Q.1\_ what is BDD?

Ans. Behaviour driven development

A simple text file – simple plan engllish.

Q.2 what is cucumber?

Ans. Cucumber is a Testing approach, it use to implements BBD. Using Gherkin language(Given, When, Then,And,But)

Example- Given I open chrome browser

And I enter the url in the address bar

And I hit the Enter button

Then I see the landing page appearing

Class(java) file

Esko execute krne ke liye krte hai –

TestNG,Maven ,Junit

Simple text file .

We write our test scenarios in it . **src/test/resorces**

s

s

Class (java) file

Esme annotatin dete hai jo feature file m use hua hai like\_@Given,@When,@Then,@

Matching glue code

Feature file Step definition file Runner file

**How to make a simple BDD Cucumber Project using MAVEN**

1. Install Eclipse Plugin URL(Cucumber).
2. Maven Project.
3. Pom.xml which dependencies are important for BDD Cucumber.

Cucumber-core

Cucumber-html

Cucumber code coverage

Cucumber-reporting

Picocontainer

Cucumber-java(same version)

Cucumber-junit(same version)

Cucumber-testng(same version)

1. Featurefile (src/test/resource)

Step Defination file

Runner File

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>6.10.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-junit -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-junit</artifactId>

<version>6.10.0</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-core -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-core</artifactId>

<version>6.10.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-testng -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-testng</artifactId>

<version>6.10.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>4.1.1</version>

</dependency>

**Cucumber BDD by Code Studio**

Session – 1

What is Behavior Driven Development ( BDD) Testing :-

**In BDD,** test cases are written in a natural language that even non- programmers can read.

In BDD, whatever you write must go into **Given-When-Then-And-But** steps.

* Given [ preconditions or initial context]
* When [Event or Trigger] – user ke action esme capture krenge
* Then [Expected output]

Example : Login feature test case

**Given** URL of login page is open

**When** user enter username, password and clicks on login button

**Then** Home page shall be open

**Test Driven Development (TDD) VS Behavior Driven Developent (BDD) :**

bdd capture the application behaviour in natural language and tdd not do that that why non IT people can’t understand that framework

What is Cucumber –

Cucumber is a testing approach which supports Behavior Driven Developent (BDD). It explains the behaviour of the application in a simple English text using Gherkin language.

Gherkin is a business readable language which helps you to describe business behavior without going into details of implementation.

In Cucumber, test cases are described in cucumber feature file. Feature file should have extension .feature and each feature file should have only one feature.

Important Terms used in Gherkin Language -

* Feature [Summary
* Background [ define steps that are common to all the tests in the feature file , jaise browser launch, url open, ]
* Scenario [ Title ]
* Given [ precondition ]
* When [ Event / Action ]
* Then [ Expected Output of Action]
* And [ add conditions to your steps , multiple condition ko ek sath add krne ke liye use krte hai ]
* But [ add negative type comments ]
* Scenario Outline Examples [data driven test ke liye use kiya jata hai ]

**install Cucumber –**

* maven project m help -> install new software ->

past url - [**http://cucumber.github.io/cucumber-eclipse/update-site**](http://cucumber.github.io/cucumber-eclipse/update-site)

**Project Structure :-**

package

1. Page Object
2. Step Defination
3. Utilities

Folder –

Feature file

Target(Reports, by default created)

Create Feature file(.feature)

create pageObject class

Create step definition

**Feature File Example –**

#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 |

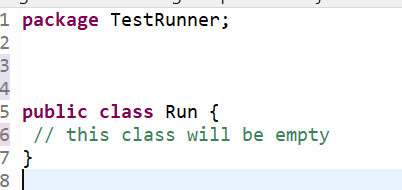
| name1 | 5 | success |

| name2 | 7 | Fail |

Dependencies for cucumber maven project –

1. webdrivermanager
2. selenium-java
3. junit
4. cucumber-core
5. cucumber-java
6. cucumber-junit
7. cucumber-testng
8. cucumber-jvm-deps
9. cucumber-reporting
10. hamcrest-core
11. gherkin
12. cobertura
13. poi
14. poi-ooxml

