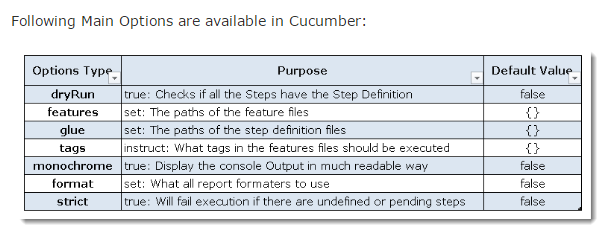
**Various CucumberOptions**



**How to catch the element into step definations passed from feature file with scenario outline.**

**\"([^\"]\*)\" is used**

**How to catch element into step definition passed from feature file say “username”**

**Need to find ()**

**1- General Selenium Webdriver Cucumber Interview Questions.**

Q-1: What is a test framework?

In general, a framework is an entity which binds several modules in a logical sequence to cover the end-to-end flows of an application. The objective of investing into a framework is to test a product which has a recurring roadmap and regular release cycle.

Q-2: What are the apparent advantages of a test framework?

Following are the possible benefits of using a test framework.

1- It reduces the complexity of using a variety of technologies inculcated in a product.

2- It organizes the unit and functional testing efforts of a developer and tester.

3- Provides early feedback on the quality of the code.

4- Helps in tracking test coverage as well as code coverage.

5- Results in easy debugging and reduces chances of errors.

Q-3: What type of tests Selenium can run?

1- You can use Selenium for the functional, regression, and load testing of the web-based applications.

2- You can employ this tool for doing the post-release validation.

3- Integrate it with the continuous integration tools like Jenkins, Hudson, QuickBuild or CruiseControl.

The above two questions were a little basic which sometimes we miss explaining during the interview. So, we thought it was worth starting with them.

Moving down, you can see all the questions below are specific to the Selenium Webdriver Cucumber interview.

2- Basic Level Selenium Webdriver Cucumber Interview Questions.

Q-4: What are the prereqs for building a Selenium Cucumber automation framework?

You might like to consider the following facts while creating a productive and scalable test framework.

1- Identify the type of application you are going to test. Is it a Web app, support mobile devices or runs on a desktop.

2- Would it require backend testing? e.g. Databases or SDK.

3- Decide on the input format. Is it static or dynamic?

4- Do you need to test the app for internationalization?

5- It must have a report which can help you trace a failure with minimum efforts.

6- It must support auto-generation of parametrization tests.

7- Have a config file to define any setup related settings or the global properties.

8- Apply abstraction at every level to separate the functionality.

If you follow the above rules, then you’ll land up with a product which is easy to maintain and free to scale.

Q-5: List down the advantages of using Selenium as a testing tool?

1- It’s an open source, so you save a lot on the cost side.

2- It gives you options to choose from a list of programming languages. e.g. Java, Python, C-Sharp, Ruby, and Python.

3- It offers easy and powerful dom-level testing.

4- You can use it in either of Agile or waterfall environment.

5- Easy integration with Jenkins, Bamboo, and some other notable CI tools.

Q-6: List down the mobile device which Selenium supports?

1- It supports Safari browser via a third-party driver. It is experimental and comes with limited functionality.

2- It provides an Android driver to run tests on its native mobile browser.

3- Intermediate Level Selenium Webdriver Cucumber Interview Questions.

Q-7: How to integrate Cucumber with Selenium Webdriver?

It’s the most obvious Selenium Webdriver Cucumber interview question which you must know. And it’s better if you give a step by step reply to the interviewer. It’ll leave a positive impression on him as you’ll show the depth of your knowledge.

1- Cucumber is a testing framework to run acceptance test cases. It creates scripts using the BDD approach.

2- It makes use of a feature file which describes the test cases in plain text format.

3- Here you write tests in simple English. And later use the Selenium Webdriver to run the test scripts.

4- To start Cucumber with Selenium, first of all, you require creating a Maven project in Eclipse.

5- In the Maven’s POM file, you add the Cucumber dependency which brings the support of annotations like the <Given>, <When>, and <Then> and many other.

XHTML

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-core</artifactId>

<version>1.1.5</version>

</dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-java</artifactId>

<version>1.1.5</version>

</dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-junit</artifactId>

<version>1.1.5</version>

</dependency>

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-core</artifactId>

<version>1.1.5</version>

</dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-java</artifactId>

<version>1.1.5</version>

</dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-junit</artifactId>

<version>1.1.5</version>

</dependency>

6- Similarly, you can introduce the Selenium dependency into the above project. Alternatively, you can download the latest version of Selenium standalone jar from their website. And then, add to your project as an external jar file.

