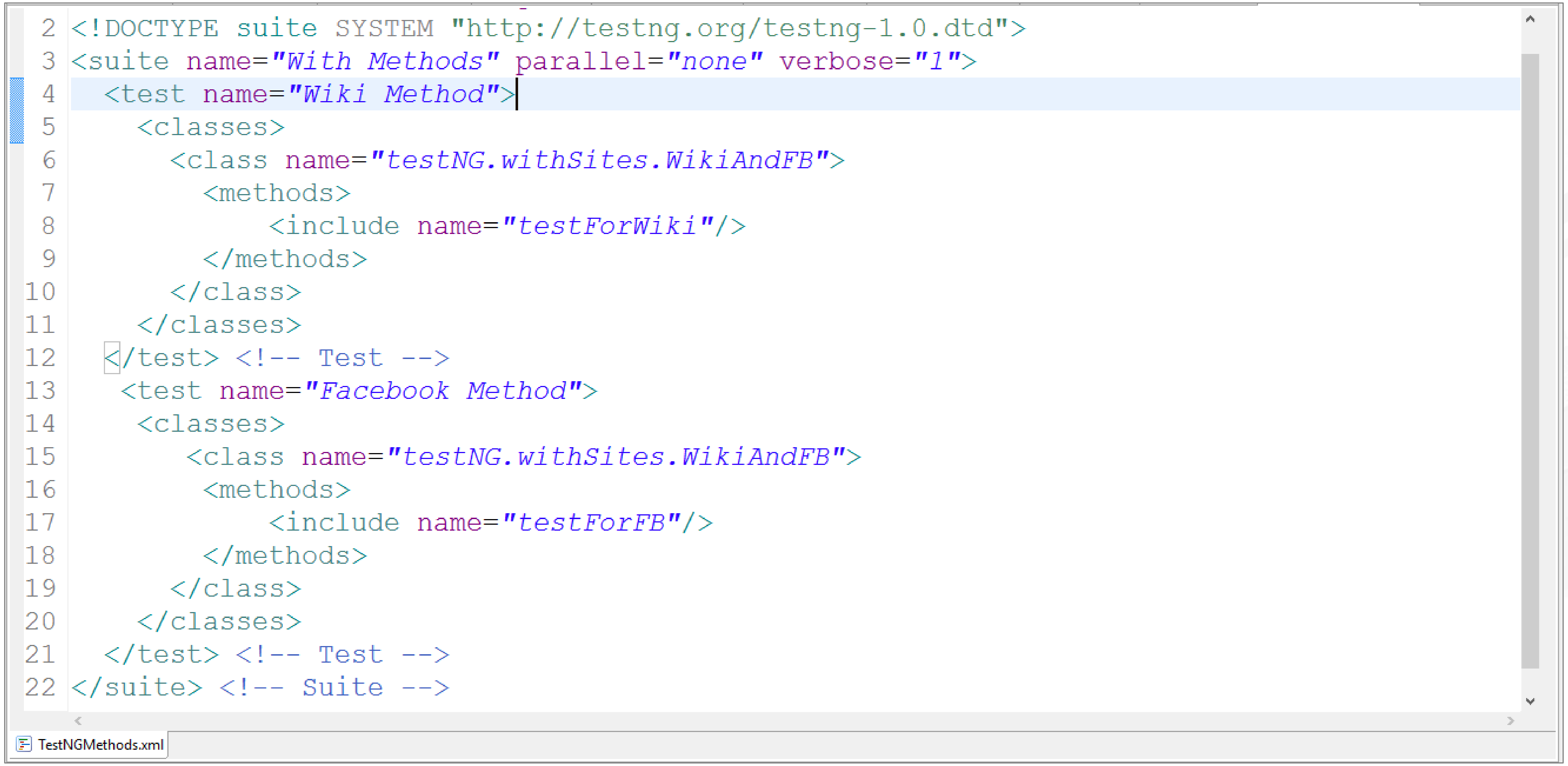


The **testng.xml** contains the following structure:

- The first element present in **testng.xml** called suite; it contains one or more test elements
- A test element is made up of one or more classes
- A class contains one or more test methods

TestNG Configuration File

Sample **testng.xml:**



```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="With Methods" parallel="none" verbose="1">
  <test name="Wiki Method">
    <classes>
      <class name="testNG.withSites.WikiAndFB">
        <methods>
            <include name="testForWiki"/>
        </methods>
      </class>
    </classes>
  </test> <!-- Test -->
   <test name="Facebook Method">
    <classes>
       <class name="testNG.withSites.WikiAndFB">
        <methods>
            <include name="testForFB"/>
        </methods>
      </class>
    </classes>
  </test> <!-- Test -->
</suite> <!-- Suite -->
```

Configuration Annotations

Annotation	Description
@BeforeSuite	Runs before suite execution starts
@AfterSuite	Runs after all tests are executed
@BeforeTest	Runs before first test in a <test> tag is executed
@AfterTest	Runs after all the tests in a <test> tag are executed
@BeforeGroups	Runs before first test method of the group(s) is executed
@AfterGroups	Runs after all the test methods of the group(s) are executed
@BeforeClass	Runs before first test in a class is executed
@AfterClass	Runs after all the tests in a class are executed
@BeforeMethod	Runs before each test in a class is executed
@AfterMethod	Runs after each test is executed

Perform All Test Annotations

Duration: 15 min.

Problem Statement:
Using Selenium TestNG, write a program to run all the test cases and understand the flow of configuration annotations on the basis of their occurrence.

Assisted Practice: Guidelines

Steps to perform all test annotations:

1. Open Eclipse
2. Create a Java project and name it TestAnnotations
3. Create a TestNG class file
4. Write the code for all configuration annotations
5. Run the program as a TestNG project
6. Push the code to GitHub repositories

FULL STACK

Working with Attributes of @Test

simplilearn

Configuration Annotations:Attributes

Attributes	Description
alwaysRun – true/false	If set to true, method(s) will run regardless of groups or failure of dependent groups or methods
groups	List of groups that the method belongs to
dependsOnGroups	Declares the **groups** on which the method depends
dependsOnMethods	Declares another method on which the method depends
Enabled – true/false	Default is true and if set to false, method will not be executed
inheritGroups – true/false	If set to true, method belongs to groups specified in @Test at class level

The attributes mentioned above are also applicable to @Test annotation.

Key Attributes of @Test

Attributes	Description
groups	The list of groups method belongs to • @Test(groups = {“Regression”, “Smoke”, “Fastest”, “ForFireFoxOnly”})
dependsOnGroups	Declares the **groups** on which the method depends • @Test(groups=“Smoke”, dependsOnGroups = {“ServerStart”, “SignIn”})
dependsOnMethods	Declares another method on which the method depends • @Test(dependsOnMethods = “Login”)
alwaysRun – true/false	If set to true, Method(s) will run regardless of its groups or failure of dependent groups/methods. • @Test(alwaysRun = true)
dataProvider	Name of the data provider that will provide data to this test • @Test(dataProvider = “SignInCredentials”)

Key Attributes of @Test

Attributes	Description
dataProviderClass	Class in which data provider is defined and the data provider method should be static. • @Test(dataProvider = “SignInCredentials”, dataProviderClass=“singin.SignInDataProvider.class)
enabled – true/false	It is set to true by default and if set to false, test method will not be executed • @Test(enabled = false)
description	• @Test(description = “This Test Calculates BMI when Height and Weight are provided as inputs”)
expectedExceptions	Used for exception and negative testing, it is the list of exceptions that the test method should throw • @Test(expectedExceptions = NoSuchElementException.class)

You can find the complete list of all the attributes at http://TestNG.org/doc/documentation-main.html

@Test at Class Level

You can have @Test at class level. This way, all the public methods inside a class get converted into tests and inherit all the groups specified in @Test.

```
@Test (groups = “Smoke”)
 public class TestClass {
   @Test(groups = {“win32”, “mac”,”linux”})
     public void testMethodOne(){
     System.out.println(“This is First Test belongs to 4 groups");
     }

     @Test(groups = “win64”)
     public void testMethodTwo(){
     System.out.println(“This is Second Test belongs to 2 groups");
     }

     private void testMethodThree(){
     System.out.println(“This is a private method and hence cannot become test and does not
     inherit groups.");
     }
 }
```

Test Groups and TestNG.xml

The groups defined in @Test can be configured in XML in various ways:

- Include/exclude group(s)
- Specify at Suite level vs. Test level
- Support for regular expressions – .* (dot-star)
- You can specify Group of Groups – MetaGroups
- This helps in minimum recompilation due to configuration flexibility

Test Groups: Naming Conventions

You can give descriptive names to test groups based on your application context.

You can name the test based on:

- Type of test (Functional, Regression, and Check-In)
- Continuous Integration (NightlyBuild, WeeklyBuild, PatchRelease, and EveryBuild)
- Speed of execution (VerySlow, VeryFast, Fast, and Slow)
- Module (CurrentAccount, SavingsAccount, and Loan)
- Independence (DataIndependent, and DataDependent)

TestNG: Order of Execution and Dependencies

- Default Order – ascending order of @Test Method names
- Dependency on method – same class/base class
- Dependency on groups – same class/other classes
- Hard vs. Soft Dependencies (use of alwaysRun = true)
- Fail vs. Skip
- Regular Expression in dependsOnGroups

```
@Test(dependsOnGroups = { "SavingAccount.*})
```

Create dependency inside xml

```
<dependencies>
  <group name="Check-In" depends-on="CurrentAccount SavingsAccount.*"/>
<dependencies>
```

Parallel Test Execution Capability

TestNG tests can run in separate threads using **parallel** and **thread-count** attributes that facilitate parallel execution.

Test method in parallel

- <suite name="parallel Suite" parallel="methods" thread-count="8">

Test classe in parallel

- <suite name="parallel Suite" parallel="classes" thread-count="3">

Tests in different instance in parallel

- <suite name="parallel Suite" parallel="instances" thread-count="2">

Tests in a test suite in parallel

- <suite name="parallel Suite" parallel="tests" thread-count="2">

Configuring an independent test in parallel

- @Test(threadPoolSize=5,invocationCount=25,timeOut=60000)

These attributes reduce execution time and can test multithreaded code in an application.

Group Test Cases and Parallel Execution

Duration: 20 min.

Problem Statement:
Using Selenium TestNG, write a program to run all the test cases in different groups and use “dependsOnMethod” annotation. Use TestNG.xml file in order to add or remove classes for executing multiple classes.

Assisted Practice: Guidelines

Steps to perform group test cases and parallel execution:

1. Open Eclipse
2. Create a Java project and name it as TestGroups
3. Create a TestNG class file
4. Write the code for the test case methods and group them together in the attribute
5. Create **testng.xml** file in order to add classes with test cases to run together
6. Run the program as a TestNG project
7. Push the code to GitHub repositories

Assertion APIs to Evaluate Tests

Assertion APIs

There should be at least one, and preferably only one, assertion for every test you write.

It is best to choose assertEquals() over assertTrue() and assertFalse() as it shows the exact mismatch between expected and actual assertion.

Assertion APIs: Examples

```
assertEquals(<primitive Type> actual, <primitive Same Type> expected)
```

```
assertEquals(<primitive Type> actual, <primitive Same Type> expected, String
message)
```

```
assertEquals(java.lang.String actual, java.lang.String expected, java.lang.String
message)
```

```
assertEquals(java.util.Collection actual, java.util.Collection expected,
java.lang.String message) // Takes two collection objects and verifies both of
the collections contain the same elements and in the same order.
```

Assertion APIs: Examples

```
assertTrue(boolean condition, java.lang.String message)
```

```
assertFalse(boolean condition, java.lang.String message)
```

```
fail(java.lang.String message)// Directly fails a test with the given message.
This method is mainly used while handling exception conditions and when we have
to failed the test forcefully.
```

Evaluating Test Cases

Duration: 20 min.

Problem Statement:
Write a program to run all test cases and attach asserts. By doing this, you can track their results in real time using Selenium TestNG.