Note – Step Defination file ko run krne ke liye humko ek runner file create krna hai [src/test/java] m



class ke ander kuch nhi likhenge upr likhenge bs,

* ye packages create krenge

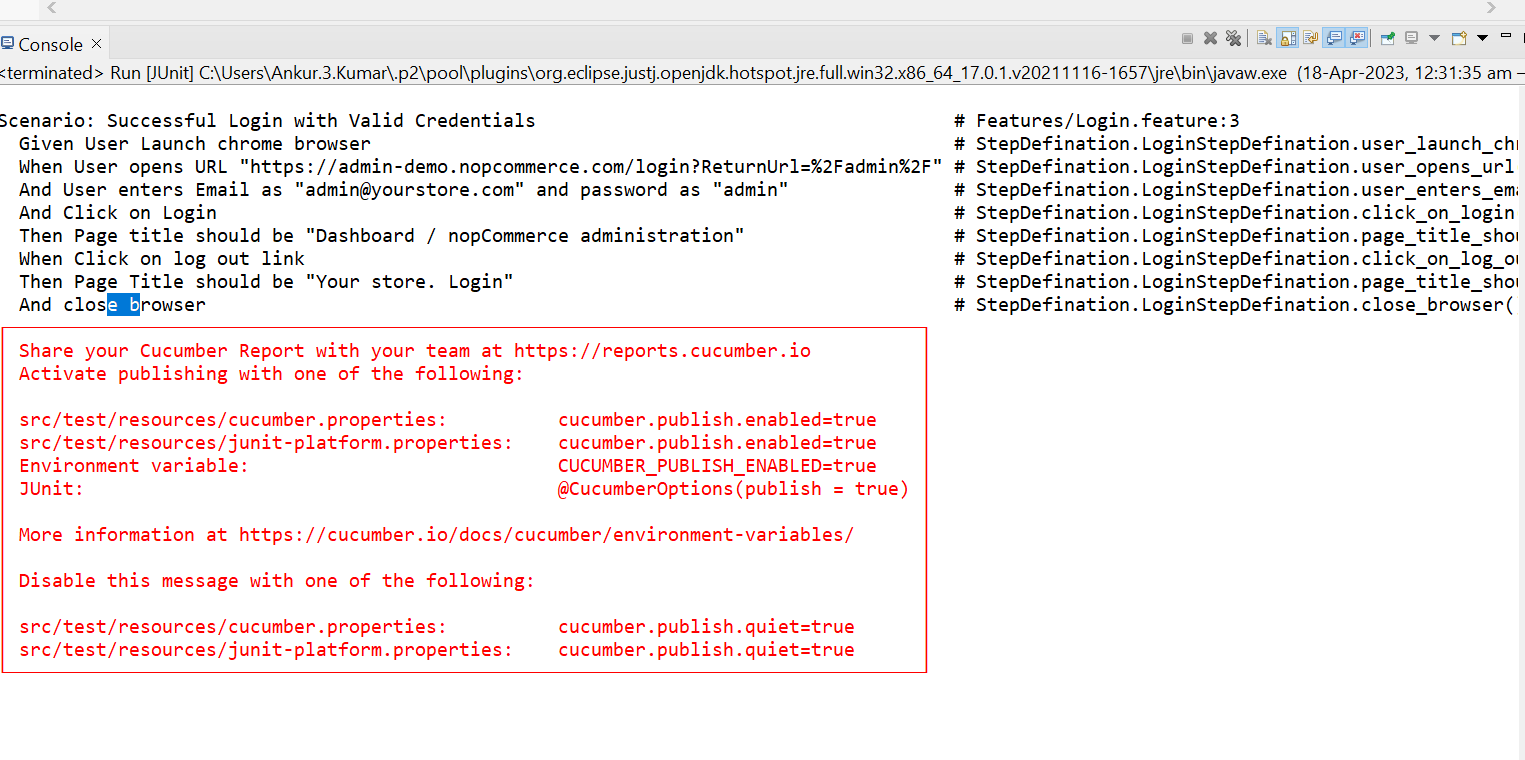
**import** org.junit.runner.RunWith;