If you want to do it via the POM file, then add the following entry.

XHTML

<dependency>

<groupid>org.seleniumhq.selenium</groupid>

<artifactid>selenium-java</artifactid>

<version>2.53.0</version>

</dependency>

1

2

3

4

5

<dependency>

<groupid>org.seleniumhq.selenium</groupid>

<artifactid>selenium-java</artifactid>

<version>2.53.0</version>

</dependency>

Q-8: Are there any readymade Selenium-Cucumber frameworks available?

Yes, there are a few we are listing down below. Though, we recommend building one of your own as it gives you more freedom.

1- Selenium-Cucumber framework for testing the web and android apps.

2- Another one is an acceptance testing framework using Cucumber and Selenium Webdriver.

💡 Restore self-confidence, run through online tests.

Online Selenium Webdriver Quiz to Practice Before Interview.

4- Advanced Level Selenium Webdriver Cucumber Interview Questions.

Now Let’s start to discuss some of the internal Selenium Webdriver Cucumber interview questions.

Q-9: What are the two files which you need to run a Cucumber test scenario?

If you want to execute a Cucumber test, then make sure it has the following two files.

1- A feature file.

2- A step definition file.

Q-10: What does a feature file contain?

A feature file in cucumber specifies parameters and conditions for executing the test code. It can combine any of the following.

1- A feature.

2- A user scenario.

3- The scenario outline.

4- A <Given> clause.

5- A <When> clause.

6- A <Then> clause.

Q-11: What is a profile in cucumber?

You can create Cucumber profiles to run a set of features and step definitions. Use the following command to execute a cucumber profile.

Shell

cucumber features -p <profile\_name>

#Example: cucumber features -p acceptance

1

2

3

cucumber features -p <profile\_name>

#Example: cucumber features -p acceptance

Q-12: What are before, after, beforeStep and afterStep hooks?

1- Before: executes before the feature file execution.

2- After: executes after the feature file execution.

3- BeforeStep: executes before the each step execution.

4- AfterStep: executes after the each step execution.

Q-13: What are cucumber tags? And why do we use them?

Cucumber tags help in filtering the scenarios. We can tag the scenarios and then run them based on tags.

1- We can add tags to scenarios with <@> symbol.

2- We can use the following command to run a cucumber tagged scenario.

Shell

cucumber features -t @<tag\_name>

#Example: cucumber features -t @test

1

2

3

cucumber features -t @<tag\_name>

#Example: cucumber features -t @test

💡 Read more interview questions and plan in advance.

Appium Mobile Testing Interview Questions.

5- Expert Level Selenium Webdriver Cucumber Interview Questions.

Q-14: What is the purpose of cucumber dry-run?

We use to compile the cucumber feature files and step definitions. If there occur any compilation errors, then it shows them on the console.

Q-15: Why do you use the scenario outline?

We use it to execute the same scenario with different test data.

Q-16: What if you don’t use the cucumber keywords in test steps?

Please note that it’s not mandatory to write keywords in test steps.

For example, we can build a test step like the one shown in the next line.

e.g.- We are testing using cucumber.

Q-17: List out some of the main differences between Jbehave and Cucumber?

However, the Cucumber and Jbehave share the same perspective, but there are few key differences.

1- Jbehave is Java-based and Cucumber is Ruby-based.

2- Jbehave is story-driven whereas the Cucumber is feature-driven.

Q-18: When would you use RSpec and when to use Cucumber?

1- RSpec is more successful in doing unit testing.

2- As you know that Cucumber is a behaviour-driven development tool. So you can use it for System and Integration testing.

Q-19: What are the steps to generate a report in Cucumber?

We run the following command to produce HTML reports.

Shell

cucumber <featurename>.feature --format html --out report.html --format pretty

1

cucumber <featurename>.feature --format html --out report.html --format pretty

Q-20: What is the right way to execute a specific scenario from the feature file?

We can select the target scenario from a feature file by providing its line number.

