**Automation Testing Using Cucumber Tool and Selenium**

In last Selenium module we introduced you to Selenium Grid which is **a *distributed test execution* environment to speed up the execution of a test pass**.

Now at the end of this comprehensive Selenium training series we are learning advanced Selenium testing and related concepts.

In this and the next module we will be introducing you to the**Cucumber – a Behavior Driven Development (BDD) framework which is used with Selenium for performing acceptance testing.**

### ****Cucumber**** ****Introduction:****

Cucumber is a tool based on Behavior Driven Development (BDD) framework which is used to write acceptance tests for web application. It allows automation of functional validation in easily readable and understandable format (like plain English) to Business Analysts, Developers, Testers, etc.

Cucumber feature files can serve as a good document for all. There are many other tools like JBehave which also support BDD framework. Initially Cucumber was implemented in Ruby and then extended to Java framework. Both the tools support native JUnit.

Behavior Driven Development is extension of Test Driven Development and it is used to test the system rather than testing the particular piece of code. We will discuss more about the BDD and style of writing BDD tests.

Cucumber can be used along with Selenium, Watir, and Capybara etc. Cucumber supports many other languages like Perl, PHP, Python, .Net etc. In this module we will concentrate on Cucumber with Java as a language.

### ****Cucumber Basics:****

In order to understand cucumber we need to know all the features of cucumber and its usage.

**#1) Feature Files:**

Feature files are essential part of cucumber which is used to write test automation steps or acceptance tests. This can be used as live document. The steps are the application specification. All the feature files ends with .feature extension.

**Sample feature file:**

**Feature**: Login Functionality Feature

In order to ensure Login Functionality works,  
I want to run the cucumber test to verify it is working

**Scenario**: Login Functionality

**Given** user navigates to SOFTWARETETINGHELP.COM  
**When** user logs in using Username as “USER” and Password “PASSWORD”  
**Then** login should be successful

**Scenario**: Login Functionality

**Given** user navigates to SOFTWARETETINGHELP.COM  
**When** user logs in using Username as “USER1” and Password “PASSWORD1”  
**Then** error message should be thrown

**#2) Feature:**

**T**his gives information about the high level business functionality (Refer to previous example) and the purpose of Application under test. Everybody should be able to understand the intent of feature file by reading the first Feature step. This part is basically kept brief.

**#3) Scenario:**

Basically a scenario represents a particular functionality which is under test. By seeing the scenario user should be able to understand the intent behind the scenario and what the test is all about. Each scenario should follow given, when and then format. This language is called as “gherkin”.

1. **Given:**As mentioned above, given specifies the pre-conditions. It is basically a known state.
2. **When**: This is used when some action is to be performed. As in above example we have seen when the user tries to log in using username and password, it becomes an action**.**
3. **Then:**The expected outcome or result should be placed here. For Instance: verify the login is successful, successful page navigation.
4. **Background:**Whenever any step is required to perform in each scenario then those steps needs to be placed in Background. For Instance: If user needs to clear database before each scenario then those steps can be put in background.
5. **And**: And is used to combine two or more same type of action.

**Example:**

**Feature**: Login Functionality Feature

**Scenario**: Login Functionality  
**Given** user navigates to SOFTWARETETINGHELP.COM  
**When** user logs in using Username as “USER”  
**And**password as “password”  
**Then** login should be successful  
**And** Home page should be displayed

**Example of Background:**

**Background:**

**Given** user logged in as databases administrator  
**And** all the junk values are cleared

**#4) Scenario Outline:**

Scenario outlines are used when same test has to be performed with different data set. Let’s take the same example. We have to test login functionality with multiple different set of username and password.

**Feature**: Login Functionality Feature

In order to ensure Login Functionality works,  
I want to run the cucumber test to verify it is working

**Scenario Outline**: Login Functionality

**Given** user navigates to SOFTWARETESTINGHELP.COM  
**When** user logs in using Username as <**username**> and Password <**password**>  
**Then** login should be successful

**Examples:**  
|username         |password          |  
|Tom                     |password1        |  
|Harry                   |password2        |  
|Jerry                    |password3        |

**Note:**

1. As shown in above example column names are passed as parameter to **When** statement.
2. In place of Scenario, you have to use Scenario Outline.
3. Examples are used to pass different arguments in tabular format. Vertical pipes are used to separate two different columns. Example can contain many different columns.

**#5) Tags:**

Cucumber by default runs all scenarios in all the feature files. In real time projects there could be hundreds of feature file which are not required to run at all times.

**For instance**: Feature files related to smoke test need not run all the time. So if you mention a tag as smokeTest in each feature file which is related to smoke test and run cucumber test with @SmokeTest tag . Cucumber will run only those feature files specific to given tags. Please follow the below example. You can specify multiple tags in one feature file.

**Example of use of single tags:**

**@SmokeTest**

------------

**Feature**: Login Functionality Feature

In order to ensure Login Functionality works,  
I want to run the cucumber test to verify it is working

**Scenario Outline**: Login Functionality

**Given** user navigates to SOFTWARETESTINGHELP.COM  
**When** user logs in using Username as <**username**> and Password <**password**>  
**Then** login should be successful

**Examples:**  
|username         |password          |  
|Tom     |password1        |  
|Harry   |password2        |  
|Jerry    |password3        |

**Example of use of multiple tags:**

As shown in below example same feature file can be used for smoke test scenarios as well as for login test scenario. When you intend to run your script for smoke test then use @SmokeTest. Similarly when you want your script to run for Login test use @LoginTest tag.

Any number of tags can be mentioned for a feature file as well as for scenario.

