**Step -1 : Set up and Configurations**

1. JDK
2. Eclipse/ Intelli
3. Cucumber Plugin & Natural plugin in Eclipse Through thrpparket place/ Cucumner pluging for intellij
4. Set Environment Variable ---
5. Java Home
6. Maven
7. M2home

**Step 2: Create Maven project and update pom.xml with below dependencies.**

1. Cucumber-core
2. Cucumber-html
3. Cobertura code coverage
4. Cucumber -junit
5. Cucumber -Java
6. Cucumber -jvm-deps
7. Cucumber-reporting
8. Hamcrest-core
9. Gherkin
10. TestNG
11. Selenium-java

**Step 4: Create Folder Structure**

Project Name

|

Features(Folder)

|

pageObjects(Package)

|

StepDefinitions

|

Utilities

|

Drivers(folder)

|

Target(Reports) [Report will be generated inside the Target folder . Which is already in the project ]

|

Pom.xml [Which is already in the project ]

**\*\* Features(Folder) ---Folder – All folder should be part of Project level. Under this folder we can create multiple folder**

**Package – All package should be part of Source folder. Under the package we can create multiple java classes and we can also create multiple file.**

**Step 5: Copy Drivers into “Drivers” folder**

**### First we will see what is the application which we are going to automate**

**Step 6: Automating login Test cases --- for https://admin-demo.nopcommerce.com/login**

**6.1: Create a login.feature under Feature Files Folder**

**6.2: Create Page object Class LoginPage.java under pagesObjects package**

**6.3: Create Steps.java under stepDefinition Package and Implement required methods**

**6.4: Create TestRunner.java under myRunner package.**

**Step 7: Automating Add new customer / Search customer**

**7.1: Create a Customer.feature under Feature Files Folder**

**7.2:**

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

**### First we will see what is the application which we are going to automate. I am going to automate an E-commerce Application. It has a number of scenario. Let’s see Login is one scenario.**

**6.1: Create a login.feature under Feature Files Folder**

1. **Login is one scenario----**
2. **Click on the Login and once you login here you can see either catalog and**
3. **go to the Customers , and inside the Customer ,Click on the Customers and here we can add a new customer. So click on customer – add new and here**
4. **we need to provide all the details and here once you click on the save , it will give you a confirmation message .”Customer is added successfully ” and here**
5. The Email id should be unique for Every customer.

So these are the details which we need to pass. And this is one scenario.

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

1. **Other Scenario is Search the Customer**
2. When Click on the customers here it will provide some email ID > Click on the Search then that Email ID details will be displayed in the particular table .and
3. We can search Email ID , First name , lastname also we can do that. So this are the diff scenarios.

**First we will do for normal login. And we will go for add customers, Search Customers and other diff Scenarios.**

Feature: Login

Scenario: Successful Login with Valid Credentials

Given User Launch Chrome browser

When User opens URL "https://admin-demo.nopcommerce.com/login"

And User enters Email as "admin@yourstore.com" and password as "admin"

And Click on Login

Then Page Title should be "Dashboard / nopCommerce administration"

When User Click on Log out link

Then Page Title should be "Your store. Login"

And Close browser

**This is My Feature file, Called Login Scenario—which contain one scenario.**

--🡪 So We need start for Feature file and It will be created by Gherkin Keywords.

-🡪 First part should be given , So you need to start with the given . and then followed by Rest of the Keywords.

**-🡪 From this particular scenario– I am going to pass some parameters throw my scenario**.

**Parameter**

**#### This URL |**"https://admin-demo.nopcommerce.com/login"**| I am going to pass as a Parameter.**

**--🡪I will also pass this** 🡪 "admin@yourstore.com" as Username and password as

"admin"

**-🡪 I will also This Title as Parameter 🡪** Dashboard / nopCommerce administration"

**--🡪 Also I will also pass this Title as a Parameter --** Your store. Login"

So We need to pass some parameter to the Step Definition file .

