Cucumber plugin installation steps:

1. Open eclipse->go-to help->eclipse marketplace->search->cucumber->press enter –click on install
2. Once after install restart eclipse

Create maven project:

Add below cucumber dependencies

**1. cucumber core**

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

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-core</artifactId>

<version>1.2.5</version>

</dependency>

**2.cucumber java**

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

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-java</artifactId>

<version>1.2.5</version>

</dependency>

**3.cucumber testing**

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

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-testng</artifactId>

<version>1.2.5</version>

</dependency>

**Gherkin keyword:-**

Given, When, Then, And, But

**Feature file:- it contains scenario in a simple English language**

**Feature:** Login feature

**Scenario:** user login to the application as a valid user

*Given* user launches the browser and enter the url

*When* user enter username and password and click on login button

*Then* Home page should display

**StepDefinition:- it implements empty code snippets given by feature file**

**public** **class** LoginStepDef {

WebDriver driver;

@Given("^user launches the browser and enter the url$")

**public** **void** user\_launches\_the\_browser\_and\_enter\_the\_url() **throws** Throwable {

driver=**new** ChromeDriver();

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

driver.manage().timeouts().implicitlyWait(10, TimeUnit.***SECONDS***);

driver.get("http://localhost:8888/");

}

@When("^user enter username and password and click on login button$")

**public** **void** user\_enter\_username\_and\_password\_and\_click\_on\_login\_button() **throws** Throwable {

driver.findElement(By.*name*("user\_name")).sendKeys("admin");

driver.findElement(By.*name*("user\_password")).sendKeys("manager");

driver.findElement(By.*id*("submitButton")).click();

}

@Then("^Home page should display$")

**public** **void** home\_page\_should\_display() **throws** Throwable {

String actualTitle = driver.getTitle();

System.***out***.println(actualTitle);

Assert.*assertEquals*(actualTitle, "Administrator - Home - vtiger CRM 5 - Commercial Open Source CRM");

driver.close();

}

}

**Runner file:- it consists cucumber options like feature, glue, monochrome, dryrun, plugin, tags**

**It is used to run the testscript**

**package** cukeRunner;

**import** org.testng.annotations.Test;

**import** cucumber.api.CucumberOptions;

**import** cucumber.api.testng.AbstractTestNGCucumberTests;

@Test

@CucumberOptions(features = ".\\src\\test\\java\\feature\\Login.feature",

glue = "stepDefinition")

**public** **class** Runner **extends** AbstractTestNGCucumberTests{

}

**Feature(String[])->** mention the path of feature file

**Glue-(String[])**where exactly your step definition

**dryRun**-> (accepts Boolean true or false)it will map feature file and step definition and make sure that all steps have step definition

**monochrome(true or false)->**remove special character in console output

**plugin(String[])-**gererating the report

**1.How to achieve data driven testing in cucumber**

a. directly passing data within double quotes **“data”** in feature file

b. using **scenario outline** with **examples** (In scenario outline the scenario will execute multiple times with different set of data or value)

c. by using dataTables inside scenario and converting it into lists or maps(in list we fetch value by using index but in map we fetch value by using first row as key )

**2.What is the use of BackGround in cucumber?**

Whenever we have a common steps in every scenario we keep it in background and it will execute for each and every scenario in a current feature file

**3.what are Hooks in cucumber?**

in cucumber @Before and @After we call it as Hooks and this hooks will execute for each and every scenario. And no need to extends Hooks class for every step definition and we have to keep our hooks class inside step definition

**4.what are Tags?**

If we want to group our scenarios we for tags here we mention tags like @SmokeTest ,@Functional @End2End, @Regression etc.. and in runner class we are mentioning which tag to execute using tags options

**5.What are all the reports ur generated in cucumber?**

We have generated html report and also we generated extent report

**package** cukeRunner;

**import** org.testng.annotations.Test;

**import** cucumber.api.CucumberOptions;

**import** cucumber.api.testng.AbstractTestNGCucumberTests;

@Test

@CucumberOptions(features = {".\\src\\test\\java\\feature"},

glue = {"stepDefinition"},

dryRun = **false**,

monochrome = **true**,

tags = {"@Regression"},

plugin = {"pretty",

"html:target/cucucumber-reports",

"json:target/Destination/cucumber.json"})

**public** **class** Runner **extends** AbstractTestNGCucumberTests{

}

//tags = {"@SmokeTest,@Regression"} OR

//tags = {"@SmokeTest","@Regression"} AND

//tags = {"~@SmokeTest,@Regression"} exclude tag

**6.What is tagged Hooks?**

These hooks will execute only for specified tags for particular scenario and it will not execute for any other scenario

Eg:

@Before("@SmokeTest")

**public** **void** runBeforeSmokeTest() {

System.***out***.println("run Before smoke test");

}

@After("@SmokeTest")

**public** **void** runAfterSmokeTest() {

System.***out***.println("run After smoke test");

}

Feature: Login feature

Background:

Given user launches the browser and enter the url

@SmokeTest

Scenario: user login to the application as a valid user

When user enter username and password and click on login button

Then Home page should display

@Functional

Scenario: user login to the application as a Invalid user

When user enter wrong username and password

And user click on login button

Then error message should display

**Scenario having @smokeTest is execute and not for functional**

**7. How will you order or sequence your execution @before and @after?**

We passing an attribute order in @before and @After in order to sequentialize our execution

@Before will execute in Ascending order (0,1,2,3…)

@After will execute in descending order(3,2,1,0…)

**public** **class** Hooks {

@Before(order=0)

**public** **void** runBeforeSmokeTest() {

System.***out***.println("======================");

System.***out***.println("start execution first");

System.***out***.println("======================");

}

@Before(order=1)

**public** **void** setUp() {

System.***out***.println("======================");

System.***out***.println("start execution second");

System.***out***.println("======================");

}

@After(order = 1)

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

System.***out***.println("======================");

System.***out***.println("end execution first ");

System.***out***.println("======================");

}

@After(order = 0)

**public** **void** runAfterSmokeTest() {

System.***out***.println("======================");

System.***out***.println("end execution second");

System.***out***.println("======================");

}

}

**8. How to take a screenshot for failed script?**

**public** **class** Hooks {

**public** **static** WebDriver *driver*;

@Before

**public** **void** setUp(Scenario scenario) {

System.***out***.println("initialize property file");

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

System.***out***.println(scenario.getName());

}

@After

**public** **void** tearDown(Scenario scenario) **throws** IOException {

**if**(scenario.isFailed()) {

EventFiringWebDriver event=**new** EventFiringWebDriver(*driver*);

File src = event.getScreenshotAs(OutputType.***FILE***);

File dest=**new** File("./ScreenShot/"+scenario.getName()+".PNG");

Files.*copy*(src, dest);

}

*driver*.close();

}

}