Assisted Practice: Guidelines

Steps to evaluate test cases:

1. Open Eclipse
2. Create a Java project and name it TestAssert
3. Create a TestNG class file and name it Assertions.java
4. Write the code for test cases methods and use the assert method
5. Create **testng.xml** file
6. Run the program as a TestNG Suite
7. Push the code to GitHub repositories

FULL STACK

Reports Using TestNG

simplilearn

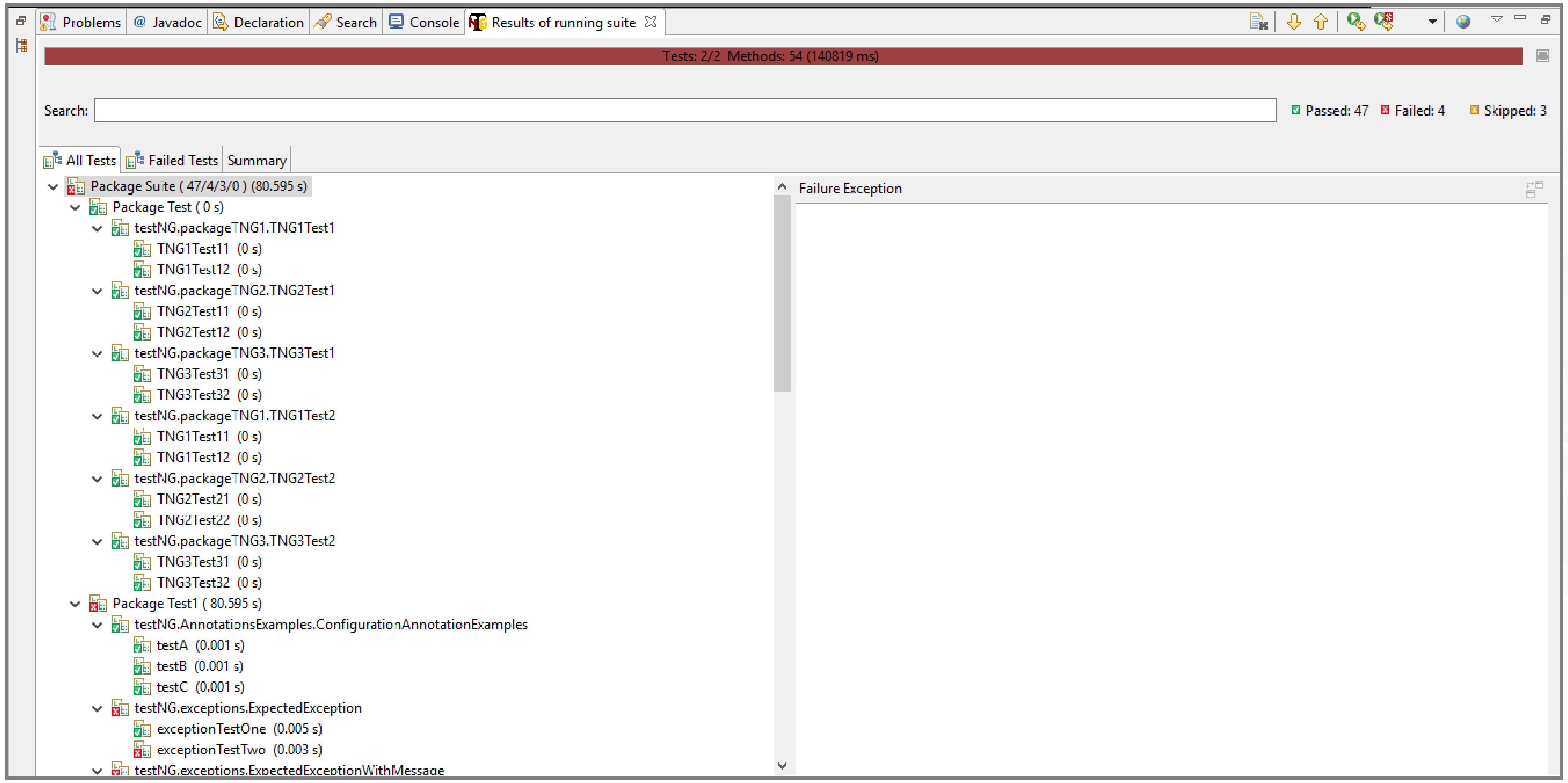
Reports Using TestNG

TestNG supports logging and reporting. Using TestNG, you can:

- Utilize default reports provided by TestNG
- Generate a JUnit HTML report
- Generate reports using ReportNG
- Create your own custom reporters and logger

TestNG: Default HTML and XML Reports

TestNG generates default HTML and XML reports for each run. These reports are generated in the test-output folder.



Tests: 2/2 Methods: 54 (140819 ms)

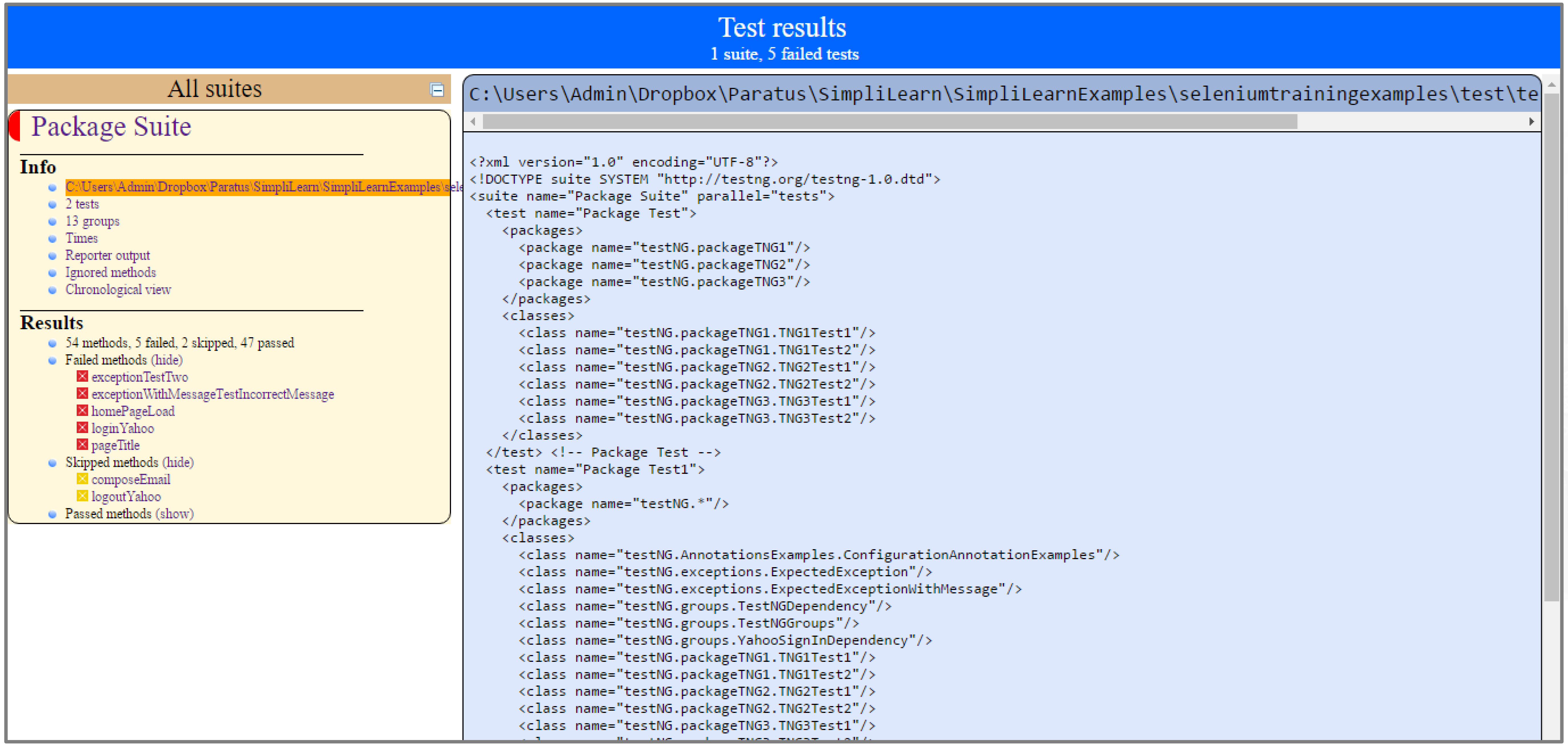
Search:

Passed: 47 Failed: 4 Skipped: 3

```
Package Suite ( 47/4/3/0 ) (80.595 s)
  Package Test ( 0 s)
    testNG.packageTNG1.TNG1Test1
      TNG1Test11  (0 s)
      TNG1Test12  (0 s)
    testNG.packageTNG2.TNG2Test1
      TNG2Test11  (0 s)
      TNG2Test12  (0 s)
    testNG.packageTNG3.TNG3Test1
      TNG3Test31  (0 s)
      TNG3Test32  (0 s)
    testNG.packageTNG1.TNG1Test2
      TNG1Test11  (0 s)
      TNG1Test12  (0 s)
    testNG.packageTNG2.TNG2Test2
      TNG2Test21  (0 s)
      TNG2Test22  (0 s)
    testNG.packageTNG3.TNG3Test2
      TNG3Test31  (0 s)
      TNG3Test32  (0 s)
  Package Test1 ( 80.595 s)
    testNG.AnnotationsExamples.ConfigurationAnnotationExamples
      testA  (0.001 s)
      testB  (0.001 s)
      testC  (0.001 s)
    testNG.exceptions.ExpectedException
      exceptionTestOne  (0.005 s)
      exceptionTestTwo  (0.003 s)
    testNG.exceptions.ExpectedExceptionWithMessage
```

TestNG: Default HTML Report

The image below shows various sections of a default HTML report:



Test results
1 suite, 5 failed tests

All suites

Package Suite

Info

- C:\Users\Admin\Dropbox\Paratus\SimpliLearn\SimpliLearnExamples\sele
- 2 tests
- 13 groups
- Times
- Reporter output
- Ignored methods
- Chronological view

Results

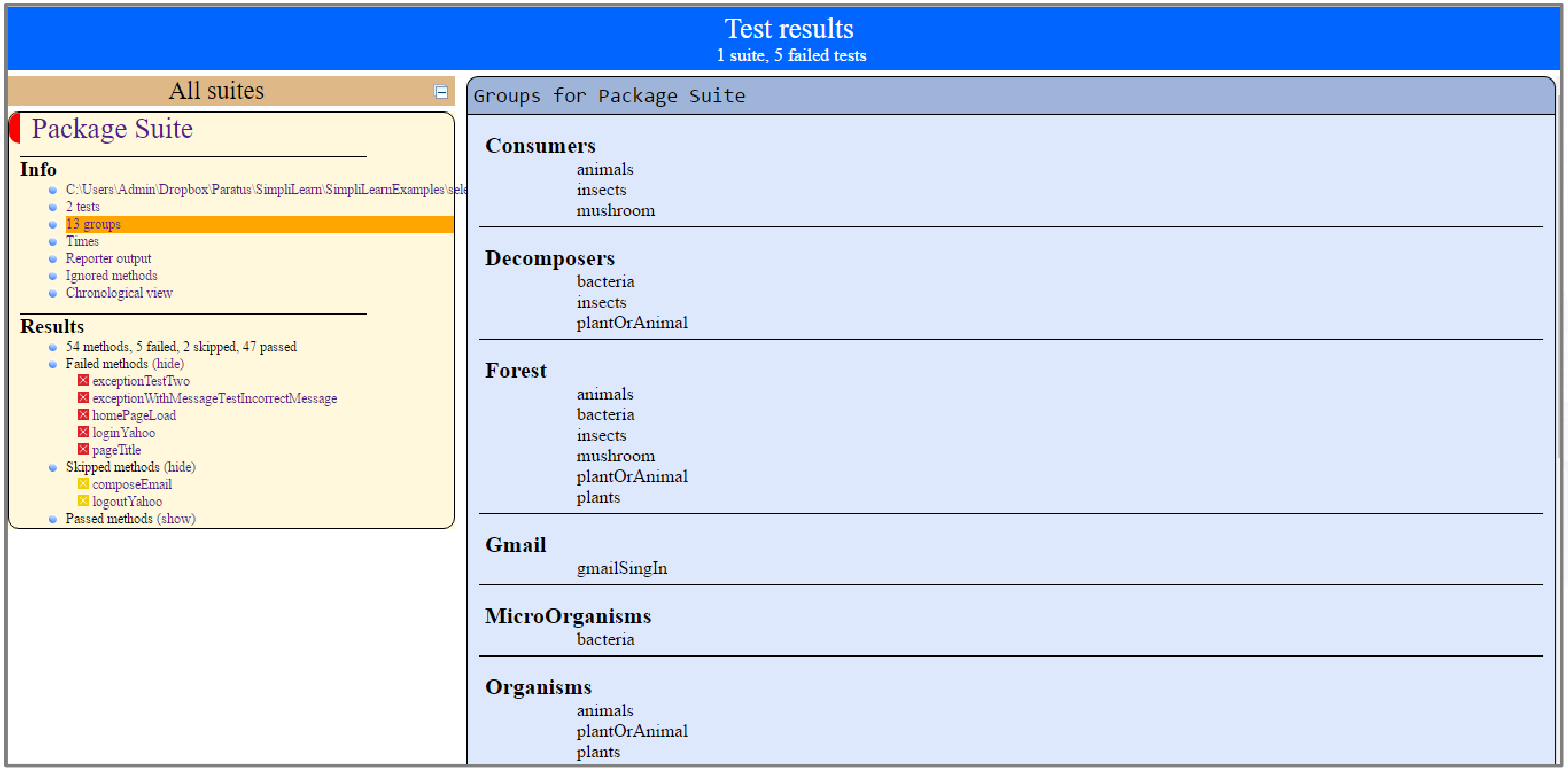
- 54 methods, 5 failed, 2 skipped, 47 passed
- Failed methods (hide)
  - exceptionTestTwo
  - exceptionWithMessageTestIncorrectMessage
  - homePageLoad
  - loginYahoo
  - pageTitle
- Skipped methods (hide)
  - composeEmail
  - logoutYahoo
- Passed methods (show)

