Note : only High level concepts for the Cucumber he is explaining

187. Important Note - How much Cucumber does this course cover?

Cucumber is one of the exclusive VDD framework which will help us to design test automation frameworks.

TestNG is also one of the testing framework which helps us to design test automation frameworks. You can use all the features, what we have seen in TestNG from last few sections

In Cucumber also, all the features what TestNG is providing to design a framework like groups, retry reports, and runner files. Everything even Cucumber provides.

what I suggest is either choose writing Cucumber framework or go with the Java TestNG framework.

what we do is let's develop a hybrid framework So that means on top of our existing Java TestNG framework I will show how to incorporate and bring Cucumber into this framework.

So our framework will support both Cucumber tests as well as the existing Java TestNG test.

now we will see how Cucumber can play a role and leverage Java existing framework to the next level.

in this course we will use a hybrid model of both TestNG and Cucumber so that way you will be

in a position to understand how things work across.

188. Introduction to cucumber and its terminologies with examples

people generally put in meetings as a sign off, Whenever QA develop test cases, they get signed off the test cases with the business analyst. That's what people traditionally do to overcome this problem.

To overcome this this disadvantage, misleading actual behaviour Gherkins comes in to picture.

 if BA wants to write or wants to show a software business requirement, he has to define that in Gherkin syntax. So Gherkin defines some standards where if he used that standards in describing this test case, then it would be very clear for developer or QA to interpret it.

and we say as a BDD, behavior driven development. We explicitly define the behavior of software requirement using this Gherkin language.

Scenario :

each and every scenario is nothing but a Gherkin language.

Given precondition -  credit card payment may be from multiple pages, or when you navigate to your net banking, same credit card payment could be from left side navigation links, or you can access it from credit card tab, or there could be any link in the full tab page, for instant, a quick link references to pay credit card. So first we need to understand on which tab or which module we are testing this. For that, he has to provide a given, like preconditions are mentioned.

This is what we say as a behavior driven development. So in Cucumber, this way of writing a test case is called as scenario.

please remember that don't assume that if business analyst does not give you test case in this way, you cannot work with Cucumber,

If the test case is one plain single line if ypu want to work on cucumber framework just convert your test case into this model, and then you can use Cucumber automation to automate your test cases.

why do I need to convert this line into this standard to automate? So in Cucumber, whatever line you define like this, we will tag our real development code, which is our automation code,

to each and every of this line.

if you've written a code to navigate to the pay credit card page using your mobile testing

or web testing, so the code, what you wrote for navigation will be tied to this line.

this is positive testing, checking if something is displaying, use (And )

this is to check if something is not displaying, sounds like a negative test case for this particular scenario. So that, you can use a keyword (But )

if you want to depend on Gherkin syntax and Gherkin conventions, you have to use “But” keyword to showcase negativity testing, and “And” keyword to show positivity testing.

Feature & Feature File :

If client ask you to test a credit card payment flow, so it is your duty to derive as many test cases

to verify that particular business requirement.

So here, feature is credit card payment.Now, to satisfy this, I have written multiple scenarios,

nothing but multiple test cases.

in every test, whenever you define scenario, it's mandatory that you use a feature You cannot just define scenario to test your Cucumber test case. So when you define this, Cucumber will expect what is the feature for this scenario. So credit card payment is a feature for this scenario.

You have to define this.

If you don't give the feature statement, your test will fail, asking that feature is missing.

FeatureFile :

where exactly you have to write all this stuff ?

so if you want to write an automation test in REST API or Selenium, you use Java classes.

If you have done that anytime in the past, obviously, you'll use Java class.

But in Cucumber, to define all this, you have to use a feature file.

A feature file can be created by simply giving .feature extension to it ( Ex : Payment.feature )

you can dump all these tests into that file, which is nothing but a feature file.