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

So Once we have a feature file login.feature , We can Directly implement Step Definition file.and I am also going to use Page Object Model pattern .

So Further login page we need to create Page Object Class.

**6.2: Create Page object Class LoginPage.java under pagesObjects package**

**Now We need to create Page object class with content the page Element from the Login page.**

**It contains one Constructure which will initiate the Driver and for every element which is present on the login page and We need to Specify all the Element and Also Action Method.**

**\*\*\*\* I also have identified LogOut because After successful login , if you want to take another import, We must logout from a application when you will perform your data driven testing.**

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

**6.3: Create Steps.java under stepDefinition Package and Implement required** methods.

Every Step Definition file contain multiple Test steps. For every step of feature file ,We need to write a step Definition in some other class.

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

Summary – I have/ just created one feature file and this particular feature file contains a number of steps.

So when I implement this feature file , I need a step definition file. So I will write all the steps /methods in the another file. So When I execute the step definition file , It will internally call the PAGE Element from the Page OBJECT class .

So Step definition file will interact with the page object. So Page object class will be used by the Step Definition file internally.

So these three [feature file , Step definition file and Page object class ] they are communicated with each other.

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

So We can create our own step definition file or you can also write your own method based upon your keywords from the feature file. Or we can generate the methods also.

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

In step definition file contains a step definition from the multiple future file.

To do this I will run the feature file, So when I run this feature file it will give you **Unimplemented method** .

\*\*\*\*Before run the feature file , we can update the Maven project> go to project> Maven> update.

\*\*\*\*Configuration -Go to Project > Run As Configuration >

Now I need to implement step definition methods for every steps in the login feature.

Page object class – Identifying elements, Performing actions within the step definition file separated them into page Object class

First Scenario of dot feature file ---

#Author: your.email@your.domain.com

#Keywords Summary :

#Feature: List of scenarios.

#Scenario: Business rule through list of steps with arguments.

#Given: Some precondition step

#When: Some key actions

#Then: To observe outcomes or validation

#And,But: To enumerate more Given,When,Then steps

#Scenario Outline: List of steps for data-driven as an Examples and <placeholder>

#Examples: Container for s table

#Background: List of steps run before each of the scenarios

#""" (Doc Strings)

#| (Data Tables)

#@ (Tags/Labels):To group Scenarios

#<> (placeholder)

#""

## (Comments)

#Sample Feature Definition Template

@tag

Feature: Title of your feature

I want to use this template for my feature file

@tag1

Scenario: Title of your scenario

Given I want to write a step with precondition

And some other precondition

When I complete action

And some other action

And yet another action

Then I validate the outcomes

And check more outcomes

@tag2

Scenario Outline: Title of your scenario outline

Given I want to write a step with <name>

When I check for the <value> in step

Then I verify the <status> in step

Examples:

| name | value | status |

| name1 | 5 | success |

| name2 | 7 | Fail |

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

My runner Class:

package hgtest.runner;

import io.cucumber.testng.AbstractTestNGCucumberTests;

import io.cucumber.testng.CucumberOptions;

import org.testng.annotations.DataProvider;

import org.testng.annotations.Test;

@CucumberOptions(plugin = "json:target/cucumber-report.json",

features="classpath:features",

glue="hgtest.stepdefinitions"

)

public abstract class CustomCucumberAbstractTestng extends AbstractTestNGCucumberTests {

public CustomCucumberAbstractTestng() {

}

@Test(

groups = {"cucumber"},

description = "Runs Cucumber Feature",

dataProvider = "features"

)

@Override

@DataProvider(parallel = true)

public Object[][] scenarios() {

return super.scenarios();

}

}

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

package Runners;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

import org.junit.runner.RunWith;

@RunWith(Cucumber.class)

@CucumberOptions(

dryRun = false,

strict = true,

monochrome = true,

features = {"src/test/resources/features"},

glue = {"StepDefs","WebConnector"},

plugin = {

"pretty"