C:\Users\Admin\Dropbox\Paratus\SimpliLearn\SimpliLearnExamples\seleniumtrainingexamples\test\te

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Package Suite" parallel="tests">
  <test name="Package Test">
    <packages>
      <package name="testNG.packageTNG1"/>
      <package name="testNG.packageTNG2"/>
      <package name="testNG.packageTNG3"/>
    </packages>
    <classes>
      <class name="testNG.packageTNG1.TNG1Test1"/>
      <class name="testNG.packageTNG1.TNG1Test2"/>
      <class name="testNG.packageTNG2.TNG2Test1"/>
      <class name="testNG.packageTNG2.TNG2Test2"/>
      <class name="testNG.packageTNG3.TNG3Test1"/>
      <class name="testNG.packageTNG3.TNG3Test2"/>
    </classes>
  </test> <!-- Package Test -->
  <test name="Package Test1">
    <packages>
      <package name="testNG.*"/>
    </packages>
    <classes>
      <class name="testNG.AnnotationsExamples.ConfigurationAnnotationExamples"/>
      <class name="testNG.exceptions.ExpectedException"/>
      <class name="testNG.exceptions.ExpectedExceptionWithMessage"/>
      <class name="testNG.groups.TestNGDependency"/>
      <class name="testNG.groups.TestNGGroups"/>
      <class name="testNG.groups.YahooSignInDependency"/>
      <class name="testNG.packageTNG1.TNG1Test1"/>
      <class name="testNG.packageTNG1.TNG1Test2"/>
      <class name="testNG.packageTNG2.TNG2Test1"/>
      <class name="testNG.packageTNG2.TNG2Test2"/>
      <class name="testNG.packageTNG3.TNG3Test1"/>
```

TestNG: Default HTML Report

The image below shows details of groups provided in the HTML report:



Test results
1 suite, 5 failed tests

All suites

Package Suite

Info
- C:\Users\Admin\Dropbox\Paratus\SimpliLearn\SimpliLearnExamples\sele
- 2 tests
- 13 groups
- Times
- Reporter output
- Ignored methods
- Chronological view

Results
- 54 methods, 5 failed, 2 skipped, 47 passed
- Failed methods (hide)
  - exceptionTestTwo
  - exceptionWithMessageTestIncorrectMessage
  - homePageLoad
  - loginYahoo
  - pageTitle
- Skipped methods (hide)
  - composeEmail
  - logoutYahoo
- Passed methods (show)

Groups for Package Suite

Consumers
animals
insects
mushroom

Decomposers
bacteria
insects
plantOrAnimal

Forest
animals
bacteria
insects
mushroom
plantOrAnimal
plants

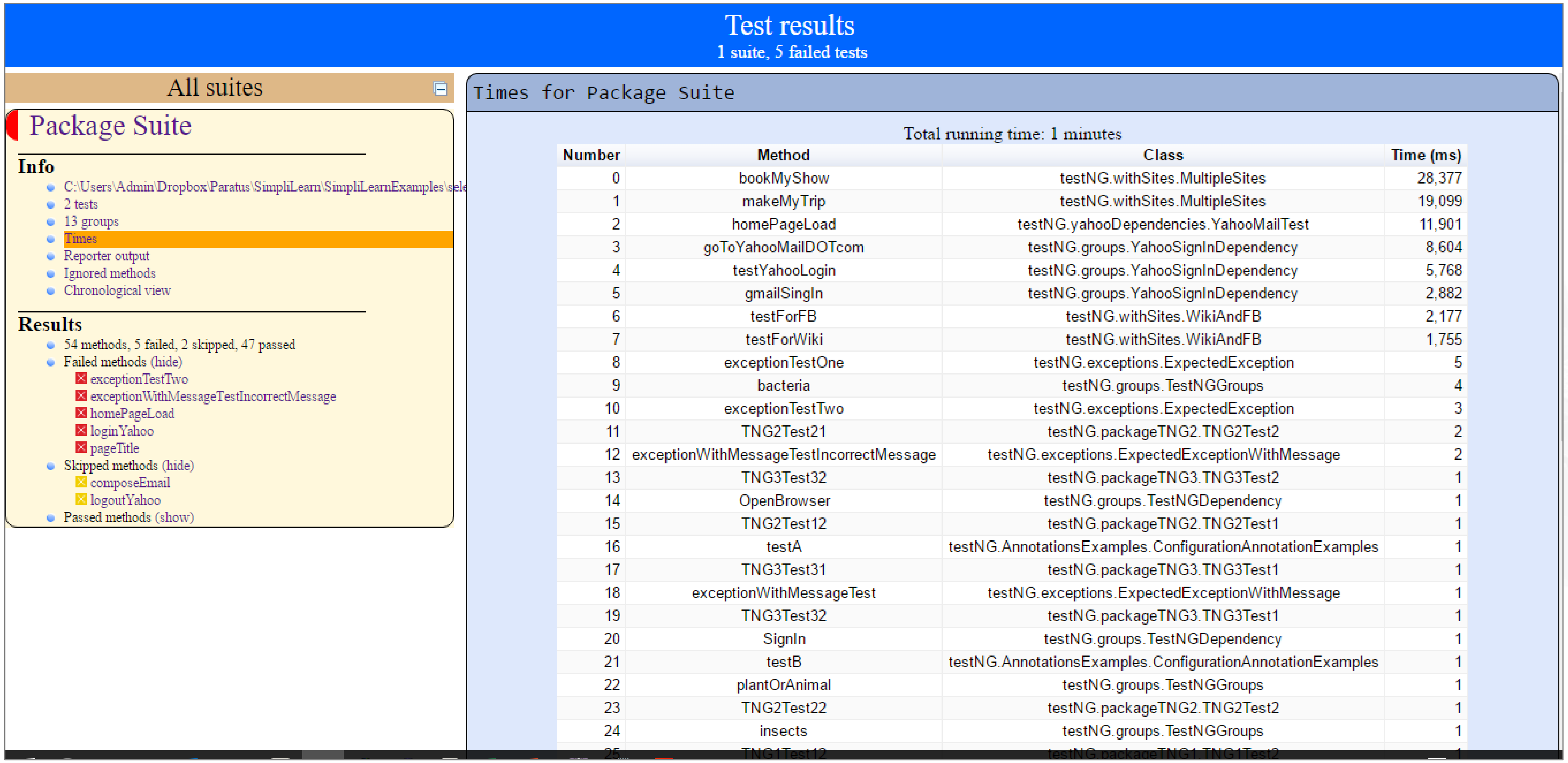
Gmail
gmailSingIn

MicroOrganisms
bacteria

Organisms
animals
plantOrAnimal
plants

TestNG: Default HTML Report

The screenshot below shows execution time for each test:



Test results

1 suite, 5 failed tests

All suites

Package Suite

Info

- C:\Users\Admin\Dropbox\Paratus\SimpliLearn\SimpliLearnExamples\sele
- 2 tests
- 13 groups
- Times
- Reporter output
- Ignored methods
- Chronological view

Results

- 54 methods, 5 failed, 2 skipped, 47 passed
- Failed methods (hide)
  - exceptionTestTwo
  - exceptionWithMessageTestIncorrectMessage
  - homePageLoad
  - loginYahoo
  - pageTitle
- Skipped methods (hide)
  - composeEmail
  - logoutYahoo
- Passed methods (show)

Times for Package Suite

Total running time: 1 minutes

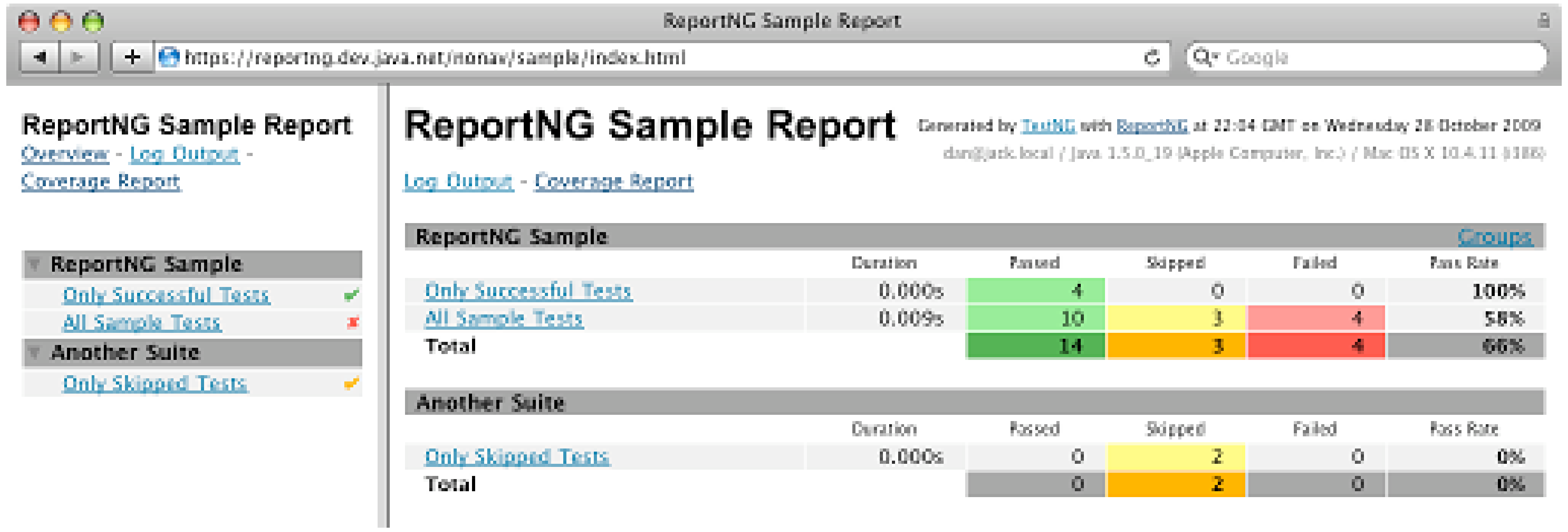
Number	Method	Class	Time (ms)
0	bookMyShow	testNG.withSites.MultipleSites	28,377
1	makeMyTrip	testNG.withSites.MultipleSites	19,099
2	homePageLoad	testNG.yahooDependencies.YahooMailTest	11,901
3	goToYahooMailDOTcom	testNG.groups.YahooSignInDependency	8,604
4	testYahooLogin	testNG.groups.YahooSignInDependency	5,768
5	gmailSingIn	testNG.groups.YahooSignInDependency	2,882
6	testForFB	testNG.withSites.WikiAndFB	2,177
7	testForWiki	testNG.withSites.WikiAndFB	1,755
8	exceptionTestOne	testNG.exceptions.ExpectedException	5
9	bacteria	testNG.groups.TestNGGroups	4
10	exceptionTestTwo	testNG.exceptions.ExpectedException	3
11	TNG2Test21	testNG.packageTNG2.TNG2Test2	2
12	exceptionWithMessageTestIncorrectMessage	testNG.exceptions.ExpectedExceptionWithMessage	2
13	TNG3Test32	testNG.packageTNG3.TNG3Test2	1
14	OpenBrowser	testNG.groups.TestNGDependency	1
15	TNG2Test12	testNG.packageTNG2.TNG2Test1	1
16	testA	testNG.AnnotationsExamples.ConfigurationAnnotationExamples	1
17	TNG3Test31	testNG.packageTNG3.TNG3Test1	1
18	exceptionWithMessageTest	testNG.exceptions.ExpectedExceptionWithMessage	1
19	TNG3Test32	testNG.packageTNG3.TNG3Test1	1
20	SignIn	testNG.groups.TestNGDependency	1
21	testB	testNG.AnnotationsExamples.ConfigurationAnnotationExamples	1
22	plantOrAnimal	testNG.groups.TestNGGroups	1
23	TNG2Test22	testNG.packageTNG2.TNG2Test2	1
24	insects	testNG.groups.TestNGGroups	1
25	TNG1Test12	testNG.packageTNG1.TNG1Test2	1

Reports Using ReportNG

- ReportNG is a simple plug-in for the TestNG unit-testing framework, which generates HTML reports as a replacement of default TestNG HTML reports. You can also customize HTML reports with the help of TestNG listeners.
- To use ReportNG reports, we need to follow these steps:

Step 1: Add the following .jar files to your project.

reporting-1.1.4.jar
velocity-dep-1.4.jar
guice-3.0.jar



ReportNG Sample Report
Overview - Log Output -
Coverage Report

ReportNG Sample
Only Successful Tests ✓
All Sample Tests ✗
Another Suite
Only Skipped Tests ✓

ReportNG Sample Report

Generated by TestNG with ReportNG at 22:04 GMT on Wednesday 28 October 2009
dan@jack.local / Java 1.5.0_19 (Apple Computer, Inc.) / Mac OS X 10.4.11 (i386)

Log Output - Coverage Report

ReportNG Sample — Groups

	Duration	Passed	Skipped	Failed	Pass Rate
Only Successful Tests	0.000s	4	0	0	100%
All Sample Tests	0.009s	10	3	4	58%
Total		14	3	4	66%

Another Suite

	Duration	Passed	Skipped	Failed	Pass Rate
Only Skipped Tests	0.000s	0	2	0	0%
Total		0	2	0	0%

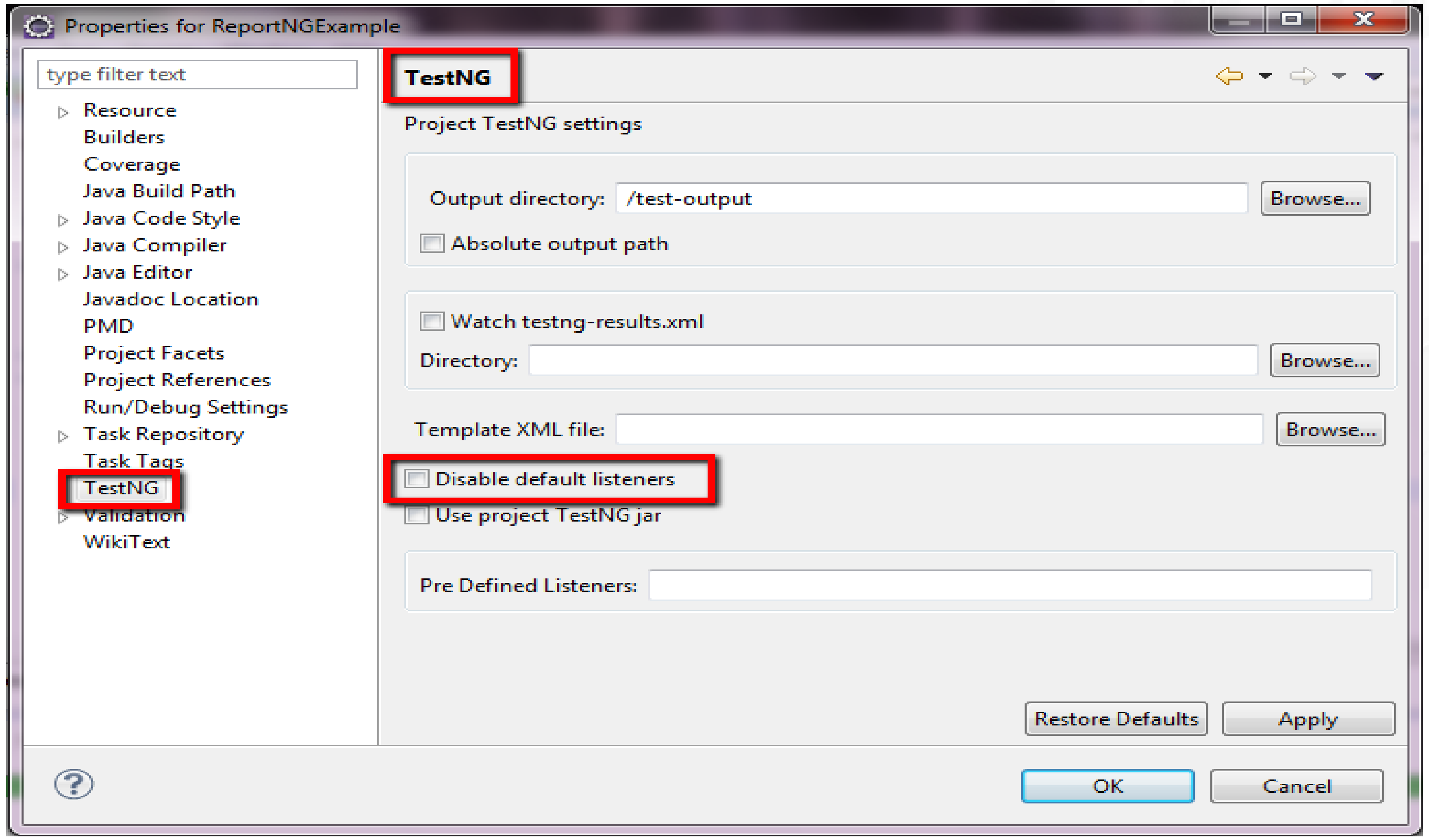
Reports Using ReportNG

Step 2: Disable the default TestNG listeners to make sure ReportNG reports run.

To disable:

1. Right click on properties
2. Click on TestNG
3. Select **Disable default listeners**
4. Click on **Apply** to save **project preferences**
5. Now, click on **OK**

The screen looks like:



Properties for ReportNGExample

type filter text

- Resource
- Builders
- Coverage
- Java Build Path
- Java Code Style
- Java Compiler
- Java Editor
- Javadoc Location
- PMD
- Project Facets
- Project References
- Run/Debug Settings
- Task Repository
- Task Tags
- TestNG
- Validation
- WikiText

TestNG

Project TestNG settings

Output directory: /test-output Browse...

Absolute output path

Watch testng-results.xml

Directory: Browse...

Template XML file: Browse...

Disable default listeners

Use project TestNG jar

Pre Defined Listeners:

Restore Defaults Apply

OK Cancel

Reports Using ReportNG

Step 3: Add the following listeners to the testing.xml file.