Shell

cucumber features/test.feature:10 --format html > testfeature.html

1

cucumber features/test.feature:10 --format html > testfeature.html

Notes:

Running Cucumber from command line

(without maven)

Cucumber lets you store and reuse commonly used cucumber command line arguments for a project in a cucumber.yml or cucumber.yaml file. This file must be in a .config subdirectory or config subdirectory of your current working directory.

## Defining Profiles

# config/cucumber.yml

##YAML Template

---

html\_report: --format progress --format html --out=features\_report.html

bvt: --tags @bvt

Defining a template requires a name and then the command-line options that you want to execute with this profile. The example above generates two profiles: the first, named html\_report, with a list of command-line options that specify new output formats and a second, named bvt which executes all features and scenarios [tagged](https://github.com/cucumber/cucumber/wiki/Tags) with @bvt.

## Executing Profiles

[user@system project] cucumber --profile html\_report

[user@system project] cucumber -p bvt

The execution of a profile simply requires the use of the flag --profile or -p.

During execution you can also specify additional parameters alongside the profile.

[user@system project] cucumber --profile html\_report --tags ~@wip

Even multiple profiles can be specified together. The following executes all the features and scenarios tagged as @bvt with the specified progress and html output.

[user@system project] cucumber -p html\_report -p bvt

## Default Profile

It is often the case that you will want to execute Cucumber with a particular profile a majority of the time. The Cucumber configuration file uses a default profile to provide this functionality. When you specify a default profile you are stating that Cucumber should apply this command-line options to an execution when you do not specify a profile.

Using the same example, perhaps we want the html\_report profile to be our default execution.

# config/cucumber.yml

##YAML Template

---

default: --profile html\_report --profile bvt

html\_report: --format progress --format html --out=features\_report.html

bvt: --tags @bvt

The default profile is a special profile that when present, is applied to the execution of Cucumber when you have not specified a profile.

[user@system project] cucumber

So now, by default, Cucumber is going to use both the bvt profile and html\_report profile testing all features and scenarios tagged as @bvt with the progress output and html output.

(with maven)

* Integrating Cucumber with Maven
* Running Cucumber from the Terminal
* Overriding options from the Terminal

*(For more resources related to this topic, see*[*here*](https://www.packtpub.com/books/content/running-cucumber#more)*.)*

Integrating Cucumber with Maven

Maven has a lot of advantages over other build tools, such as dependency management, lots of plugins and the convenience of running integration tests. So let's also integrate our framework with Maven. Maven will allow our test cases to be run in different flavors, such as from the Terminal, integrating with Jenkins, and parallel execution.

So how do we integrate with Maven? Let's find out in the next section.

Getting ready

I am assuming that we know the basics of Maven (the basics of Maven are out of the scope of this book). Follow the upcoming instructions to install Maven on your system and to create a sample Maven project.

1. We need to install Maven on our system first. So, follow instructions mentioned on the following blogs:

For Windows:

<http://www.mkyong.com/maven/how-to-install-maven-in-windows/>

For Mac:

<http://www.mkyong.com/maven/install-maven-on-mac-osx/>

1. We can also install the Maven Eclipse plugin by following the instructions mentioned on this blog:

<http://theopentutorials.com/tutorials/eclipse/installing-m2eclipse-maven-plugin-for-eclipse/>.

1. To import a Maven project into Eclipse, follow the instructions on this blog:

<http://www.tutorialspoint.com/maven/maven_eclispe_ide.htm>.

How to do it…

Since it is a Maven project, we are going to change the pom.xml file to add the Cucumber dependencies.

1. First we are going to declare some custom properties which will be used by us in managing the dependency version:
2. <properties>
3. <junit.version>4.11</junit.version>
4. <cucumber.version>1.2.2</cucumber.version>
5. <selenium.version>2.45.0</selenium.version>
6. <maven.compiler.version>2.3.2</maven.compiler.version>

</properties>

1. Now, we are going to add the dependency for Cucumber-JVM with scope as test:
2. <!—- Cucumber-java-->
3. <dependency>
4. <groupId>info.cukes</groupId>
5. <artifactId>cucumber-java</artifactId>
6. <version>${cucumber.version}</version>
7. <scope>test</scope>

</dependency>

1. Now we need to add the dependency for Cucumber-JUnit with scope as test.
2. <!-— Cucumber-JUnit -->
3. <dependency>
4. <groupId>info.cukes</groupId>
5. <artifactId>cucumber-junit</artifactId>
6. <version>${cucumber.version}</version>
7. <scope>test</scope>

</dependency>

That's it! We have integrated Cucumber and Maven.