Feature file is a test suite, and inside it, there are multiple test cases.

once we define entire feature file like this, now comes into picture, which is a step definition.

you will create a step definition.Java file, and in that file, for every line here, you will develop a real code there.

there is a file call step definition which will actually implement the code

scenario outline is almost same as a scenario. The only difference is here,the data we take from that table here, this scenario needs to run for multiple data sets.

if you remember a TestNG, there we have data provider, Where you repeat your scenario with the different data sets.

Similarly, here also, if that test needs parameterization to run with different data sets, then what we do is, instead of giving the data what we need here, we create one table, and give that table name as examples, And you run your scenario. So what happens is this complete scenario is now represented as four scenarios, so in each and every scenario,

[Cucumber Scenario Outline Example - JavaPointers](https://javapointers.com/automation/cucumber/cucumber-scenario-outline-example/)

Cucumber Terminologies:

What is Gherkin ?

It is a Business Readable, Domain Specific Langauge, that lets you describe software’s behaviour.

Example : pop up messaged is displayed when buttons are clicked and errors are gone

Developer : pop up messaged is displayed and errors are gone on button clicked

QA : pop up message is displayed – only when buttons are clicked and errors are gone

Keywords used in Cucumber : Scenario, Feature, Feature file, Scenario outline, step Definition

Scenarios:

In Cucumber Testcases are represented as scenarious.

Scenarious contain steps which are equivalent to test steps and use the following keywords(Gherklin syntax) to denote them**: Given,When,Then,But and And(case sensitive)**

* **Given**: Precondtions are mentioned in the Given keyword
* **When**: The Purpose of the when steps is to describe the user action.
* **Then**: The purpose of Then steps is to observe the expected output. The observations should be related to the business value/benefit of your Feature description.
* **Sceanrio:** Make minimum Due Payment
* **Given** user is on pay credit card page
* **When** user fills all details and select minimum amount option
* **And** user clicks on pay button
* **Then** Credit Card Confirmations Page is displayed

And if reference number is displayed

But error message is not displayed

When we specify a business req, sometimes there are multiple pre-conditions,user actions and expected outcomes

We are going to add one more scenario and will use the And and But Keywords:

**And** : This is used for statements that are an addition to the previous steps and represent positive statements.

**But** : This is used for statements that are an addition to previous steps and represent negative statements.

Feature and Feature File :

Feature represents Business Reqirement.

Feature File acts as a Test Suite which consists of all scenarios.

In CuCumber, Feature files contain Scenarios, we can simply create feature file with feature extension scenarios belonging to specific area of Application will be grouped into one Feature file.

The text that immediately follows the Feature keyword, and is in the same line, is that Title of the Fetaure file.

Feature file should contain either Scenario or Scenario outline. The naming Conventions for feature files should be lowercase with, feature extension

**Feature** : Credit Card payment

**Scenario** : Make Minimum Due payment

**Given**: user is on Pay Credit card page

**Then** user fills all details and select minimum amount option

**And** user clicks on Pay button

**Then** Credit Card confirmation page is displayed

**Scenario**: Pay Statement Balance

**Given** user is on Pay Credit Card page

**Then** user fills all details and select statement Balance option

**And** user clicks on pay button

**Then** Credit Card confirmation page is displayed

**Scenario** : Enter another Amount as 0

**Given** user is on pay Credit Card Page

**Then** user fills all details and select other Amount and enter 0

**And** user clicks on Pay button

**Then** Credit Card confirmation page is not displayed

**But** error message is displayed.

189. Setting up cucumber dependencies into Framework and create feature files

 right now our framework is exclusively on selenium testng. So let's give the flavor of Cucumber

by quickly getting the dependencies.

Google – mvn repo – cucumber java – copy and paste the dependencies in the pom.xml file

Google – mvn repo- cucumber testng – copy and paste the dependencies in the pom.xml file –( to run testng tests with the help of Cucumber )

when you go with Cucumber, then you cannot use the features like @DataProvider, which are exclusively to testing, You can use all these design patterns like page object, all these you can use but the data provider things and here groups, whatever you see exclusively for testng,

cannot be used for Cucumber because both are competitive tools which complement each other providing same output, you cannot mix them but the code-wise you can use.

src/test/java – new – package (cucumber ) – finish

here we need one plugin to support writing your feature files with nice syntax.