```
<listeners>

<listener class-name="org.uncommons.reportng.HTMLReporter"/>
<listener class-name="org.uncommons.reportng.JUnitXMLReporter"/>

</listeners>
```

Selenium with Jenkins

Why Selenium and Jenkins?

1. TestNG test: Jenkins can be configured to run the automation test build on TestNG after each build of SVN.

1. With Jenkins, you can run test cases every time there is a change in your application, making it almost bug free.

1. Jenkins can schedule your tests to run at specific times.

1. With the help of Jenkins, you can save the history of the test cases and also generate periodic test reports.

1. Jenkins also supports Maven and testing of applications with continuous integration.

Integrating Selenium with Jenkins

Duration: 30 min.

Problem Statement:
Write a program to integrate TestNG and Selenium with Jenkins. Implement continuous integration and execute test cases.

Assisted Practice: Guidelines

Steps to integrate selenium with Jenkins:

1. Open Eclipse and create a Maven project

2. In pom.xml, add the Maven, JUnit, Selenium, and TestNG dependencies

3. Add test cases code in the new test class created with several configuration annotations

4. Convert it into TestNG, which will create TestNG.xml

5. Add Maven-compiler-plugin, Maven-surefire-plugin, TestNG.xml into pom.xml, and run as Maven test

6. Install and open Jenkins

7. Create a new job with Maven

8. Select manage Jenkins=>configure Jenkins=>build section=> provide absolute path of pom.xml

9. Build the project and then click on the project name. Check the console for test results

10. Push the code to GitHub repositories

Selenium Grid

Selenium Grid

Selenium Grid allows you to:

Set up a distributed execution environment

Split and run your tests in parallel across multiple machines, reducing the execution time

Run your tests across multiple platforms (Windows, Mac, Linux, and mobile devices)

Do cross browser testing

When to Use Selenium Grid?

Q

- Consider a startup company which has created a web application for hospital and clinic management. They support all the key browsers in the market, like IE, Firefox, Chrome, Safari, and Opera. They support all the key platforms, like Windows—all versions, Mac, and Linux variants.

- They have automated approximately 12000 regression test cases using Selenium. It takes around 4-5 days to execute them in a sequence in any particular environment.

- They follow an agile methodology with sprint of 5 weeks. They have factored around 1.5 weeks of testing towards the end of sprint. Even though they are well funded, the challenge is how to ensure these 12K test cases are not broken during every sprint across the browsers and platforms they support.

When to Use Selenium Grid?

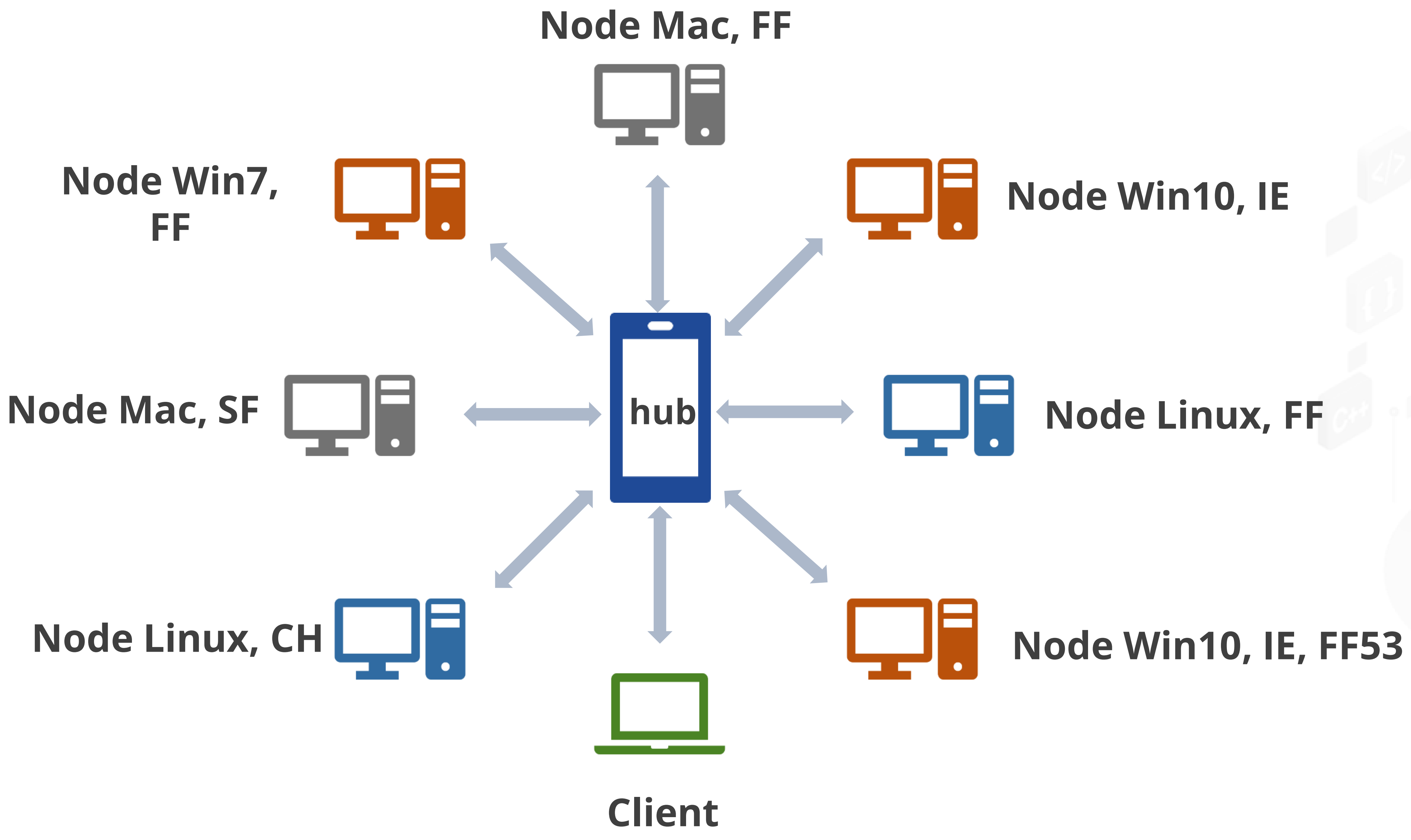
A

- Selenium Grid, along with TestNG, will help the company to reduce execution time significantly.
- Using Selenium Grid, the company can:
    - Set up VMs having a different combination of browsers and platforms
    - Set up VMs having same browser but different versions across multiple VMs
    - Split the Test Cases across multiple VMs

- This will reduce the execution time and increase the coverage. Developers now will get faster feedback on their changes, and the company will be in a good position to set up Continuous Integration using tools like Jenkins/Bamboo.

Hubs and Nodes

Example of one hub connected to multiple nodes:



Hubs and Nodes

Following is the information about the grid setup:

- There are one hub and seven nodes.
- Each node supports a specific platform and browser(s).
- Each node is registered with a hub.
- Client machine containing test scripts sends a request to hub for executing tests.
- Test script also sends the capabilities it requires from the test, which is to be executed on, such as platform, browser, and version of the browser.