**import** io.cucumber.junit.Cucumber;

**import** io.cucumber.junit.CucumberOptions;



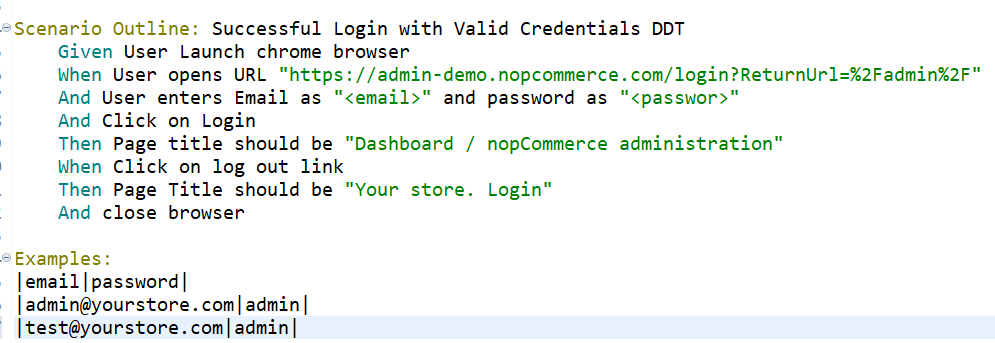
* cucumber options m features m feature file path denge
* glue m step defination file ka path(naam) dena hai
* dryRun = true [ye check krta hai feature file ke sb steps ke liye implementation method create kiya hai ya nhi(jisko step definition m create krte hai wo, sirf mapping dekhne ke liye true kre nhi to false set kre)

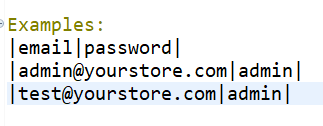
ye check krta hai ki sb feature steps ke liye implementation method hai ya nhi actual run ke time dryRun=false krte hai

* monochrome = true [ output ko readable krne ke liye hum monochrome use krte hai or esko ture kr dete hai]
* HTML report generate krne ke liye plugin m {“pretty”, se aage “html:target/cucumber-reports/reports1.html”}
* Json report generate krne ke liye plugin m {“pretty”, se aage “json:target/cucumber-reports/report\_json.json”}
* XML report generate krne ke liye plugin m {“pretty”, se aage “junit:target/cucumber-reports/report\_xml.xml”}
* Agr tino m report chahiye to tino ko comma se separator kr denge

Session – 2

Data Driven Testing ke liye feature file m Scenario Outline: keyword ka use krte hai