Help – Eclipsemarketplace – search cucumber – click on install plugin and restart your Eclipse.

once it is installed, and in this Cucumber package, right click New File (SubmitOrder.feature )– finish

Note : Feature : Title of the feature ( feature level, this is like a test suite )

One feature can have multiple scenarios just like multiple @Tests what we have.

ll’y a feature file can have multiple scenarios.

So in testng terminology, each method is considered as a test when you have @Test annotation.

In Cucumber terminology, each method we will write as a scenario, So a feature can have multiple scenarios.

first you have to land on that application. So that is a pre-requisite step, So in Cucumber you can actually separate pre-requisite test from your actual scenario.

There is one keyword called Background. So basically this executes before each and every scenario.

If you think that's a prior request and common to all scenarios, you can create a separate keyword called background.

@tag

Feature: Purchase the order from Ecommerce Website

I want to use this template for my feature file

Backgroud:

Given I landed on Ecommerce Page

@tag2

Scenario Outline: Positive Test of submittig the order

Given Logged in with username <name> and password <password>

When I add product <productName> to Cart

And Checkout <poductName> and submit the order

Then "THANKYOU FOR THE ORDER." message is displayed on ConfirmationPage

Examples:

| name | password | productName |

| rahulshetty@gmail.com | IamKing@000 | ZARA COAT 3 |

190. Implement Step definitions for features and understand regular expressions

Src/test/java – right click – new package( rahulshettyacademy.stepDefinitions ) – finish

In the above package create one class ( StepDefinitionImpl ) – finish

Under background line - this does not include any data

When I say outlined data that means when you see arrows like this and if data is coming from here, that means it is outlined data.

In basetest class copy the method for intilize the driver ( launchapplication() )

basically this method( launchapplication() ) returns landingpage object that is what we are catching here.

if you carefully observe here it is bringing name. So you cannot write any parameter here.

So it could be name or any variable, it doesn't matter. Step definition has to know that this is something where it is parameterized and the value is coming at runtime.

The value could be anything, It could be string, interior, we don't know. So you have to put regular expression there. How do you put it? So simply if you put dot plus, ( .+ ) that represents any character or any value basically, it does not care.

whenever you put regular expressions like this, you have to tell your string that this is a regular expression. So to represent that, you have to start with one small cap this symbol ( ^ )

and finally end this with a dollar ( & )

go to submit order test and copy the first line email password code and paste it

In StepDefinitionImpl class

191. Inject Selenium code in Step definition and introduction to Tidy Gherkin plugin

So basically this “And” is a conjunction for the previous step. So our previous one is a when,

so you can directly use at the rate when annotation here. So it applies to that as well.

Both works here, either when or And.

It will not work for the case of direct string providing in the step itself.

It will work if the data is driven from examples, That means from here, but if you have the string

directly on the step itself, then another simpler way. what type of data it is string -

So simply put curly braces here, tell that I'm expecting string, that's all. You don't need this dollar and cap also. Now it is still treated as this static step.

If the data is coming from example section, then that is pure regular expression because at runtime dynamically these values will fall. and represent that entire statement with cap(^) and dollar($)

now you see for these four steps in the feature file, we have four-step definition implementations

Now what you should do, you just need to run that feature file and that, at runtime, will connect to your step definition and it will execute all the code in the sequence how your steps are different.

to create all these step definition files, There is one nice plugin called Tidy Gherkin.

Just Google it. it's just a Chrome plugin  if you can download this plugin and have it to the Chrome, and if you start that,

the very simple shortcut, you know, to generate this syntaxes, just copy your entire feature file here, Place it in the column, what you see on top. And automatically, you know what,

it generates all that steps for you.