How it works…

By following these Steps, we have created a Maven project and added the Cucumber-Java dependency. At the moment, this project only has a pom.xml file, but this project can be used for adding different modules such as Feature files and Step Definitions.

The advantage of using properties is that we are making sure that the dependency version is declared at one place in the pom.xml file. Otherwise, we declare a dependency at multiple places and may end up with a discrepancy in the dependency version.

The Cucumber-Java dependency is the main dependency necessary for the different building blocks of Cucumber. The Cucumber-JUnit dependency is for Cucumber JUnit Runner, which we use in running Cucumber test cases.

Running Cucumber from the Terminal

Now we have integrated Cucumber with Maven, running Cucumber from the Terminal will not be a problem. Running any test framework from the Terminal has its own advantages, such as overriding the run configurations mentioned in the code.

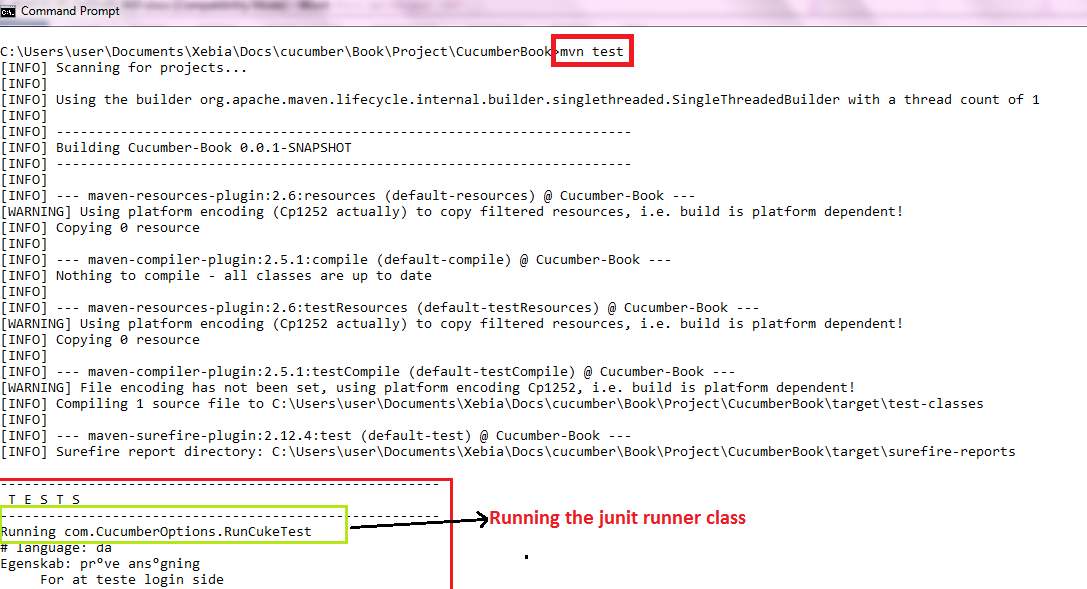
So how do we run Cucumber test cases from the Terminal? Let's find out in our next section.

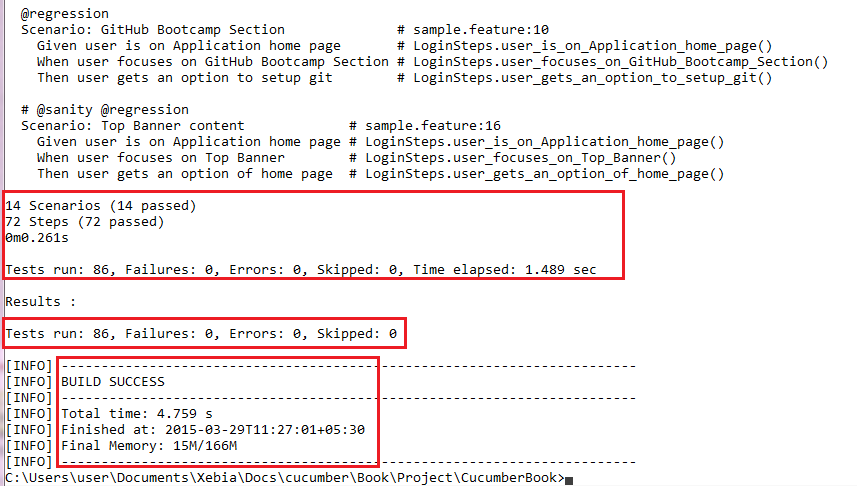
How to do it…

1. Open the command prompt and cd until the project root directory.
2. First, let's run all the Cucumber Scenarios from the command prompt. Since it's a Maven project and we have added Cucumber in test scope dependency and all features are also added in test packages, run the following command in the command prompt:

**mvn test**

This is the output:



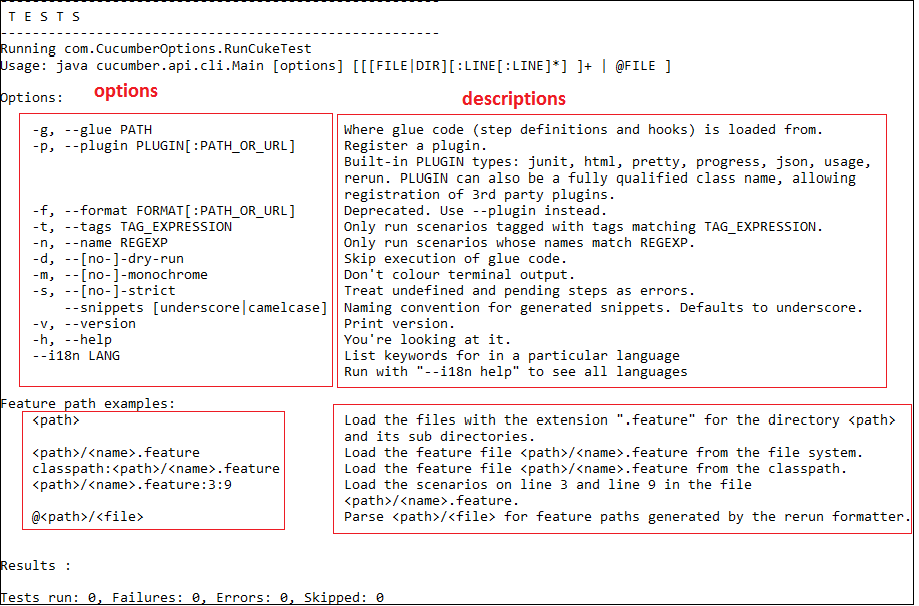


1. The previous command runs everything as mentioned in the JUnit Runner class. However, if we want to override the configurations mentioned in the Runner, then we need to use following command:

**mvn test –DCucumber.options="<<OPTIONS>>"**

1. If you need help on these Cucumber options, then enter the following command in the command prompt and look at the output:

mvn test -Dcucumber.options="--help"

This is the output:  


How it works…

mvn test runs Cucumber Features using Cucumber's JUnit Runner. The @RunWith (Cucumber.class) annotation on the RunCukesTest class tells JUnit to kick off Cucumber. The Cucumber runtime parses the command-line options to know what Feature to run, where the Glue Code lives, what plugins to use, and so on. When you use the JUnit Runner, these options are generated from the @CucumberOptions annotation on your test.

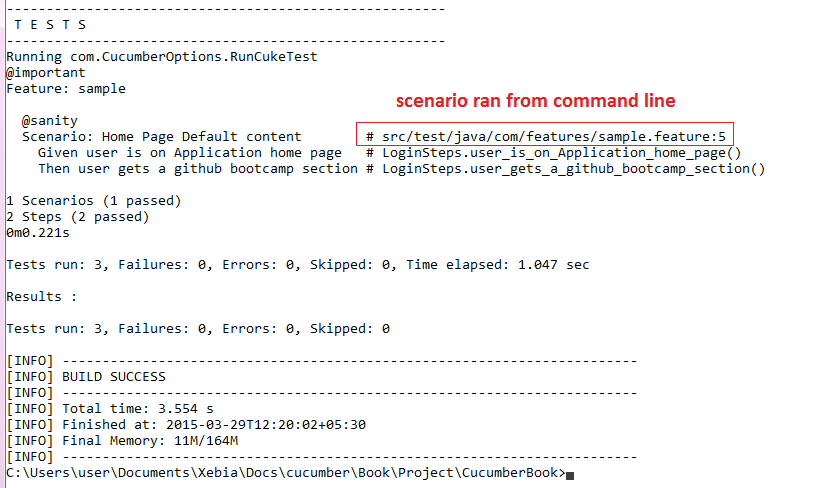
Overriding Options from the Terminal

When it is necessary to override the options mentioned in the JUnit Runner, then we need Dcucumber.options from the Terminal. Let's look at some of the practical examples.