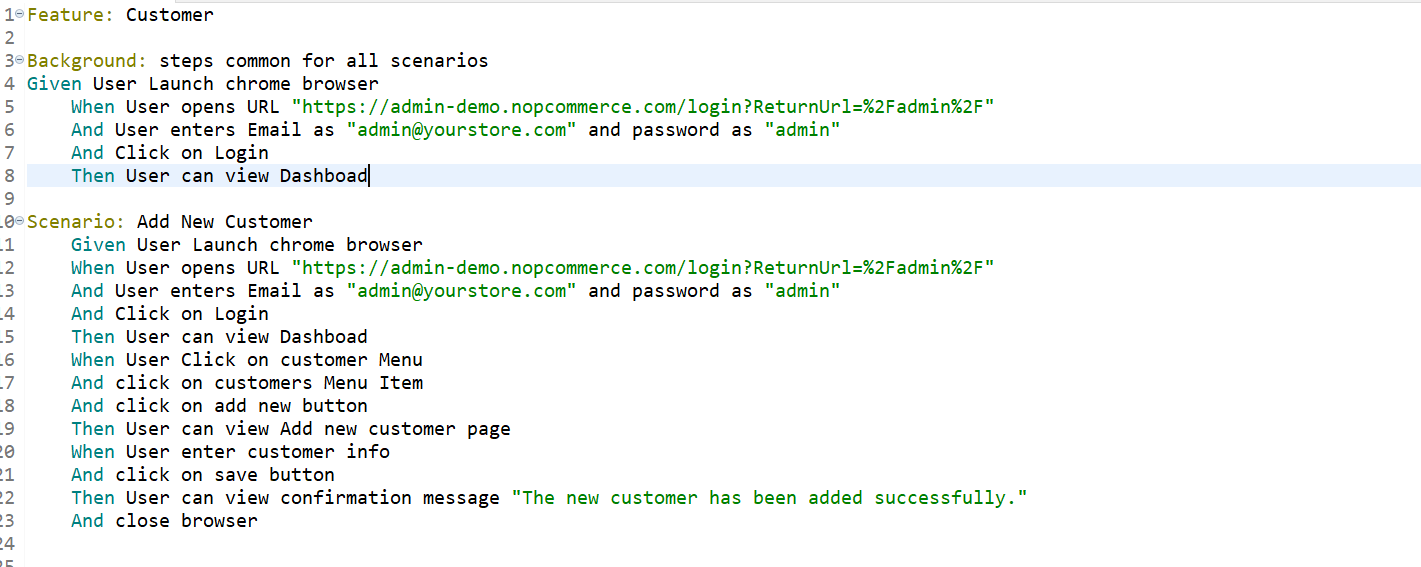




ye ek dataset hai data driven testing ke liye esme jitne data set add krenge utne he baar scenario run hoga

Session – 3(Background keyword)

Note – jo steps har scenario m use ho rhe hai usko Background: keyword m write krenge