- Hub assigns an appropriate node with the required capabilities to execute the test.
- After executing, the node sends the results to hub.
- Hub, in turn, sends the results to the Client machine from where the test script execution was requested.

Creating a Hub

Key points about Hub:

- It is like an agency or a broker that routes the client requests to node(s).
- It has information on the platform and browser supported by all the nodes.
- It is a Selenium standalone server, which needs to be started with specific parameters.
- By default, it listens for the requests on port number 4444.

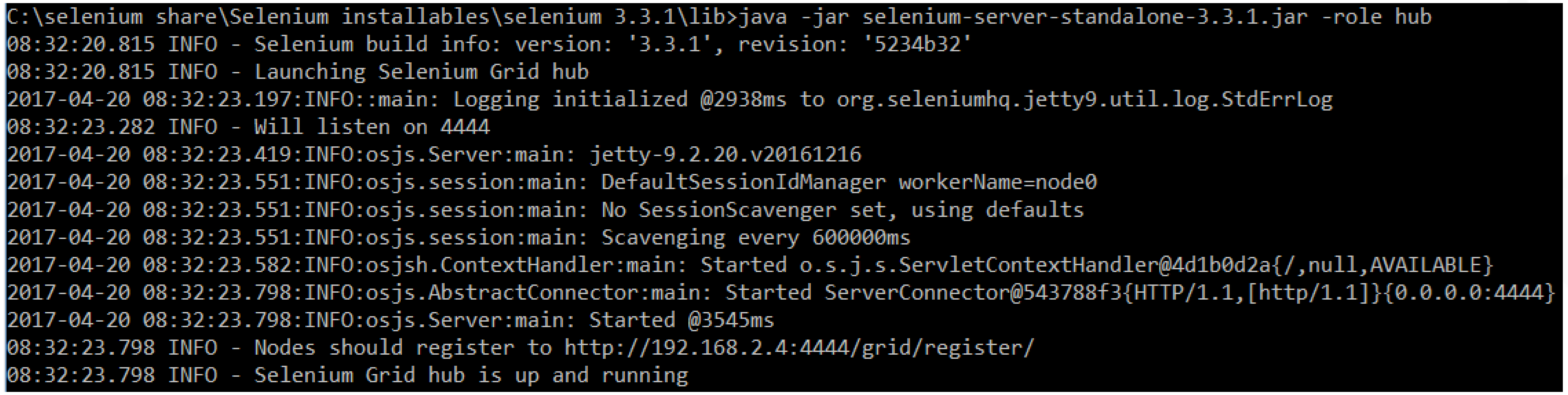
What you need for Hub Machine:

- Java should be installed on the machine which you want to make a hub.
- The machine should have enough memory and network bandwidth to communicate with the client machine and various nodes.
- Download selenium-server-standalone-<latest version>.jar file from Selenium website.
- The link to download Selenium standalone server version 3.3.1 is https://goo.gl/uTXEJ1.

Creating a Hub

Starting Hub:

1. Go to command prompt
2. Go to the location where you have kept the Selenium Standalone Server jar file
3. Execute the jar file with following parameters: Java –jar selenium-server-standalone-3.3.1.jar –role hub –newSessionWaitTimeout 100000



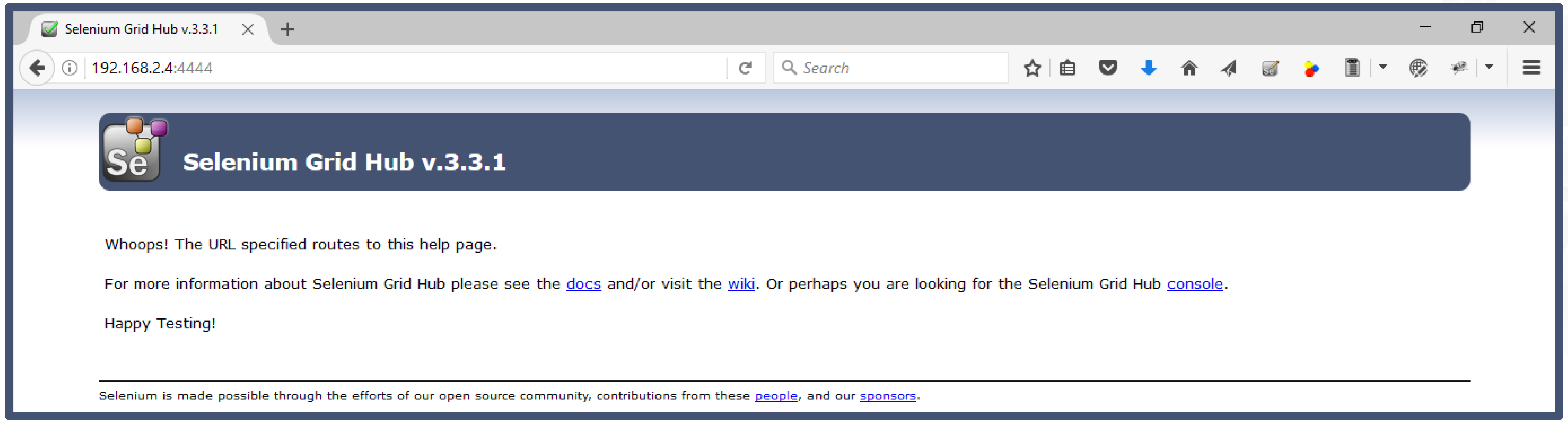
```
C:\selenium share\Selenium installables\selenium 3.3.1\lib>java -jar selenium-server-standalone-3.3.1.jar -role hub
08:32:20.815 INFO - Selenium build info: version: '3.3.1', revision: '5234b32'
08:32:20.815 INFO - Launching Selenium Grid hub
2017-04-20 08:32:23.197:INFO::main: Logging initialized @2938ms to org.seleniumhq.jetty9.util.log.StdErrLog
08:32:23.282 INFO - Will listen on 4444
2017-04-20 08:32:23.419:INFO:osjs.Server:main: jetty-9.2.20.v20161216
2017-04-20 08:32:23.551:INFO:osjs.session:main: DefaultSessionIdManager workerName=node0
2017-04-20 08:32:23.551:INFO:osjs.session:main: No SessionScavenger set, using defaults
2017-04-20 08:32:23.551:INFO:osjs.session:main: Scavenging every 600000ms
2017-04-20 08:32:23.582:INFO:osjsh.ContextHandler:main: Started o.s.j.s.ServletContextHandler@4d1b0d2a{/,null,AVAILABLE}
2017-04-20 08:32:23.798:INFO:osjs.AbstractConnector:main: Started ServerConnector@543788f3{HTTP/1.1,[http/1.1]}{0.0.0.0:4444}
2017-04-20 08:32:23.798:INFO:osjs.Server:main: Started @3545ms
08:32:23.798 INFO - Nodes should register to http://192.168.2.4:4444/grid/register/
08:32:23.798 INFO - Selenium Grid hub is up and running
```

! For help on various Grid configurations, use java –jar selenium-server-standalone-3.3.1.jar –help.

Viewing a Hub in Browser

Monitoring Hub from Browser:

1. By default, a hub starts listening on port 4444
2. Go to any browser and provide IP address: Port No. (e.g. 192.168.2.4:4444)
3. Browser will show the hub console as shown in the screenshot below:



Whoops! The URL specified routes to this help page.

For more information about Selenium Grid Hub please see the docs and/or visit the wiki. Or perhaps you are looking for the Selenium Grid Hub console.

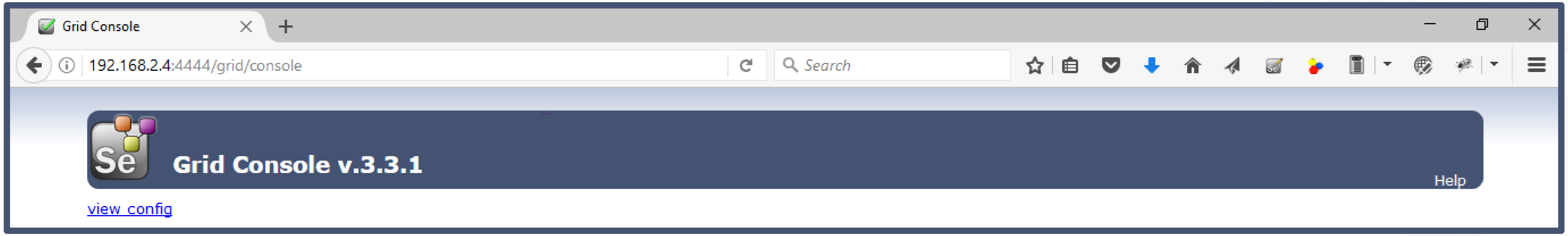
Happy Testing!

Selenium is made possible through the efforts of our open source community, contributions from these people, and our sponsors.

! Hub on other port - java –jar selenium-server-standalone-3.3.1.jar –role hub –port 4445

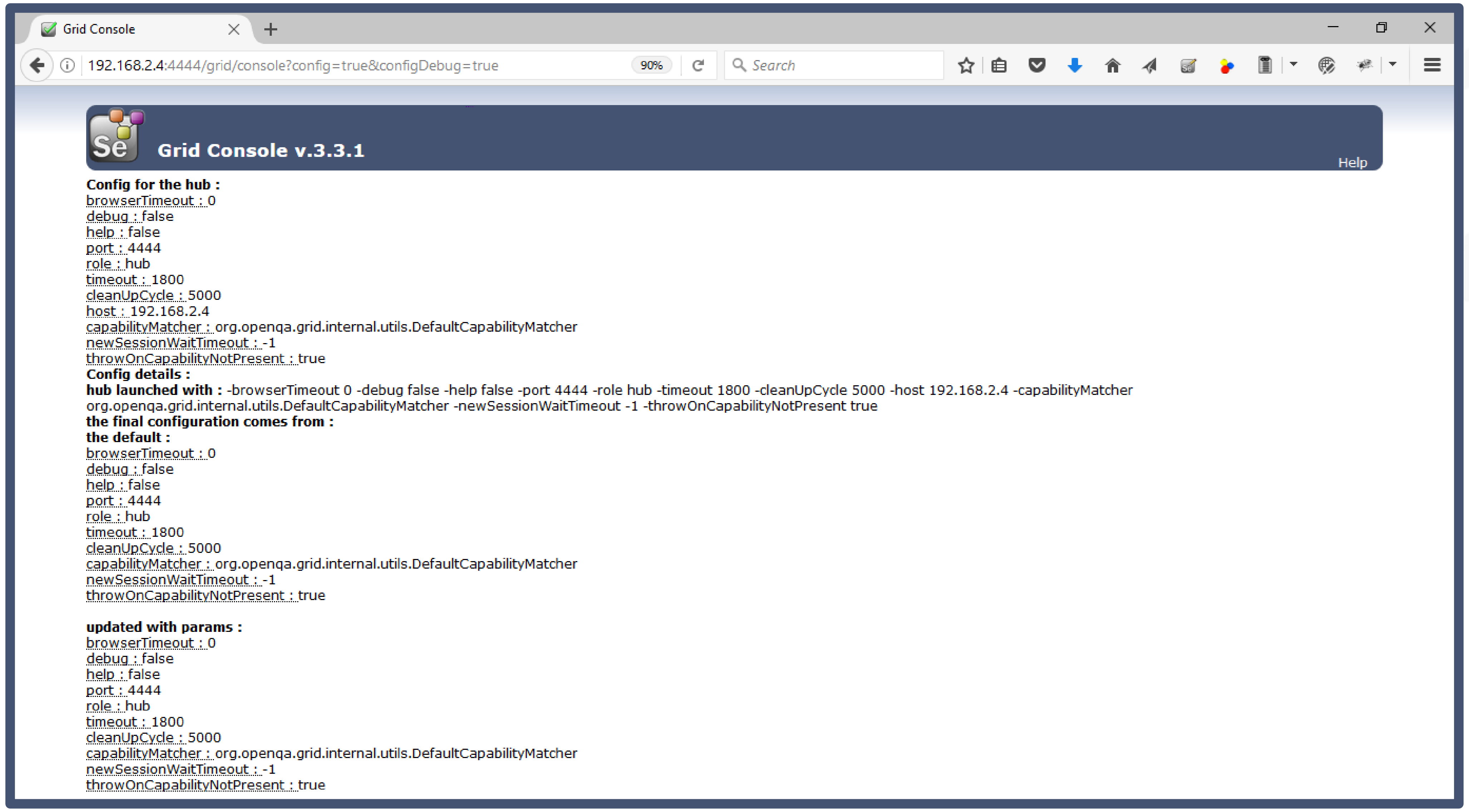
Viewing a Hub Configuration

You can view the hub configuration when you click on the **console** link and it will take you to the screen below:



Viewing a Hub Configuration

Click on **view config** link in the previous screen and it will take you to Grid Console Configuration. The screen below shows various configuration settings on Hub:



```
Config for the hub :
browserTimeout : 0
debug : false
help : false
port : 4444
role : hub
timeout : 1800
cleanUpCycle : 5000
host : 192.168.2.4
capabilityMatcher : org.openqa.grid.internal.utils.DefaultCapabilityMatcher
newSessionWaitTimeout : -1
throwOnCapabilityNotPresent : true
Config details :
hub launched with : -browserTimeout 0 -debug false -help false -port 4444 -role hub -timeout 1800 -cleanUpCycle 5000 -host 192.168.2.4 -capabilityMatcher
org.openqa.grid.internal.utils.DefaultCapabilityMatcher -newSessionWaitTimeout -1 -throwOnCapabilityNotPresent true
the final configuration comes from :
the default :
browserTimeout : 0
debug : false
help : false
port : 4444
role : hub
timeout : 1800
cleanUpCycle : 5000
capabilityMatcher : org.openqa.grid.internal.utils.DefaultCapabilityMatcher
newSessionWaitTimeout : -1
throwOnCapabilityNotPresent : true

updated with params :
browserTimeout : 0
debug : false
help : false
port : 4444
role : hub
timeout : 1800
cleanUpCycle : 5000
capabilityMatcher : org.openqa.grid.internal.utils.DefaultCapabilityMatcher
newSessionWaitTimeout : -1
throwOnCapabilityNotPresent : true
```

Creating a Node

Key points about Node:

- Each node has a specific capability in terms of platform and browsers it can serve.
- A node is created using Selenium standalone server.
- By default, it listens for requests on port number 5555.
- A node can be a physical or a virtual machine.

What you need for Node Machine:

- Java should be installed on the machine which you want to make a Node.
- All browsers and their driver executables should be installed and kept on a certain path.
- Machine should have enough memory and network bandwidth to communicate with Hub and start multiple browsers it is configured for.
- Download selenium-server-standalone-<latest version>.jar file from Selenium website.
- The link to download Selenium standalone server version 3.3.1 is https://goo.gl/uTXEJ1.

Creating a Node

Starting Node:

1. Go to the command prompt
2. Go to the location where you have kept the Selenium Standalone Server jar file
3. Execute the jar file with following parameters:

```
java -Dwebdriver.gecko.driver="\selenium share\Selenium installables\selenium
3.3.1\resources\geckodriver.exe" ^
-Dwebdriver.chrome.driver="\selenium share\Selenium installables\selenium
3.3.1\resources\chromedriver.exe" ^
-Dwebdriver.ie.driver="\selenium share\Selenium installables\selenium
3.3.1\resources\ieDriverServer.exe" ^
-jar selenium-server-standalone-3.3.1.jar -role node ^
-hub http://192.168.2.4:4444/grid/register ^
-browser "browserName=firefox,maxInstances=1,platform=Windows" ^
-browser "browserName=chrome,maxInstances=3,platform=Linux" ^
-browser "browserName=internet explorer,maxInstances=2" ^
-browser "browserName=opera,maxInstances=6" ^
-browser "browserName=android,maxInstances=4" ^
-browser "browserName=safari,maxInstances=5"
```

Understanding Node Creation

Configuring the path of browser driver executable using –D switch:

```
java -Dwebdriver.gecko.driver="\selenium share\Selenium
installables\selenium 3.3.1\resources\geckodriver.exe" ^
-Dwebdriver.chrome.driver="\selenium share\Selenium installables\selenium
3.3.1\resources\chromedriver.exe" ^
-Dwebdriver.ie.driver="\selenium share\Selenium installables\selenium
3.3.1\resources\ieDriverServer.exe" ^
```

Understanding Node Creation

Following code invocates the Selenium-server-standalone jar file as a node and provides hub address to which the invocated node is connected:

```
java -Dwebdriver.gecko.driver="\selenium share\Selenium
installables\selenium 3.3.1\resources\geckodriver.exe" ^
-Dwebdriver.chrome.driver="\selenium share\Selenium
installables\selenium 3.3.1\resources\chromedriver.exe" ^
-Dwebdriver.ie.driver="\selenium share\Selenium installables\selenium
3.3.1\resources\ieDriverServer.exe" ^
-jar selenium-server-standalone-3.3.1.jar -role node ^
-hub http://192.168.2.4:4444/grid/register ^
```

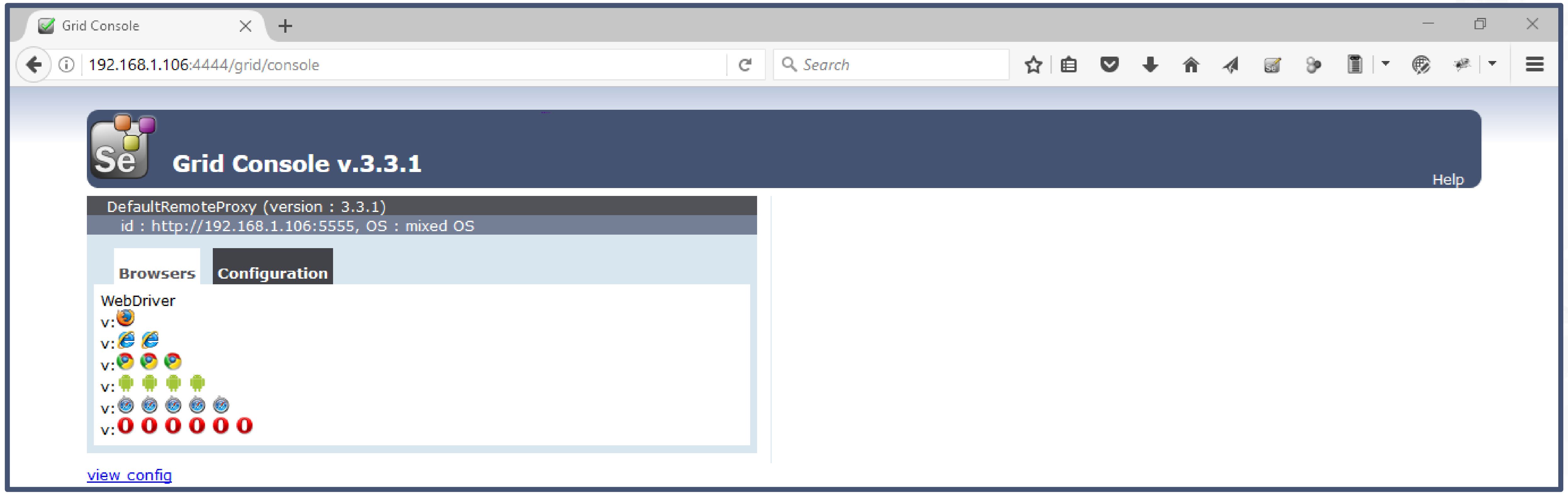
Understanding Node Creation

Configuring browsers, version, platform, and number of instances:

```
java -Dwebdriver.gecko.driver="\selenium share\Selenium installables\selenium
3.3.1\resources\geckodriver.exe" ^
-Dwebdriver.chrome.driver="\selenium share\Selenium installables\selenium
3.3.1\resources\chromedriver.exe" ^
-Dwebdriver.ie.driver="\selenium share\Selenium installables\selenium
3.3.1\resources\ieDriverServer.exe" ^
-jar selenium-server-standalone-3.3.1.jar -role node ^
-hub http://192.168.2.4:4444/grid/register ^
-browser "browserName=firefox,maxInstances=1,platform=Windows,version=53.5” ^
-browser "browserName=chrome,maxInstances=3,platform=Linux" ^
-browser "browserName=internet explorer,maxInstances=2" ^
-browser "browserName=opera,maxInstances=6" ^
-browser "browserName=android,maxInstances=4" ^
-browser "browserName=safari,maxInstances=5"
```

Monitoring Node from Browser

You can monitor the node from the browser in Grid Console. The browser will show the node as shown in the screenshot below:



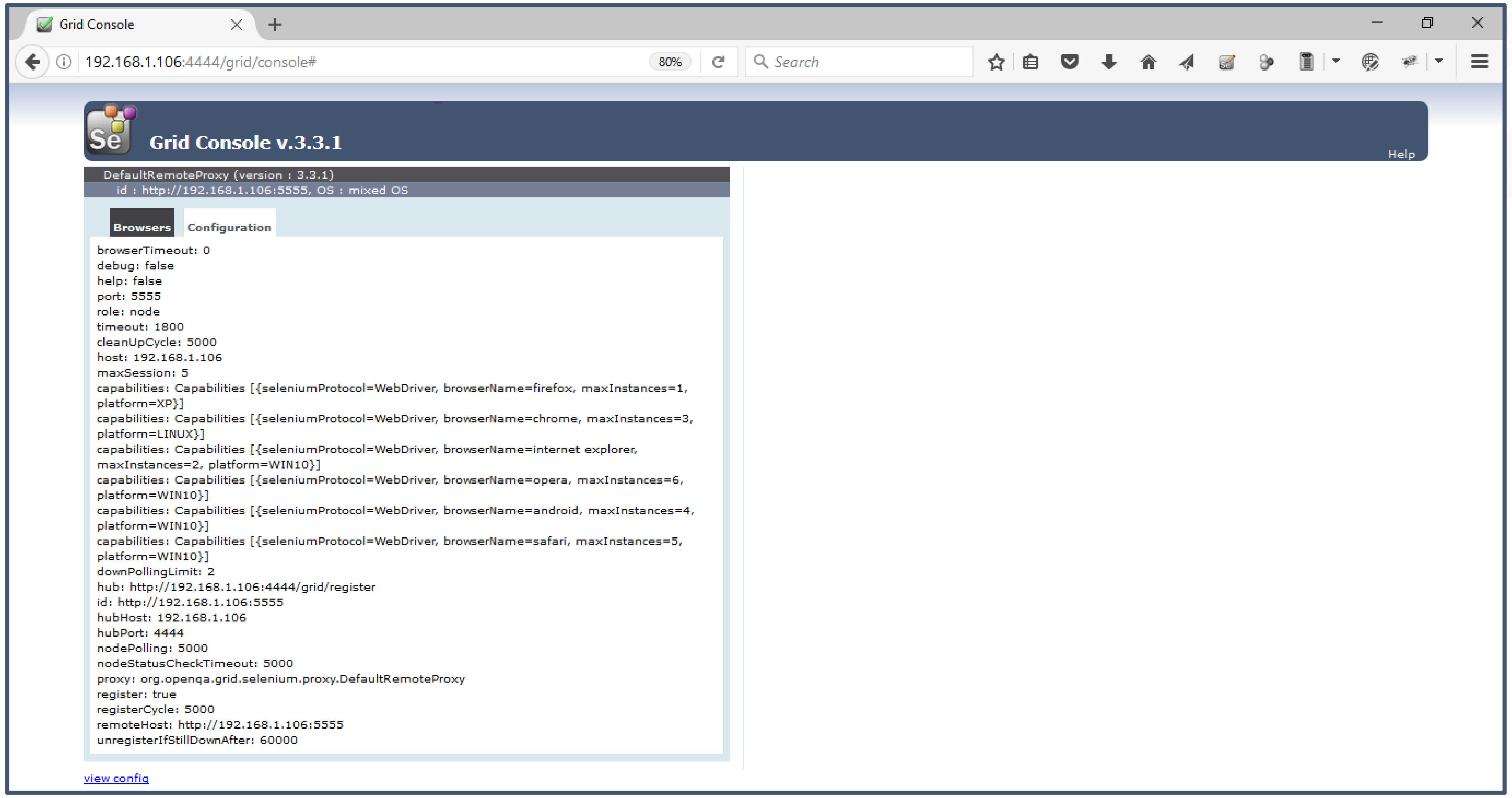
DefaultRemoteProxy (version : 3.3.1)
id : http://192.168.1.106:5555, OS : mixed OS

WebDriver
v:
v:
v:
v:
v:
v:

By default, node starts listening on port 5555.

Monitoring Node from Browser

The Grid Console—>Configuration View snapshot is as shown below:

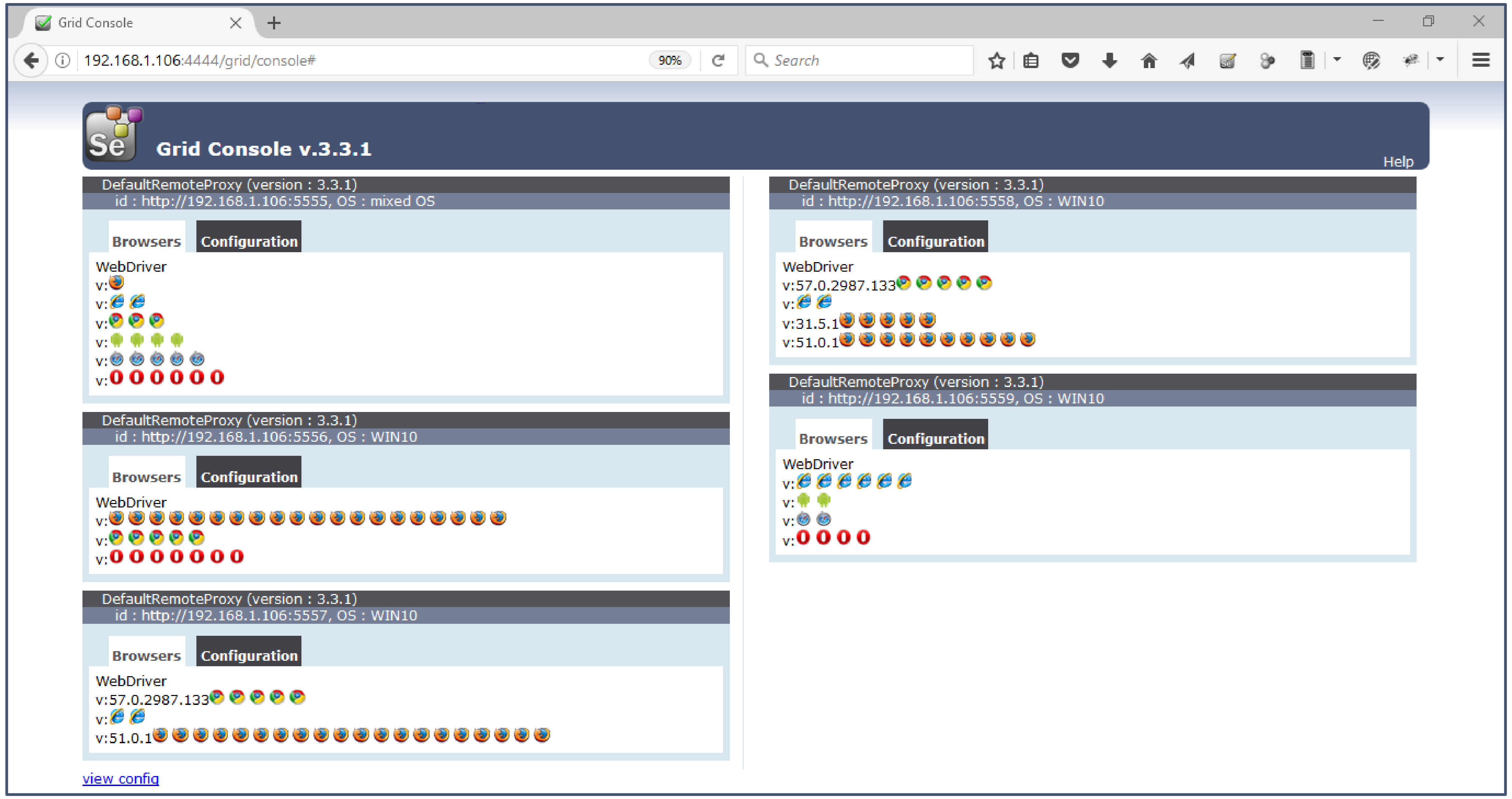