}

)

public class Runner {

}

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

package Runners;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

import org.junit.runner.RunWith;

@RunWith(Cucumber.class)

@CucumberOptions(

features = ".//Features/Login.feature",

glue = "stepDefinitions", // this is package name

dryRun = false,

strict = true,

monochrome = true,

plugin = { "pretty",”html:test-output”}

)

public class Runner {

}

[Automation Step by Step - Raghav Pal](https://www.youtube.com/channel/UCTt7pyY-o0eltq14glaG5dg)

**Automation Step by Step - Raghav Pal**

Selenium Cucumber Java BDD Framework 1 - Setup

Step 1 - Create a new maven project

Step 2 - Add maven dependencies

Cucumber Java | Cucumber JUnit | JUnit | Selenium Java

Step 3 - Create a folder Features under src/test/resources

Step 4 - Under features folder create a new feature file login.feature

Step 5 - Download cucumber plugin from Eclipse Marketplace

Step 6 - Create feature file and add contents

Feature

Scenario

Steps

Scenario Outline

Example

Tags

Comments

Step 7 - Try to run the feature file

Step 8 - Add Step Definitions / Glue Code under src/test/java package

Step 9 - Create a Runner class

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

we have added a feature and we have added scenario we have added steps—

**Feature: feature to test login functionality**

**Scenario: Check login is successful with valid credentials**

**Given user is on login page**

**When user enters username and password**

**And clicks on login button**

**Then user is navigated to the home page**

we can also add a scenario outline because sometimes you want to parameterize some data.

for example—

1. create a scenario and just copy this scenario again I will copy this and paste this --

**Scenario Outline : Check login is successful with valid credentials**

**Given user is on login page**

**When user enters username and password**

**And clicks on login button**

**Then user is navigated to the home page and**

1. I have to parameterize the credentials username and password and for that I will provide this inside conical brackets <> like this <username> and <password> and to get the data or values for this username and password, I will say Examples and colon Like this

Examples:

1. and in the next line I will put a pipe symbol and say username and password these are the headers and then from the next line onwards I can add values for example user1 and then pass1 then I can add more values user2 and pass2 like this –

**Examples:**

**|username|password|**

**|user1|pass1|**

**|user2|pass2|**

1. and I will do a right-click and say pretty format and

if you are using **Examples:** and if you are doing parameterization, that why we have to put this as **Scenario Outline**.then

1. save and do a right-click and click on pretty format and Then you will see all this formatting is done here.
2. so now this scenario will run two times with these two sets of values.

So whenever you want to do parameterization and want to run with multiple set of values, you can create a scenario outline and create

examples with data. and then

**Tagging--** for example I can create a tag and I want to Tag for Feature---like this ---

@SmokeScenario

Feature: feature to test login functionality

And I can also do tagging for scenario—like this

@SmokeTest

Scenario: Check login is successful with valid credential

So we can put this tags and we can run only the selected features or

Scenarios. this is just for example this is how you can do tagging in a feature file.

**Comments--** We can add comments using the hash symbol-# .

Example—

#Author

#Date

#Description

I

so we can do commenting like this.

#Author

#Date

#Description

@SmokeScenario

Feature: feature to test login functionality

@SmokeTest

Scenario: Check login is successful with valid credentials

Given user is on login page

When user enters username and password

And clicks on login button

Then user is navigated to the home page

Scenario Outline: Check login is successful with valid credentials

Given user is on login page

When user enters <username> and <password>

And clicks on login button

Then user is navigated to the home page

Examples:

| username | password |

| user1 | pass1 |

| user2 | pass2 |

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

**Step 7 - Try to run the feature file**

Error: Could not find or load main class cucumber.api.cli.Main

Caused by: java.lang.ClassNotFoundException: cucumber.api.cli.Main

**Steps –First Method--**

1. Open Eclipse IDE
2. Removing Existing JRE an add new one JDK[Module path]
3. Now click on build project
4. Run the code and then it works