Session – 4

-Base class

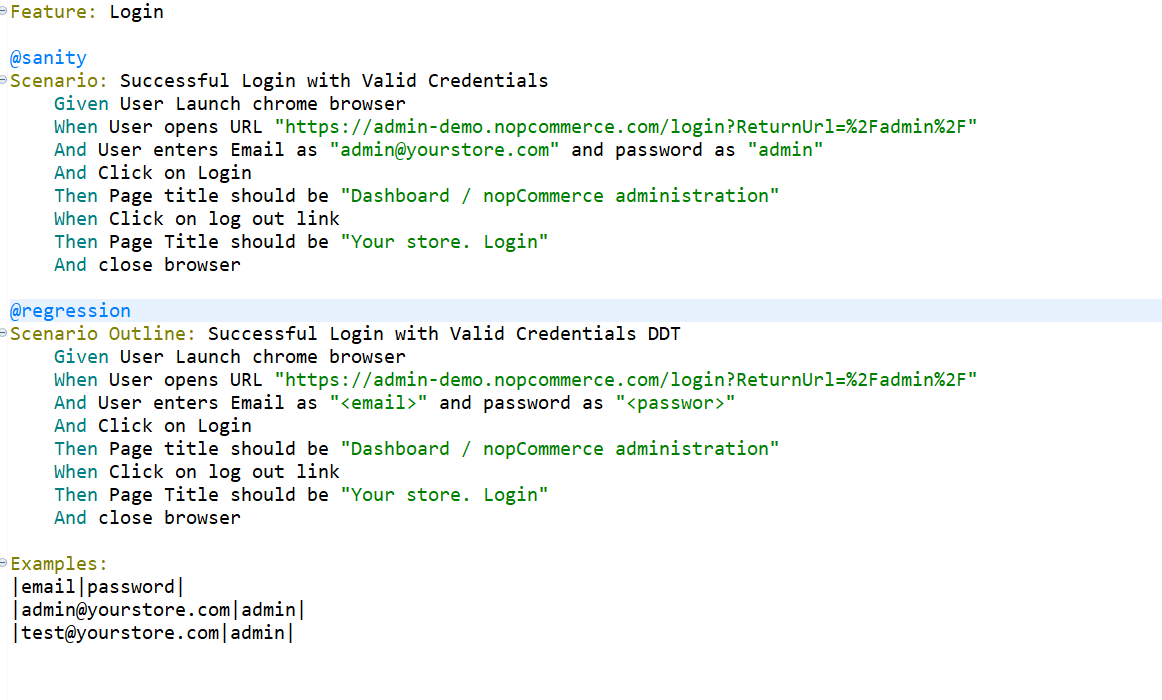
-Excuting multiple features file

-@Tags ( or , and & and not operator)]

feature file execution –

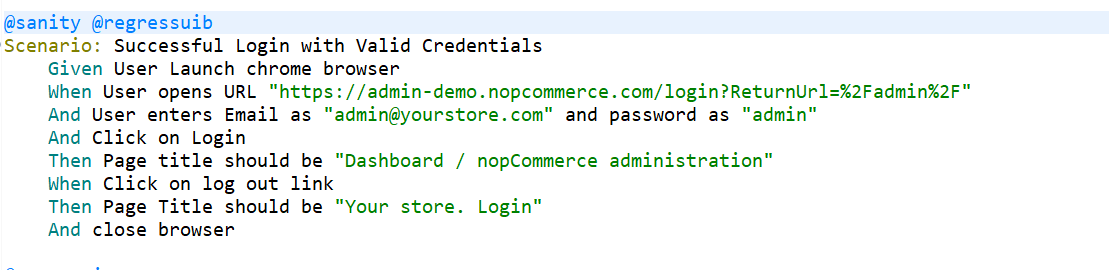
* features = ".//Features/Customers.feature", // ye sirf ek ko ko krengi
* features = ".//Features/", // agr folder specify krenge to sb feature file execute hogi
* features = {".//Features/Customers.feature",".//Features/Login.feature" }, // esme sirf jinka naam diya we execute hogi {} ka use krte hai esme

Note- scenario ko “tags” ke sath associate kr skte hai fir jo value tag m denge whi run hoga





* scenario ko 1 se jyda tag ke sath associate krna , ye regression or sanity dono m run hoga

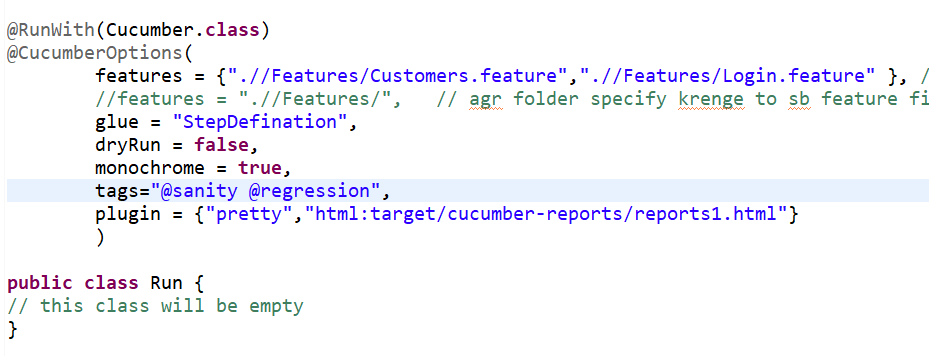


tags=”@sanity” // will run scenario which is tagged with sanity, does ot matter if its has other tags as well.

tags=”@sanity or @regression” // esme dono ke test scenario run honge sanity or regression ke

tags=”@sanity and @regression” // esme sirf we run honge jo sanity or regression dono m aate hai mtlb jinpe @sanity or @regression likha ho

tags=”@sanity and not @regression” // esme scenario exclude krte hai agr hume sanity ke execute krne sirf ‘and not’ regression ke scenario ke exclude kr dega.

tags m value dete rhenge bs

Session – 5

Hooks – Hooks are block of code that runs before and after each scenario.

