* Cucumber - JVM (BDD framework) with Se*

- Behavior Driven Development.
- Language - Gherkin
- Cucumber with JVM (Java) - to define BDD framework.
  or cucumber with Ruby.
- JBehave - java (not user friendly)
- Three different components in cucumber
  (1) Feature file (2) Step definition file
  (3) ~~Test~~ Step runner file.

(1) In feature file, login.feature example
  Keywords - Given, When, Then, As, But, *
  Feature
  Scenario
  Scenario Outline
  ↑ Test cases.

(2) Step definition ~~file~~
  (Use selenium code + Java + different annotations)

(3) TestRunner - JUnit
  - To run your feature
  - To generate the output/report.

---

src/main/java
- MyTestRunner
  - TestRunner.java
- StepDefinition
  - LoginSteps.java
- Features
  - Login.feature

vol : freecrm

(1) Create maven project.
Group Id, Artifact Id same.
- Add dependencies
< dependencies > - - - </dependencies>
imp [cucumber-java, cucumber-jvm,
cucumber-junit] ← dependencies with cucumber
- Selenium dependencies (3.5.3)

(1) First create feature file.
(Eg: login.feature)
Go to Help — Eclipse Marketplace —
Search natural — Press enter.
↑
(BDD plugin)

Feature: Free CRM login feature.
Scenario: Free CRM login test scenario
Given User is in already on login page
When ~~User~~ title of login page is FREE CRM
Then User enters username and password
Then ~~User~~ User clicks on login button
And User is on home page.

(2) Package: Stepdefinition
class: Login StepDefinition & save it.
@ Given ("^User is in already on login page$")

public void user_already_on_login_page() {

. . .

}

(3) Package : Runner
class : TestRunner

@ RunWith (cucumber . class)
@ CucumberOptions (features = "Featurefile" package
glue = { "stepDefinition" },
format = { "pretty" "html:test-output" })
public class TestRunner {

}

for reports. folder name

import org.junit.runner.RunWith;
import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;

(4) Run the TestRunner class as junit.

(Execution happens in the sequence written in the feature file)

(pretty means good and understandable report)

HW Scenario : user is able to create new contact

- One feature file can have multiple scenarios.
- Advantage with cucumber BDD entire agile team can contribute. (feature file)

Part 2 : Different cucumber options

(1) dryRun (2) Features (3) Glue (4) Tags
(5) Monochrome (6) Format (7) strict.

will be defined in Runner class.

(1) dryRun — If there are say ten steps in feature file and while writing the methods in stepdefinition, you miss out any step, you can do a dryRun to check, if all steps are present.

@CucumberOptions (dryRun = true)

else dryRun = false to execute the test cases. (dryRun is to check the mapping is proper between feature file and step definition File)

(2) Features — location of feature file.

(3) glue — in which package stepdefinition is located.

(4) tags —

(5) monochrome — display the o/p in readable format.

monochrome = true will generate readable o/p in the console.

(6) format = {"pretty", "html:test-output"}

format = {"pretty", "html:test-output", "json:json_output/cucumber.json"}

// to generate o/p in json file.

---

format = {"junit:junit_xml/cucumber.xml"}

(to generate junit xml format file)

- format is used to generate different types of reporting.

---

(7) strict = true

If dryRun = false and strict = true, then all the steps definition is written properly.

If any missing steps are there in step definition, it will show pending execution.

(It will check if any step is not defined in step definition)

To comment in feature file use #

Part 3 : Data driven testing framework in selenium.

(1) Simple data driven — without example keyword.

- write username & password in double quotes in the feature file.
- Eg: @ Then (^ user enters "(.*)" and "(.*)" (regular expression)

```
public void user_enters_username_and_pass
        (String username, String password)
    driver.findElement(By.name("username")).
                sendKeys(username);
    driver.findElement(By.name("password")).
                sendKeys(password);
}
```

---

(2) with Examples + scenario outline.

Then user enters "<username>" and "<password>"

Note : Then user enters contacts ↑ if same, error

Two lines in feature file should not be same.

Examples :

username	password
naveenk	test@123
tom	test456

Note : whenever you are using data driven without example keyword, use scenario.

and

whenever you are using data driven with example keyword, use scenario outline.

HW Login — create 5 contacts.

(3) Using tables : → Part 4

Data tables in cucumber framework

- Login crmpro → click on deal
  → create deal → save.

— Use scenario.

* Then user enters username and password
  | naveenk | test@123 | ← test data table

— write data after Then.

— Create DealStepDefinition.

```
@Then("^user enters username and password$")
public void user_enters_username_and_password
(DataTable credentials) {
List<List<String>> data = credentials.raw();
driver.findElement(By.name("username")).
sendKeys(data.get(0).get(0));
— password — (data.get(0).get(1));
}
```

Disadvantage :- If we have multiple set of data, we should use example keyword only and not data table.

Part 5 : Cucumber Datadriven with maps in Selenium. (for parameterization)

Then user enters username and password

username	password
naveenk	test123

(for formating control + shift + F)

Class : DealStepWithMapDefinition.

```
@Then("^user enters username and password$")
public void user_enters_username_and_password(
        DataTable credentials) {
for(Map<String, String> data: credentials.asMaps(String.class,
                                                  String.class)) {
driver.findElement(By.name("username")).sendKeys
        (data.get("username"));
}
```

Advantage : We can have multiple set of data.

Part 6 : Cucumber Tags.

If you want to execute a particular set of test cases say for example smoke test cases or regression test cases, then we can use cucumber tags.

---

create tagging.feature feature file.

@SmokeTest @RegressionTest
Scenario: —
Given

@RegressionTest
Scenario:
Given

@CucumberOptions(
tags = {"SmokeTest"}
)

public class TestRunner { or tags = {"@SmokeTest, @RegressionTest"}

}

// OR : tags = {"@SmokeTest, @RegressionTest"}
- execute all tests tagged as @SmokeTest OR @RegressionTest.

// AND : tags = {"@SmokeTest", "@RegressionTest"}
- execute only those test cases which are both smoke & regression test cases.

// If you want to ignore any test case use tilde character
tags = {"~@SmokeTest", "@RegressionTest"}

Part 7 : Hooks in cucumber.

In stepDefinition class use annotation

Hooks { @Before - for launching the browser
@After - close the browser.

But import @Before & @After from cucumber & api jar.

@ cucumberoptions (

public class TestRunner () {

}...

- Hooks are used to define pre condition and post conditions. Before each and every Sanario, @Before and @After will be executed.

- Tagged Hooks :- If we have three Sanario say for example with tags @First @Second @Third, and we want to execute hooks @Before and @After only for @Second Sanario, then

- @Before () @After are called
global hooks.

- @Before ("@Second")
@After ("@Second") } Tagged Hooks
(only for second Sanario)

- For example in our application, if you want to set a prerequisite like delete a deal and then create new deal, you can use tagged books ie only for a particular scenario. This is missing in Testing.

- @Before(order = 0)
  @After(order = 0) } first preference.
  @Before(order = 1)
  @After(order = 1) } second preference.

Part 8 : POM with cucumber BDD framework

- src/main/java
  - com.qa.pages — LoginPage, HomePage.
  - com.qa.config — config.properties
  - com.qa.util — TestBase.java, TestUtil.java
  - com.qa.features — freecrm.feature
  - com.qa.stepDefinitions — HomePageSteps
  - com.qa.runner — TestRunner

---

multiple sets data in xml, XCel