**Steps –Second Method—By Changing Work Space**

1. Create a new folder like name Work Space
2. Change work space –for that go to Eclipse file> Right Click> Scroll down > see Switch WorkSpace> click on it > Select other>

Browse> Select created folder> Launch > Wait for Eclipse Relaunch

now for now I am

17:04

just going to delete the snare to

17:06

outline for this particular example I

17:09

will keep it very simple and also I am

17:11

going to delete the tags and I will save

17:17

this so we have done step number 6 and

17:20

then step number 7 is let us try to run

17:23

the feature file so I am on the feature

17:26

file I will do a right-click and I will

17:28

say run as and here you can see I am

17:30

getting an option for cucumber and this

17:34

option we are getting from the cucumber

17:35

Eclipse plugin so I will click here and

17:38

let us see the output so you can see I

17:42

am getting

17:43

so there may be some errors do not worry

17:45

about that I will show you later so here

17:48

this is successful so here you can see

17:51

the output and Here I am getting

17:53

undefined scenarios this is showing me

17:55

the count of scenarios and the count of

17:57

steps and here it is saying the scenario

18:01

and steps are undefined and then we are

18:03

getting a message here you can implement

18:08

missing steps with the snippet below and

18:10

it is showing us the snippet so this is

18:13

actually the step definition or the glue

18:15

code so you can see here we are getting

18:17

the same statements user enters username

18:20

and password when user clicks on login

18:22

button user is navigated to the home

18:25

page we are not getting given let me

18:27

just check again so let me try this now

18:38

I have written the given statement again

18:41

and I will save and run this again run

18:47

as cucumber feature and now if I go to

18:51

the console you can see I am getting

18:54

this so this is the step definition or

18:56

the glue code and basically this means

18:58

the statements that we have written here

19:01

all these statements there has to be a

19:03

back end code for these statements so

19:05

the system or the processor will know

19:09

that when I come here what is the exact

19:12

code I have to execute and this is what

19:14

we call as step definition or glue code

19:17

so that is the next step we have to add

19:20

the step definition or glue code under

19:24

SRC test package under SRC test Java

19:27

package now again you can add it

19:29

anywhere it is your wish but to keep it

19:31

consistent and with the proper

19:36

guidelines we will create under SRC test

19:38

Java package so we have this SRC test

19:41

Java package here I can do a right click

19:44

new and first I will create a new folder

19:48

so I can put all the step definitions

19:50

inside this folder and I will name this

19:53

as step to finish

19:56

you can say steps or step definitions

20:01

whatever you like

20:31

finish and inside the step definition

20:34

now I'm going to create a class so I

20:39

will do a right-click new class and I

20:45

will say this is login steps you can

20:47

name it anything and I will say finish

20:51

so this is the class and I can increase

20:55

the font using ctrl + on my keyboard or

20:58

control - to decrease the font I hope

21:01

this is visible and now I am just going

21:03

to copy this and I think from here that

21:07

I have got in the console and I will go

21:10

back to my class and inside these two

21:14

curly brackets of the class I am going

21:17

to paste this so I have added everything

21:20

here and here I have to make one

21:23

correction this should be end and let me

21:27

check from the feature file and yes this

21:29

is end so here we have created the step

21:34

definitions now you can also create this

21:36

manually it is not necessary that you

21:38

have to take it from the console so it

21:41

is basically functions with some

21:43

annotations let me show you from scratch

21:45

for example I can create a simple

21:48

function I will say public void verify

21:52

login page you can give any name and

21:55

brackets and a curly bracket start and a

21:58

curly bracket stop so this is a function

22:01

and we can annotate it with the cucumber

22:05

keywords or annotations for example if I

22:08

want that whenever the control comes

22:11

here given user is on login page then

22:14

this should get executed this function

22:16

should get executed for that I will say

22:19

head given and in brackets I'm going to

22:23

use the exact same statement with that

22:26