1. Scenario hooks : runs before / after each scenario

@Before Annotation // steps to performs before start of testing of each scenario

Setup code :

* To start a web driver
* set up of data base connections
* set up of test data
* set up of browser cookies
* Navigation to a certain page

@Before

**public** **void** setup()

{

ChromeOptions option = **new** ChromeOptions();

option.addArguments("--remote-allow-origins=\*");

WebDriverManager.*chromedriver*().setup();

driver = **new** ChromeDriver(option);

driver.manage().window().maximize();

}

**note** - @Before tag ek se jyda use kr skte hai jisko pahle execute krna hai usko order=1(mtlb chota nbr denge) dusre ko bda.

@Before(order = 1) pahle ye hoga execute

@Before (order=2) baad m ye

@After Annotation // steps to perform after testing of each scenario

Cleanup code / teardown code:

* To stop the web driver
* to close DB connections
* to clear the test data
* to clear browser cookies
* to log out from the application
* printing reports or logs
* taking the screenshots of error

@After

**public** **void** tearDown()

{

driver.quit();

}

**note** - @After tag ek se jyda use kr skte hai jisko pahle execute krna hai usko order=max(mtlb bda nbr denge) dusre ko chota pahle bde wala execute hoga.

@After (order = 3) pahle ye execute hoga

@After(order =1)

2 – Step hooks : runs before / after each steps.

esme jo step har steps ke pahle ya har steps ke baad execute krna ho usko likhte hai

@BeforeStep

@AfterStep

@BeforeStep

**public** **void** beforeStepMehtodDemo()

{

System.***out***.println("this is before step....");

}

@AfterStep

**public** **void** afterStepMehtodDemo()

{

System.***out***.println("this is after step....");

}

3 - Conditional hooks : we can associate hooks with tags for conditional execution.

@BeforeStep(“@sanity”)

**public** **void** beforeStepMehtodDemo()

{

System.***out***.println("this is before step....");

}

@AfterStep(“@regression”)

**public** **void** afterStepMehtodDemo()

{

System.***out***.println("this is after step....");

}

jb sanity tag ke scenario run krenge to first wala execute hoga or regression scenario ke liye 2nd wala hoga

Note- esme jo hooks hote hai we testng m listner ke jaise hote hai

Failed scenario Screenshot – (brower close hone se pahle ss ka code likh denge)

@After

**public** **void** tearDown(Scenario sc) **throws** IOException

{

**if**(sc.isFailed())

{

TakesScreenshot screenshot = ((TakesScreenshot)driver);

//step2: call getScreenshotAs method to create image file

File scr = screenshot.getScreenshotAs(OutputType.**FILE**); //last m file ka naam denge

File dest = **new** File("C:\\Users\\Ankur.3.Kumar\\eclipse-workspace\\CucumberByCodeStudio\\Screenshots\\failedscreenshot.png");

// step3: copy image file to destination file , appache commans IO library add ke baad

FileUtils.copyFile(scr, dest);

}

driver.quit();

}

session – 6

Logger integration in eclipse

apache – Log4j2 (log4j2.properties file ko likh ke Logger class ka object create krenge )

Session – 7

* run test case / feature file on specific browser
* configuration.properties

Session – 8

Execute test case using pom.xml within eclipse

Execute test case outside eclipse using maven(a build tool)

