CUCUMBER:

What is Cucumber? Why is it used?

1. Cucumber is a testing tool based on Behaviour Driven Development (BDD) framework.
2. It is used to run functional tests written in plain text and develop test cases for software functionality. It plays a supporting role in automated testing.
3. In other words, we can say that "Cucumber is a software tool used by the testers to develop test cases for the testing of behaviour of the software."

2) In which language is Cucumber software written?

Cucumber software is written in a Ruby programming language.

3) What is the main aim of the behaviour Driven Development (BDD) framework?

1. The main aim of the behaviour Driven Development framework is to make various project roles such as Business Analysts, Quality Assurance, Developers, etc., understand the application without diving deep into the technical aspects.

4) What language is used by the Cucumber tool?

1. The Cucumber tool uses the Gherkin language, a simple English representation of the application behaviour.
2. The Gherkin language uses several keywords to describe the behaviour of applications such as Feature, Scenario, Scenario Outline, Given, When, Then, etc.

5) What are the two files required to execute a Cucumber test scenario?

Following are the two files required to execute a Cucumber test scenario:

* Features
* Step Definition

6) What do you understand by a feature file?

1. A feature file is used to provide a high-level description of an Application Under Test (AUT).
2. The first line of the feature file must start with the keyword 'Feature' followed by the description of the application under test.
3. A feature file may include multiple scenarios within the same file, and the extension of the feature file must be ".feature".

7) What are the various keywords used in the Cucumber tool for writing a scenario?

Following are the keywords that are used for writing a scenario in the Cucumber tool:

* Given
* When
* Then
* And

8) What is the use of the Background keyword in Cucumber?

1. In the Cucumber tool, the Background keyword is used to group multiple given statements into a single group.
2. The keyword is mostly used when the same set of given statements are repeated in each scenario of the feature file.

9) What is the usage of a Scenario Outline in the Cucumber tool?

1. In Cucumber, a Scenario outline is used as a parameter of scenarios.
2. This is used when the same scenario needs to be executed for multiple sets of data; however, the test steps remain the same.
3. Scenario Outline must be followed by the keyword 'Examples', which specify the set of values for each parameter.

10) What do you understand by the term step definition in Cucumber?

In Cucumber, a step definition is the actual code implementation of the feature mentioned in the feature file.

11) Which programming languages are supported by Cucumber?

1. The Cucumber tool supports multiple programming languages such as Java, .Net, Ruby, etc.
2. You can also integrate this tool easily with multiple tools such as Selenium, Capybara, etc.

12) What are the differences between Jbehave and Cucumber?

Although Cucumber and Jbehave are designed for the same purpose, the most distinctive difference between them is that Jbehave is based on stories while Cucumber is based on features.

13) What do you understand by regular expressions?

1. A regular expression is a pattern used to describe a certain amount of text.
2. The most basic regular expression consists of a single literal character.

14) What software is used to run a Cucumber Web Test case?

Following software is used to run a Cucumber Web Test case:

* Cucumber
* Ruby and its Development Kit
* IDE like ActiveState
* Watir ( To simulate browser)
* Ansicon and RSpec (if required)

15) What are the most significant advantages of the Cucumber framework?

Following is the list of advantages of the Cucumber framework that makes Cucumber an ideal choice for rapidly evolving agile methodology in today's corporate world:

Advertisement

* Cucumber is an open-source and free-to-use tool.
* It is easy to use. Even non-technical users can also understand the scenarios because of its plain text representation.
* It bridges the communication gap between various project stakeholders such as Business Analysts, Developers, and Quality Assurance personnel.
* With the Cucumber tool, we can develop automation test cases that are easier to maintain and understand.
* It is easy to integrate with other tools such as Selenium and Capybara.

16) What do you understand by test harness in Cucumber?

1. In Cucumber, the test harness allows for separating responsibility between setting up the context and interacting with the browser, and cleaning up the step definition files.
2. It collects stubs, drivers, and other supporting tools required to automate test execution in testing.

17) What is the difference between RSpec and Cucumber? When should we use RSpec and when to use Cucumber?

RSpec and Cucumber both are the example of testing frameworks. RSpec uses traditional Unit Testing. It means it uses testing a class or part of the application in isolation from the rest of the application. So your model does what your model is supposed to do, the controller does what it is supposed to do, etc. RSpec and Cucumber both are used for Acceptance Testing, also called ATDD, BDD, etc.

**Difference between RSpec and Cucumber**

* The main difference between RSpec and Cucumber is the business readability factor.
* RSpec is mainly used for Unit Testing. On the other hand, Cucumber is mainly used for behaviour-driven development. We can also use it for System and Integration Testing.
* In Cucumber, the specifications or features are separate from the test code, so the product owners can provide or review the specification without going through the code. These are the .feature files that you make in Cucumber.
* RSpec also has a similar mechanism, but instead of describing a step with a Describe or Context, it uses the business specification to execute that statement. This approach is a little easier for developers to work with but a bit harder for non-technical guys.