```
DefaultRemoteProxy (version : 3.3.1)
  id : http://192.168.1.106:5555, OS : mixed OS
```

```
browserTimeout: 0
debug: false
help: false
port: 5555
role: node
timeout: 1800
cleanUpCycle: 5000
host: 192.168.1.106
maxSession: 5
capabilities: Capabilities [{seleniumProtocol=WebDriver, browserName=firefox, maxInstances=1,
platform=XP}]
capabilities: Capabilities [{seleniumProtocol=WebDriver, browserName=chrome, maxInstances=3,
platform=LINUX}]
capabilities: Capabilities [{seleniumProtocol=WebDriver, browserName=internet explorer,
maxInstances=2, platform=WIN10}]
capabilities: Capabilities [{seleniumProtocol=WebDriver, browserName=opera, maxInstances=6,
platform=WIN10}]
capabilities: Capabilities [{seleniumProtocol=WebDriver, browserName=android, maxInstances=4,
platform=WIN10}]
capabilities: Capabilities [{seleniumProtocol=WebDriver, browserName=safari, maxInstances=5,
platform=WIN10}]
downPollingLimit: 2
hub: http://192.168.1.106:4444/grid/register
id: http://192.168.1.106:5555
hubHost: 192.168.1.106
hubPort: 4444
nodePolling: 5000
nodeStatusCheckTimeout: 5000
proxy: org.openqa.grid.selenium.proxy.DefaultRemoteProxy
register: true
registerCycle: 5000
remoteHost: http://192.168.1.106:5555
unregisterIfStillDownAfter: 60000
```

Monitoring Node from Browser

Snapshot of the grid with five nodes is shown below:



Grid Console

192.168.1.106:4444/grid/console#

Grid Console v.3.3.1

DefaultRemoteProxy (version : 3.3.1)
id : http://192.168.1.106:5555, OS : mixed OS

Browsers Configuration

WebDriver
v:
v:
v:
v:
v:
v:

DefaultRemoteProxy (version : 3.3.1)
id : http://192.168.1.106:5556, OS : WIN10

Browsers Configuration

WebDriver
v:
v:
v:

DefaultRemoteProxy (version : 3.3.1)
id : http://192.168.1.106:5557, OS : WIN10

Browsers Configuration

WebDriver
v:57.0.2987.133
v:
v:51.0.1

DefaultRemoteProxy (version : 3.3.1)
id : http://192.168.1.106:5558, OS : WIN10

Browsers Configuration

WebDriver
v:57.0.2987.133
v:
v:31.5.1
v:51.0.1

DefaultRemoteProxy (version : 3.3.1)
id : http://192.168.1.106:5559, OS : WIN10

Browsers Configuration

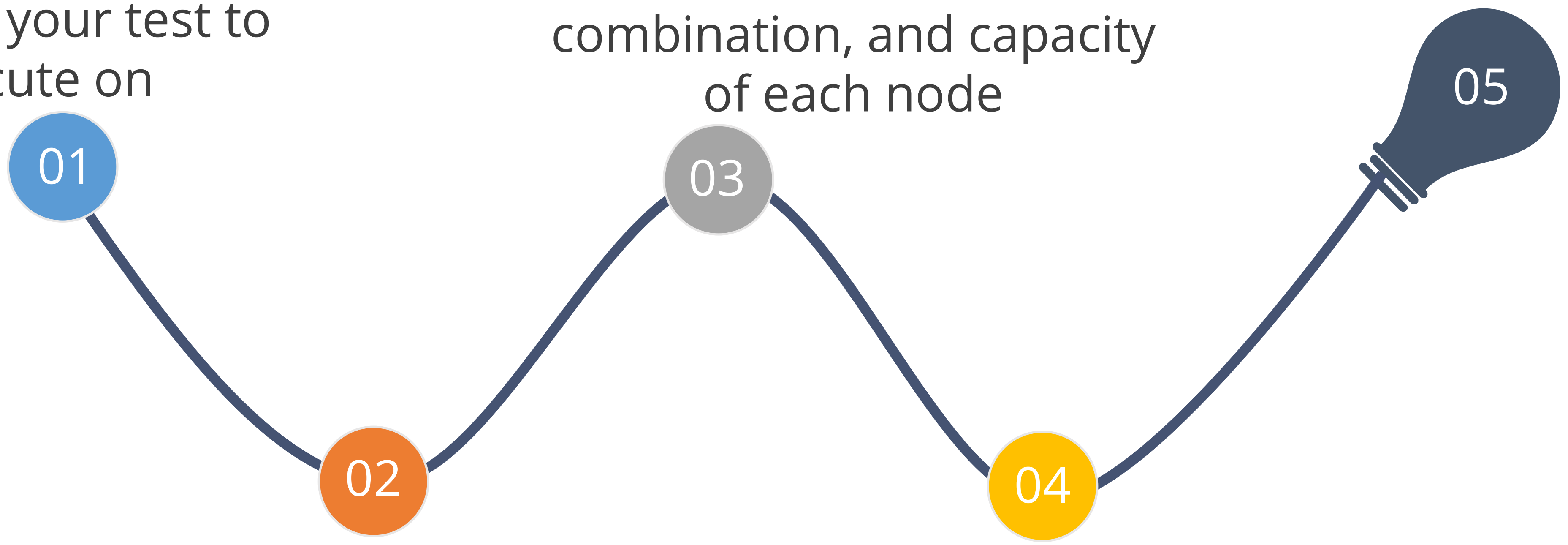
WebDriver
v:
v:
v:
v:

view config

Nodes mention browser versions in the same places

Configuring testng.xml

In Lesson 7—TestNG, you have seen how to pass parameters from testng.xml to Tests. For setting up the tests to execute on grid, you need to follow these steps:



01 Finalize the combination of platforms, browsers, and browser versions you want your test to execute on

02 Make "parallel" attribute of <suite> as "tests"

03 Make "thread-count" attribute of <suite> based on number of nodes, combination, and capacity of each node

04 Create one test for each combination in TestNG configuration file

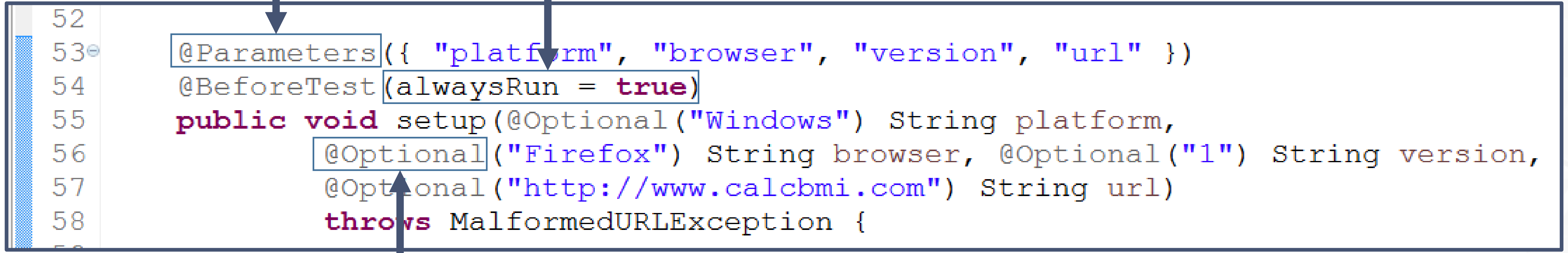
05 Declare parameters to be passed to TestNG Class file and provide necessary values

- Platform—OS you want your test to execute on
- Browser—Browser you want your test to execute on
- Version—Browser version
- Url—Test Environment on which the test should execute

Accepting TestNG Parameters

Use @Parameters annotation in your test class to receive platform and browser details in method parameters.

In this method, “alwaysRun” attribute is set to **true** as it is necessary to run this method.