**package** rahulshettyacademy.stepDefinitions;

**import** java.io.IOException;

**import** java.util.List;

**import** org.openqa.selenium.WebElement;

**import** org.testng.Assert;

**import** io.cucumber.java.en.Given;

**import** io.cucumber.java.en.Then;

**import** io.cucumber.java.en.When;

**import** rahulshettyacademy.PageObjects.CartPage;

**import** rahulshettyacademy.PageObjects.CheckoutPage;

**import** rahulshettyacademy.PageObjects.ConfirmationPage;

**import** rahulshettyacademy.PageObjects.LandingPage;

**import** rahulshettyacademy.PageObjects.ProductCatalogue;

**import** rahulshettyacademy.TestComponents.BaseTest;

**public** **class** StepDefinitionImpl **extends** BaseTest{

**public** LandingPage landingPage;

**public** ProductCatalogue productCatalogue;

**public** ConfirmationPage confirmationPage;

@Given("I landed on Ecommerce Page")

**public** **void** I\_landed\_on\_Ecommerce\_Page() **throws** IOException

{

landingPage = launchapplication();

//code

}

@Given("^Logged in with username (.+) and password (.+)$")

**public** **void** logged\_in\_username\_and\_password(String username, String password)

{

productCatalogue = landingPage.loginapplication(username,password);

}

@When("^I add product (.+) to Cart$")

**public** **void** i\_add\_product\_to\_cart(String productName) **throws** InterruptedException

{

List<WebElement> products = productCatalogue.getProductList();

productCatalogue.addProductToCart(productName);

}

@When("^Checkout (.+) and submit the order$")

**public** **void** checkout\_submit\_order(String productName)

{

CartPage cartPage = productCatalogue.goToCartpage();

Boolean match = cartPage.VerifyProductDisplay(productName);

Assert.*assertTrue*(match);

CheckoutPage checkoutPage = cartPage.goToCheckout();

checkoutPage.selectCountry("india");

confirmationPage = checkoutPage.submitOrder();

}

@Then("{string} message is displayed on ConfirmationPage")

**public** **void** message\_displayed\_confirmationPage(String string)

{

String confirmMessage = confirmationPage.getConfirmationMessage();

Assert.*assertTrue*(confirmMessage.equalsIgnoreCase(string));

driver.close();

}

@Then("^\"([^\"]\*)\" message is displayed$")

**public** **void** something\_message\_is\_displayed(String strArg1) **throws** Throwable {

Assert.*assertEquals*(strArg1, landingPage.getErrorMessage());

driver.close();

}

}

192. Introduction to TestNG Test Runner to run Cucumber feature files

in cucumber terminology, running the test means running the feature file.

what kind of Runner you want to create, in your package do you want to create JUnit Runner or TestNG Runner, so that depends upon what framework your project is using internally, okay.

So now we are on TestNG framework.

 if your code is driven by TestNG then you need to create TestNG cucumber runner,

we have to create a TestNG runner for cucumber.

Src/test/java – In cucumber package – new class(TestNGTestRunner) – finish

You need to provide lot of options in created class, so all the options you can provide,

by defining this CucumberOptions tag,using this tag, there are a lot of helper attributes

where you need to provide

if you want to run cucumber test you have to start with CucumberOptions tag and provide which feature file to run all these inside this brackets. I have to clearly tell to your runner that where feature files are there to run.

 by default you know, the output of your cucumber will come in encoded format, that results will not be in readable, so what you can do, there is one attribute called monochrome in cucumber which will give the results in readable format, so that's a default one people will set to read the report.

how do you want to generate the report? If you want HTML report, for that you have to pull that plugin, all plugin information has to be in key value pairs. So key is, like what kind of report you want and value is, let's say I want it in target/cucumber.html. so target folder will be created for you automatically when you run your Maven project, so make sure you give that information in the codes, perfect.

So basically you are telling that, run all feature files present in this package and to map them

here is MER step definitions are there. Print the results in readable format and generate the report of HTML plugin, so these are the different helper attributes you are giving

by default cucumber will not be able to scan your TestNG assertions or your TestNG libraries.