**Which should we use?**

* For a core developer, it is the best choice to use RSpec. It is easier to understand for a technical person and offers a few advantages in keeping things scoped and under control because you don't have to mess up with RegExs for test steps.
* If you are building this for a client, you should choose Cucumber for Acceptance Testing and use RSpec for Unit Testing.

18) What is Selenium?

1. Selenium is a web browser automation tool widely used for Functional Testing of web-based applications.
2. Selenium supports different programming languages such as Java, Python, Ruby, C#, etc.

19) What is the difference between Selenium and Cucumber?

Selenium and Cucumber are both open-source testing tools, and both are used for functional testing. But there are some differences between them.

Following are some critical differences between Selenium and Cucumber:

* Selenium is a web browser automation tool for web apps, while Cucumber is an automation tool for behaviour-driven development that can be used with Selenium (or Appium).
* Selenium is used for automated UI testing, while Cucumber is used for acceptance testing.
* Selenium is preferred by technical teams (SDETs/programmers), while Cucumber is typically preferred by non-technical teams (business stakeholders and testers).
* Selenium can work independently of Cucumber. Cucumber depends on Selenium or Appium for step-definition implementation.
* In Selenium, the script creation is complex, while Cucumber is simpler than Selenium.

20) Why we have to use Cucumber with Selenium?

1. Cucumber and Selenium are both testing frameworks and prevalent technologies.
2. Many organizations use Selenium for functional testing.
3. Along with Selenium, these organizations integrate Cucumber with Selenium as Cucumber makes it easy to read and understand the application flow.
4. The most significant benefit of using Cucumber with Selenium is that it facilitates developers to write test cases in simple feature files easily understood by managers, non-technical stakeholders, and business analysts.
5. It provides the facility to write tests in a human-readable language called Gherkin.
6. The Selenium-Cucumber framework supports programming languages such as Java, .NET, PHP, Python, Perl, etc.

21) What do you understand by the Step Definition?

1. Step definition is used to map the Test Case Steps in the feature files to code.
2. It executes the steps on Application under Test and checks the results.
3. To execute step definition, it must match the given component in a feature.

22) What are the maximum numbers of scenarios that we can include in the feature file?

1. In Cucumber, a feature file can contain a maximum of 10 scenarios.
2. This number can vary from project to project and from one organization to another organization.
3. It is the best practice to limit the number of scenarios included in the feature file.

23) What is the purpose of the behaviour-driven development (BDD) methodology in the real world?

1. Behavior Driven Development or BDD is a methodology used to understand the functionality of an application in the simple plain text representation.
2. The primary purpose of the Behavior Driven Development framework is to make various project roles such as Business Analysts, Quality Assurance, Developers, Support Teams understand the application without diving deep into the technical aspects.

24) What do you understand by TDD, and what are the different processes used in TDD?

1. Test-Driven Development (TDD) is a software technique where developers write test cases before the code itself.
2. Though it can extend development time, TDD improves code quality, flexibility, and reusability, and can achieve 90-100% test coverage.
3. The main challenge for developers is creating tests before the code.

**Following is the list of simple 6 step process used by TDD methodology:**

1. **First, write the test case:** You have to write an automated test case according to your requirements.
2. **Run all the test cases:** Now, run these automated test cases on the currently developed code.
3. **Develop the code for that test case:** In this process, you must write the code to make that test case work as expected if the test case fails.
4. **Run test cases again:** Now, you have to rerun the test cases and check if all the test cases developed so far are implemented.
5. **Refactor your code:** This is an optional step. But, it is advised to refactor your code to make it more readable and reusable. That's why it is essential.
6. **Repeat steps 1- 5 for new test cases:** This is the last step. Here, you have to repeat the cycle for the other test cases until all the test cases are implemented.

25) What are the similarities between BDD and TDD?

1. TDD stands for Test-Driven Development, and BDD stands for Behavior Driven Development. Both are two software development techniques.
2. BDD and TDD are both very similar as they are both testing strategies for a software application.
3. In both cases, the developers have to write the test before writing the code to pass the test.
4. The second main similarity between them is in both cases; the tests can be used as part of an automated testing framework to prevent bugs.

NOTE : The major difference between TDD (Test-Driven Development) and BDD (Behavior-Driven Development) is their focus. TDD focuses on testing individual functions or units of code, while BDD focuses on testing user behaviours and system interactions in a more understandable, business-oriented language.

27) What do you understand by cucumber dry run?

1. A Cucumber dry run is used to compile feature files and step definitions to check for any compilation errors without actually executing the tests.
2. If errors are found, they will be displayed in the output during the dry run.