How to do it…

1. If we want to run a Scenario by specifying the filesystem path, run the following command and look at the output:

**mvn test -Dcucumber.options= "src/test/java/com/features/sample.feature:5"**

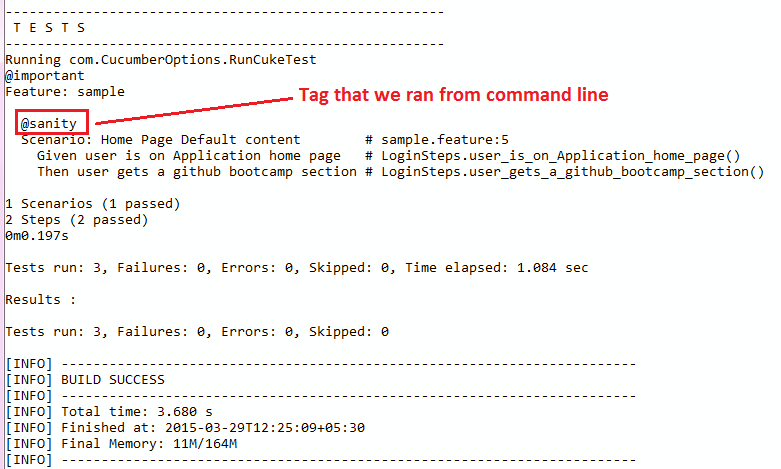


In the preceding code, "5" is the Feature file line number where a Scenario starts.

1. If we want to run the test cases using Tags, then we run the following command and notice the output:

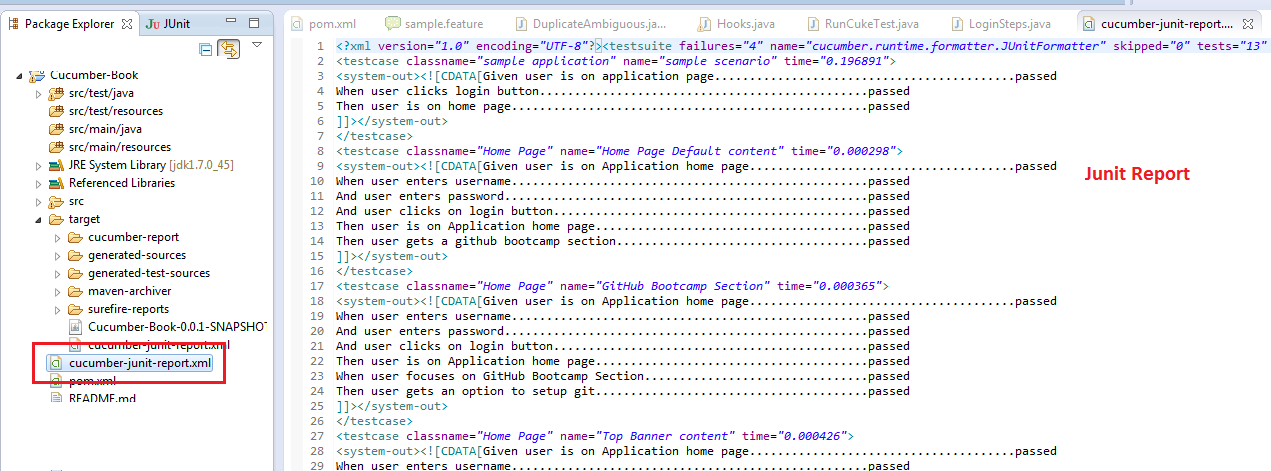
**mvn test -Dcucumber.options="--tags @sanity"**

The following is the output of the preceding command:



1. If we want to generate a different report, then we can use the following command and see the JUnit report generate at the location mentioned:

mvn test -Dcucumber.options= "--plugin junit:target/cucumber-junit-report.xml"



How it works…

When you override the options with -Dcucumber.options, you will completely override whatever options are hardcoded in your @CucumberOptions. There is one exception to this rule, and that is the --plugin option. This will not override, but instead, it will add a plugin.

=======================

How to Run Cucumber Tests in paralle using maven and cucumber combinations

How to Run only failed test cases