**@SmokeTest @LoginTest**

**Feature**: Login Functionality Feature

In order to ensure Login Functionality works,  
I want to run the cucumber test to verify it is working

**Scenario Outline**: Login Functionality

**Given** user navigates to SOFTWARETETINGHELP.COM  
**When** user logs in using Username as <username> and Password <password>  
**Then** login should be successful

**Examples:**  
|username         |password          |  
|Tom     |password1        |  
|Harry   |password2        |  
|Jerry    |password3        |

Similarly you can specify tags to run specific scenario in a feature file. Please check below example to run specific scenario.

**Feature**: Login Functionality Feature

In order to ensure Login Functionality works,  
I want to run the cucumber test to verify it is working

@positiveScenario  
**Scenario**: Login Functionality

**Given** user navigates to SOFTWARETETINGHELP.COM  
**When** user logs in using Username as “USER” and Password “PASSWORD”  
**Then** login should be successful

@negaviveScenario  
**Scenario**: Login Functionality

**Given** user navigates to SOFTWARETETINGHELP.COM  
**When** user logs in using Username as “USER1” and Password “PASSWORD1”  
**Then** error message should throw

**#6) Junit Runner:**

To run the specific feature file cucumber uses standard Junit Runner and specify tags in @Cucumber. Options. Multiple tags can be given by using comma separate. Here you can specify the path of the report and type of report you want to generate.

**Example of Junit Runner:**

|  |  |  |
| --- | --- | --- |
| 1 | import cucumber.api.junit.Cucumber;</pre> | |
| 2 | import org.junit.runner.RunWith; |

|  |  |  |
| --- | --- | --- |
| 3 | | @RunWith(Cucumber.class) |
| 4 | @Cucumber.Options(format={"SimpleHtmlReport:report/smokeTest.html"},tags={"@smokeTest"}) | | |

|  |  |  |
| --- | --- | --- |
| 5 | Public class JUnitRunner { | |
| 6 | } |

Similarly you can give instruction to cucumber to run multiple tags. Below example illustrates how to use multiple tags in cucumber to run different scenarios.

|  |  |  |
| --- | --- | --- |
| 1 | import cucumber.api.junit.Cucumber; | |
| 2 | import org.junit.runner.RunWith; |

|  |  |  |
| --- | --- | --- |
| 3 | | @RunWith(Cucumber.class) |
| 4 | @Cucumber.Options(format={"SimpleHtmlReport:report/smokeTest.html"},tags={"@smokeTest",”@LoginTest”}) | | |

|  |  |  |
| --- | --- | --- |
| 5 | Public class JUnitRunner { | |
| 6 | } |

**#7) Cucumber Report:**

Cucumber generates its own html format. However better reporting can be done using Jenkins or bamboo tool. Details of reporting are covered in next topic of cucumber.

### ****Cucumber Project Setup:****

Detail explanation of cucumber project set up is available separately in next module. Please refer to Cucumber Module Part2 from more information about project setup. Remember there is no extra software installations required for cucumber.

**Implementation of Feature file:**

We have to implement these steps in Java in order to test the feature files. Need to create a class which contains those given, when and then statements. Cucumber uses its annotations and all the steps are embedded in those annotations (given, when, then).Each phrase starts with “^” so that cucumber understands the start of the step. Similarly each step ends with “$”. User can use regular expressions to pass different test data. Regular expressions take data from feature steps and passes to step definitions. The order of parameters depends how they are passed from feature file. Please refer next module for project setup and mapping between feature files and java classes.

**Example:**

Below example is to illustrate how feature files can be implemented.

In this example we have not used any selenium API. This is to just show how cucumber works as standalone framework. Please follow next module for selenium integration with cucumber.

|  |  |
| --- | --- |
| 1 | public class LoginTest { |
| 2 | @Given("^user navigates to SOFTWARETETINGHELP.COM$") | |

|  |  |
| --- | --- |
| 3 | public void navigatePage() { |
| 4 | system.out.println(“Cucumber executed Given statement”); | |

|  |  |
| --- | --- |
| 5 | } |
| 6 | @When("^user logs in using Username as \"(.\*)\" and Password \"(.\*)\"$") | |

|  |  |  |
| --- | --- | --- |
| 7 | public void login(String usename,String password) { | |
| 8 | system.out.println(“Username is:”+ usename); |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | system.out.println(“Password is:”+ password); | | |
| 10 | | } |

|  |  |
| --- | --- |
| 11 | @When("^click the Submit button$") |
| 12 | public void clickTheSubmitButton() { | |

|  |  |  |
| --- | --- | --- |
| 13 | system.out.println(“Executing When statement”) | |
| 14 | } |

|  |  |  |
| --- | --- | --- |
| 15 | @Then("^Home page should be displayed$") | |
| 16 | public void validatePage() { |

|  |  |  |
| --- | --- | --- |
| 17 | system.out.println(“Executing Then statement”) | |
| 18 | } |

|  |  |  |
| --- | --- | --- |
| 19 | @Then("^login should be successful$") | |
| 20 | public void validateLoginSuccess() { |

|  |  |  |
| --- | --- | --- |
| 21 | system.out.println(“Executing 2<sup>nd</sup> Then statement”) | |
| 22 | } |

|  |  |
| --- | --- |
| 23 | } |

When you execute cucumber runner class, cucumber will start reading feature file steps. For example, when you execute @smokeTest, cucumber will read **Feature** step and **Given** statement of **scenario**. As soon as cucumber finds Given statement, same**Given** statement will be searched in your java files. If same step is found in java file then cucumber executes the function specified for the same step otherwise cucumber will skip the step.

### ****Conclusion:****

In this module, we have covered features of cucumber tool and its usage in real time scenario.  
Cucumber is a most favorite tool for many projects as it is easy to understand, readable and contains business functionality.