is user is on login page I will copy

22:29

this and paste it here now whatever I

22:34

write here let us say as of now I am

22:36

just writing a print statement I am

22:39

saying sys o and then ctrl spacebar on

22:42

the keyboard to autocomplete the print

22:44

statement

22:45

that is system dot out dot println and I

22:48

will just say hello world

22:51

so whenever the control comes here on

22:53

this statement it will go to its

22:56

back-end step definition which is this

22:58

and then just print out hello world on

23:00

the console in real world people have

23:03

the code the selenium code to navigate

23:05

to the login page and similarly you can

23:07

create functions for all these

23:10

statements which we have already seen

23:12

here so let me just delete this for now

23:15

and here we have caught our functions

23:18

and we can also correct the formatting I

23:22

will press ctrl a to select everything

23:23

and control I to correct the formatting

23:26

and now if you hover over all these

23:29

annotations like given you can see I am

23:32

getting this option if I hover over

23:34

given here I am getting this option

23:36

import given from i/o dot cucumber dot

23:39

Java dot en so we have to get this from

23:42

this package so I will click here I will

23:46

click here and you will see the import

23:49

statements coming here and now the error

23:51

is gone similarly I will do for when I

23:53

will import from i/o dot cucumber and

23:56

for and and for then so all these

24:07

imports are done and we don't have any

24:09

errors now also let me add some dummy

24:14

statements inside these functions so

24:17

when it comes here I will say print

24:20

inside step user is on login page and

24:26

similarly for this I will say inside

24:31

step user enters username and password

24:38

then here I will say inside step clicks

24:46

on login button and here I will say

24:53

inside step user is navigated to the

24:58

homepage now in some documentation you

25:01

will also find that there are some

25:03

regular expressions used something like

25:05

there will be a cap symbol here like

25:08

this and they will be a dollar symbol

25:11

here so basically we can use regular

25:13

expressions in these statements so that

25:16

we can match in whatever way we want and

25:19

just in case you are new to regular

25:20

expressions you can go to reg x1 comm

25:27

here is the link the website reg x1 and

25:31

you will find all the details and how to

25:34

learn regular expression as well and

25:36

that a gluten expression I was talking

25:38

about was this starts and ends match so

25:41

this is the cap and dollar symbol if I

25:44

go here you can see we have a way to

25:48

tighten our patterns to define the

25:49

pattern that describe both the start and

25:51

end of the line using the head and the

25:53

dollar sign so this is how you can use

25:56

it and just in case you are completely

25:58

new to regular expressions you can go on

26:00

my website or automation step-by-step

26:03

comm and here or you will find a link

26:08

for stories here go to stories and here

26:14

you will find Mickey and Minnie stories

26:16

and here you will see there is a story

26:19

on reg X this will help you to

26:22

understand drag x from scratch you can

26:24

read this and now coming back so let me

26:27

just keep this very simple for this

26:29

session now we have added the step

26:34

definition or the glue code or the

26:36

backend code for our login dot feature

26:38

file and we have also added the

26:40

statements there so now let me try to

26:42

run the feature file again I will say

26:44

run as cucumber feature and let us see

26:48

the control output and you can see now

26:51

we are getting the results all that we

26:55

have written inside step user is on

26:57

login pH inside step user enters

26:59

username and password all this is now

27:01

getting executed so that means it is

27:04

able to read our step definition or the

27:07

glue code so we have done step number

27:09

eight now step number nine is we can

27:12

runner class and runner class is from

27:15

where we can actually run our feature

27:18

files with some more options so until

27:21

now we are running a single feature so

27:23

we can go here and do a right-click and

27:25

say run s cucumber feature but then then

27:28

we have multiple scenarios and feature

27:30

files and we want to also create reports

27:32

and other filters we want to do we have

27:35

to create a runner class

27:38

so the runner class will look something

27:41

like this let me show you with an

27:43