```
@Parameters({ "platform", "browser", "version", "url" })
@BeforeTest(alwaysRun = true)
public void setup(@Optional("Windows") String platform,
        @Optional("Firefox") String browser, @Optional("1") String version,
        @Optional("http://www.calcbmi.com") String url)
        throws MalformedURLException {
```

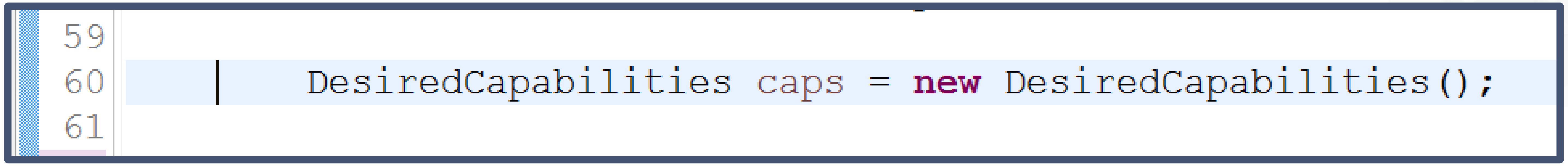
@Optional annotation ensures variables are given some default values, in case, TestNG configuration does not pass that particular parameter(s).

Configuring DesiredCapabilities

1. Creating object of DesiredCapabilities:

- You can communicate environment requirement to hub.
- You can set the browser, browser version, and platform on which you want to execute your test.

Based on the values set in DesiredCapabilities, hub identifies the appropriate node for execution.



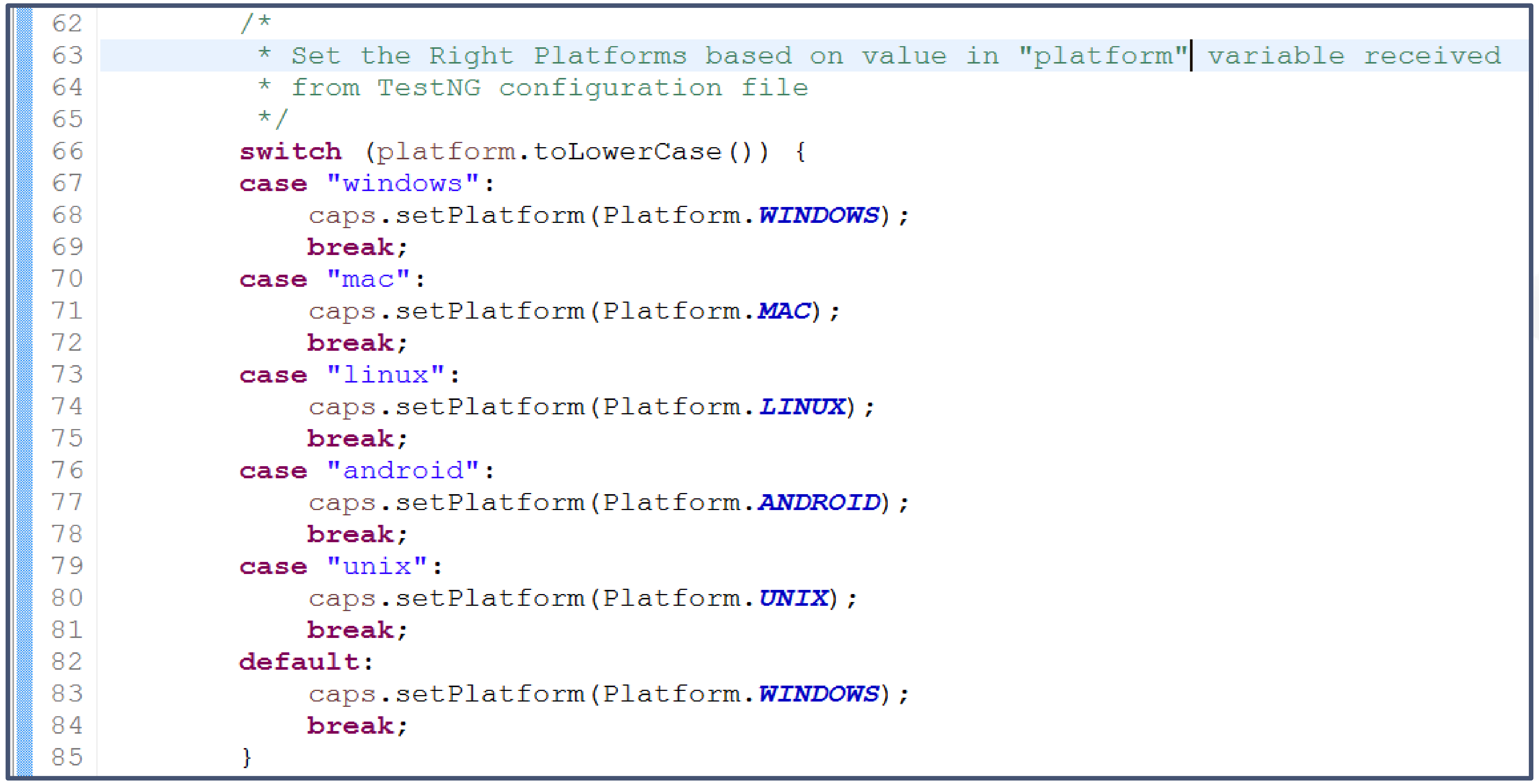
```java

        DesiredCapabilities caps = new DesiredCapabilities();

```

Configuring DesiredCapabilities

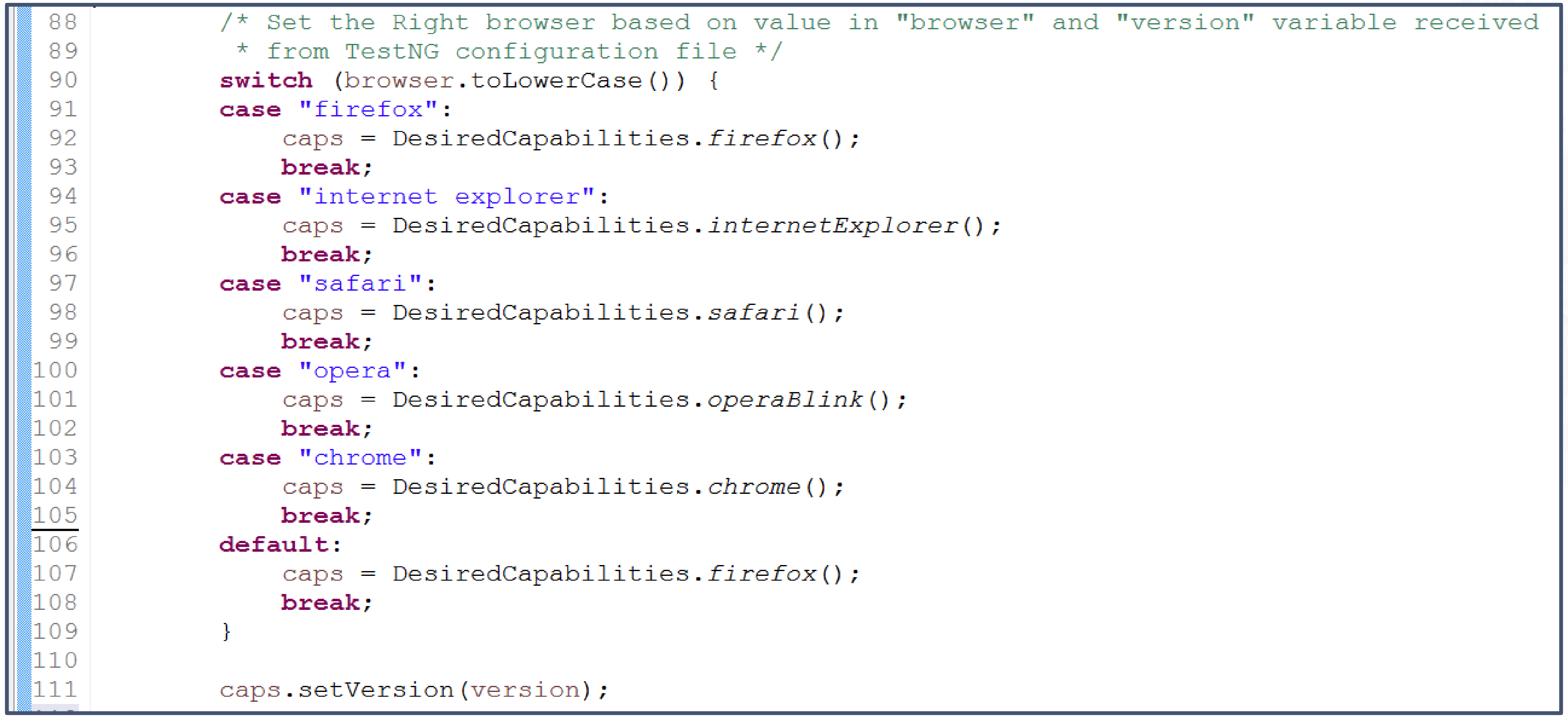
2. Based on value received in parameter “platform,” set the appropriate value of platform.



```
        /*
         * Set the Right Platforms based on value in "platform" variable received
         * from TestNG configuration file
         */
        switch (platform.toLowerCase()) {
        case "windows":
            caps.setPlatform(Platform.WINDOWS);
            break;
        case "mac":
            caps.setPlatform(Platform.MAC);
            break;
        case "linux":
            caps.setPlatform(Platform.LINUX);
            break;
        case "android":
            caps.setPlatform(Platform.ANDROID);
            break;
        case "unix":
            caps.setPlatform(Platform.UNIX);
            break;
        default:
            caps.setPlatform(Platform.WINDOWS);
            break;
        }
```

Configuring DesiredCapabilities

3. **Based on values received in parameters, browser and version, set the appropriate values in DesiredCapabilities.**

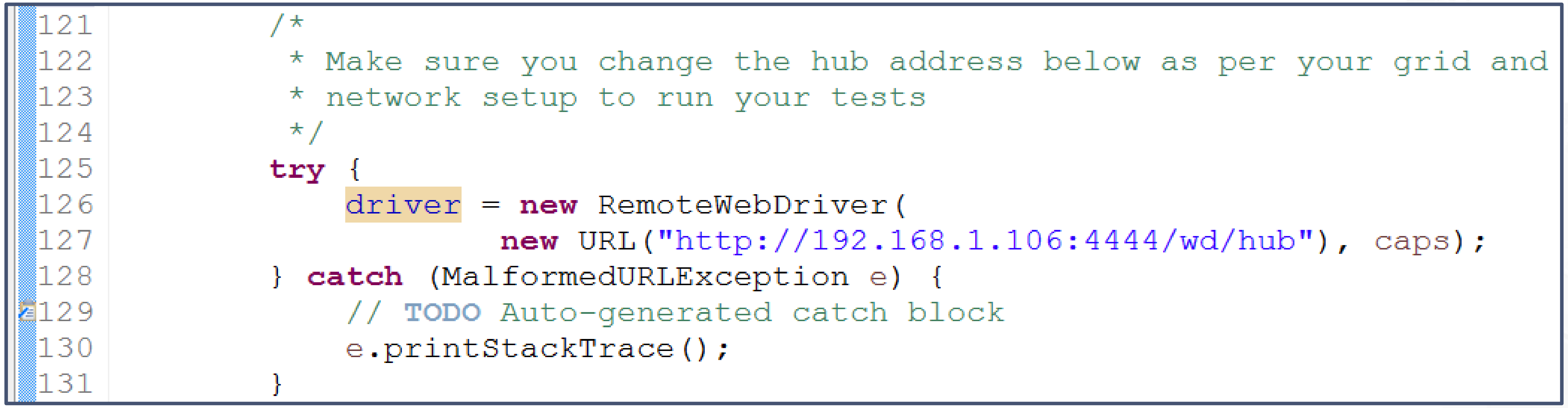


```
        /* Set the Right browser based on value in "browser" and "version" variable received
         * from TestNG configuration file */
        switch (browser.toLowerCase()) {
        case "firefox":
            caps = DesiredCapabilities.firefox();
            break;
        case "internet explorer":
            caps = DesiredCapabilities.internetExplorer();
            break;
        case "safari":
            caps = DesiredCapabilities.safari();
            break;
        case "opera":
            caps = DesiredCapabilities.operaBlink();
            break;
        case "chrome":
            caps = DesiredCapabilities.chrome();
            break;
        default:
            caps = DesiredCapabilities.firefox();
            break;
        }

        caps.setVersion(version);
```

Configuring DesiredCapabilities

4. Create an instance of RemoteWebDriver and pass Hub URL and DesiredCapabilities as parameters.



```
        /*
         * Make sure you change the hub address below as per your grid and
         * network setup to run your tests
         */
        try {
            driver = new RemoteWebDriver(
                    new URL("http://192.168.1.106:4444/wd/hub"), caps);
        } catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

Note: While running a test on Grid, there is no change required in the @Test that you have written.

Set Up Selenium Grid

Duration: 30 min.

Problem Statement:
Setup selenium grid which includes setup of hub and nodes.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to set up selenium grid:

1. Creates hubs using selenium grid
2. Creates nodes using selenium grid
3. Push the code to gitHub repositories

Grid Configuration Using JSON

Duration: 30 min.

Problem Statement:
Configure the grid using JSON.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to configure grid using JSON:

1. Configure the grid hub using JSON

2. Configure the grid nodes using JSON

3. Push the code to GitHub repositories

Running Tests on Selenium Grid

Duration: 40 min.

Problem Statement:
Run the test scripts on selenium grid.

Assisted Practice: Guidelines

Steps to run the scripts using selenium grid:

1. Run the tests on grid
2. Push the code to the GitHub repositories

Key Takeaways

- Cucumber is a tool based on behavior-driven development (BDD) methodology.
- TestNG is a testing framework for the Java programming language used for unit, functional, and end-to-end testing and integration.
- Jenkins is used to build and test software projects continuously to integrate changes.

Automate Your Web Application

Duration: 60 min.

Problem Statement:
Develop a project where you have to :

- Write test cases and generate reports using TestNG or ReportNG
- Integrate your application with Jenkins

LESSON-END PROJECT

Before the Next Class

Course(s):

1. Learn ELK stack 6.0
2. Cloud Basics and Web Hosting

You should be able to:

- Explain ELK stack
- Analyze data on Kibana dashboard
- Launch AWS EC2 instance and demonstrate AWS Cloudfront