28) What do you understand by the TestRunner class in the Cucumber testing approach? Explain with example.

In the Cucumber testing approach, the TestRunner class provides the link between the feature file and the step definition file.

The TestRunner class is generally an empty class with no class definition.

**Example of a TestRunner class in Cucumber:**

1. Package com.sample.TestRunner
2. importorg.junit.runner.RunWith;
3. importcucumber.api.CucumberOptions;
4. importcucumber.api.junit.Cucumber;
5. @RunWith(Cucumber.**class**)
6. @CucumberOptions(features="Features",glue={"StepDefinition"})
7. **public** **class** Runner
8. {
9. }

29) Should we write code within the TestRunner class?

1. It is advised not to write code under the TestRunner class.
2. It should include the tags @RunWith and @CucumberOptions.

30) What is the starting point of execution for feature files?

When Cucumber is integrated with Selenium, the starting point of execution must be from the TestRunner class.

31) How can you use the Options tag in the Cucumber framework?

In the Cucumber framework, the Options tag is a part of the TestRunner file and comes in the form of an annotation called @CucumberOptions. It contains two parameters feature and glue.

* **Feature parameter:** The feature parameter is used to specify the path of the feature file.
* **Glue parameter:** The glue parameter is used to specify the path of the step definitixon file.

**See the code implementation of TestRunner file with Option tag:**

**import** org.junit.runner.RunWith;

**import** cucumber.api.CucumberOptions;

**import** cucumber.api.junit.Cucumber;

@RunWith (Cucumber.**class**)

@CucumberOptions (

features = "src/test/java/features ",

glue = {"stepDefinitions"}

)

**public** **class** TestRunner {

}

We have to import org.junit.runner.RunWith for the @RunWith annotation and cucumber.api.CucumberOptions for the @CucumberOptions annotation.

32) What is the use of features property under the Cucumber Options tag?

In the Cucumber framework, the features property is used to identify the location of the feature files.

33) What is the use of glue property under the Cucumber Options tag?

The Glue property is used to facilitate the Cucumber framework to identify the location of step definition files.

34) What are the two build management tools that can be integrated with Cucumber?

Following are the two build management tools that can be integrated with Cucumber:

* Gradle
* Maven

35) What is the use of hooks in the Cucumber framework?

In the Cucumber framework, the hooks are used to control the flow of the program and optimize lines of code. A block of code tagged with hooks in Cucumber can run before or post a scenario with the help of @Before and @After annotations.

Some scenarios may require certain preconditions for execution, such as launching the application, establishing a database connection, configuring the test data, and so on. Also, certain postconditions should be executed, such as terminating database connection, closing the browser, refreshing test data, application log out, and so on. All these conditions are handled in Cucumber with the help of the hooks. The @Before hook executes before the actual scenario, and the @After hook executes after the actual scenario even if the test corresponding to the actual scenario fails.

For the @Before annotation, we have to import cucumber.api.java.en.Before and for the @After annotation, we have to import cucumber.api.java.en.After.

**See the code implementation with hooks:**

**The feature file:** In this feature file, we have included two scenarios:

**Feature:** Invoice Generation.

**Scenario 1:** Verify the invoice generates in pdf format.

**Given** User navigates to Invoice Page.

**Then** User should be able to open the invoice in pdf format.

**Scenario2:** Verify the invoice fields.

**Given** User opens the invoice.

**Then** User should be able to see the payment amount in invoice.

**See the step definition file:**

1. **import** cucumber.api.java.en.Given;
2. **import** cucumber.api.java.en.Then;
3. **import** cucumber.api.java.en.Before;
4. **import** cucumber.api.java.en.After;
5. **public** **class** InvoiceValidation {
6. @Given ("^User navigates to Invoice Page$")
7. **public** **void** navigate\_to\_invoice () {
8. System.out.println ("Navigation to Invoice Page");
9. }
10. @Then ("^User should be able to open the invoice in pdf format$")
11. **public** **void** open\_invoice\_pdf () {
12. System.out.println ("Invoice opened in pdf format ");
13. }
14. @Given ("^User opens the invoice$")
15. **public** **void** open\_invoice () {
16. System.out.println ("Invoice is opened");
17. }
18. @Then ("^User should be able to see the payment amount in invoice$")
19. **public** **void** verify\_payment\_invoice () {
20. System.out.println ("Invoice contains payment amount");
21. }
22. @Before
23. **public** **void** precondition () {
24. System.out.println ("The method runs before all scenarios");
25. }
26. @After
27. **public** **void** postcondition () {
28. System.out.println ("The method runs after all scenarios ");
29. }
30. }

In the above example, you can see that we have used two scenarios, the test method precondition with @Before annotation and the test method postcondition with @After annotation.