1. Command line interface(CLI)
2. Batch file
3. Jenkins

pom.xml se execute krne ke liye plugin add krenge <plugins> tag ke ander-

<build>

<pluginManagement>

<plugins>

<plugin>

<groupId>org.apache.maven</groupId>

<artifactId>maven-plugin-api</artifactId>

<version>3.9.1</version>

</plugin>

</plugins>

</pluginManagement>

</build>

</project>

NOTE – bina eclipse ke test case ko execute krne ke liye system m maven download krnege or path set krenge or cmd ke through mvn test command se execute kr skte hai

NOTE – Batch file ke though execution

1. first we create a text file with extension bat(eg. run.bat)
2. es file ko notepad m open krke cd se aage project ki location dal denge

CD C:\Users\Ankur.3.Kumar\eclipse-workspace\CucumberByCodeStudio

mvn test

esko double click krke run kr skte hai

Session – 9

(cucumber ke sath testNG kaise integrate krenge)

Step 1 – Update pom.xml

1. Add:

testng dependency ---------------------

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.7.1</version>

<scope>test</scope>

</dependency>

add cucumber dependency

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-testng -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-testng</artifactId>

<version>7.12.0</version>

</dependency>

1. Remove :

Cucumber-Junit dependency from pom.xml

© same pom.xml

Step 2 – Update Maven Project

Right click on pom.xml -> select maven -> update project

Step 3 –

1. open test runner file (run.jave)
2. Remove / comment out “@RunWith(Cucumber.class)”
3. Instead of junit imports, use testng imports in run.java
4. add “import io.cucumber.testng.AbstractTestNGCucumberTests;” in run.java
5. extend this class from “AbstractTestNGCucumberTests”

Step 3:

Create TestNG.xml file

**public** **class** Run **extends** AbstractTestNGCucumberTests{

// this class will be empty

}

ek se Jayda stepfile ko kaise manage krenge –

1. dono step file m method same nhi hone chahiye

Session – 10 (Extent Report)

Extent Report – Extent report is an open-source library. with this:

can create beautiful, interactive and detailed reports for automation tests.

Extent repots are HTML – based documents that can carry detailed information about the test executed along with custm logs, screenshots and use a pie chart to represent an overview of the test.

It can integrate with almost all the major testing frameworks like Junit, TestNG, etc.

STEPS TO INTEGRATE EXTENT REPORT IN CUCUMBER BDD FRAMEWORK:

step 1 – Firstly, add following extent report dependencies to pom.xml

<dependency>

<groupId>tech.grasshopper</groupId>

<artifactId>extentreports-cucumber7-adapter</artifactId>

<version>1.13.0</version>

</dependency>

Step 2- create extent.properties file in { src/main/resource }

extent.reporter.spark.start=true

extent.reporter.spar.out=Report/Spark.html

screenshot.dir=Screenshoots/

screenshot.rel.path=../Screenshots/

#customize the report folder name by adding folder name and non reporting pattern

basefolder.name=test-output/SparkReport

basefolder.datetimepattern=d-MMM-YY HH-mm-ss

example – SparkReport 15-may- 22 16-32-58

Step 3 :

Add following @AfterStep method in step definition file to capture screenshot and attached it to extent report

@AfterStep

public void addScreenshot(Scenario scenario){

if (scenario.isFailed())

{

final byte[] screenshot = ((TakesScreenshot)driver).getScreenshotsAs(OutputType.BYTES);

//attach image file report (data, media type, name of the attachment )

Scenario.attach(screenshot, “image/png”, scenario.getName()); //last m ss ka name de skte haisbse upr jo aayega esme scenario ka naam he attach krre hai

}

note – scenario.attach method screenshot report m attach krta hai

Step – 4 :

In Test runner file “run.java” add extent report plugin

plugin = {com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:”}

Given login with username <email> and <password>

Ye stepDefination m ese likha jayega regular expression m

@Given("^login with username (.+) and (.+)$")

It is work only when data driven only example: , using regular expression

It directly give the data in fiture file “ankurkumar” like that using another concepts

Tidy gherkin plugin of google – esme feature file paste kr denge automatically generate ho jega stepdefination ka structure.