So what you can do is, these TestNG guys, to provide seamless integration with cucumber,

they created one Class called AbstractTestNgCucumberTests, So inbuilt cucumber do not have the power of scanning your TestNG code

So if you are running using JUnit, then this is not required because by default, cucumber have the ability to run the tests which have JUnit code, okay, it comes in inbuilt, but TestNG is not something which comes inbuilt in cucumber so that's why TestNG guys have created a class

and asked cucumber to, "Hey if you can extend my wrapper class then you know what, you'll be able to read me

right click for the testngtestrunner – run

if you see on high level we are just adding one more wrapper for your already existing TestNG test, so if someone in your company ask that can we move to cucumber from any existing framework, just let them know that we just need to add one more wrapper in a plain English Gherkin language where you can do like this and you can showcase them like this, let them see this code but backend without much affecting your framework, you are still using the same core concepts of your framework and making it compatible for both Cucumber and the TestNG

193. Control the Cucumber feature files execution with Tags and Background keywords

let's quickly migrate this error validation test also in the feature file standards.

In src/test/java – cucmber package – new file(ErrorValidations.feature) – finish

I’m reuse this line, all the backend code also will be reused, right? Because they are all pointing

to the common step definition implementation. So that's one of the advantage with Cucumber,

Now, there is something called tagging, just like groups in TestNG.

Modify the scenario outline aboveline Now, here, I would call it as a @Regression

So, for error validation, let's name as @ErrorValidation.

@tag

Feature: Error validations

I want to use this template for my feature file

@ErrorValidation

Scenario Outline: Title of your scenario outline

Given I landed on Ecommerce Page

When Logged in with username <name> and password <password>

Then "Incorrect email or password." message is displayed

Examples:

| name | password |

| rahulshetty@gmail.com | Iamking@0 |

Now, let's say you want to run the tests only marked with regression.

Then, in your TestNG runner itself, you can actually provide another helper attribute

saying tags =“@Regression ”, and inside this, you can provide that tag what you want to run.

 now when you run your test, what happens is, out of all feature files it found in that folder,

it will filter down only the files which have @Regression tag, and it will only trigger them.

Cucumber cannot run on its own. So, Cucumber can provide you all the implementation framework features, like tagging, pre-request, post-request, reports. Anything, Cucumber can provide what TestNG provides. But problem here is, to run your Cucumber feature file,

you have to ultimately depend upon either a TestNG or JUnit runners.

Interview que :  how do you decide which runner you want to use?

That depends upon what kind of assertions you have in your code. Like if you use TestNG assertions completely for running your framework, then go with TestNG. If you're using JUnit assertions and libraries to completely run your framework, then go with Junit.

And now, if you want to integrate this Cucumber test with Maven and Jenkins, just like how we have done with TestNG, it's again same process, lly, let's create one more profile here,

Exclusively for Cucumber. And let's call them as a Cucumber test. So, basically, this profile will make sure that it runs this TestNG test runner. So until now, we are just right-clicking

and running the test, Instead, we again want to do the same way like how we did in past for our TestNG XMLs, that we want to trigger this using Maven command so we can integrate that with Jenkins

So you just need to make sure includes, that means give your path name, and tell what you want to include. That’s all.

go back to your terminal – mvn test -PCucumberTests - enter

 now you understood how to integrate your Cucumber runner files into the pom.xml, so that you can trigger from Maven commands.

Open the created Jenkins job and we got a new profile right go to configure in profile add one more choice

CucumberTests save build with parameters select that cucumbertests and click on build.

and your framework will run all the feature files what are configured in this test runner,

based on the tags you set.

finally, when you refresh your project, here, you will see that, in target folder, where cucumber.html will be created, because you asked us to create a file like that in the test runner,

This is how you asked, that I want a report to get generated, HTML in the target folder. Yes, it generated for you.