Sample feature file

Automation Step By Step
By Raghav Pal (youtube).

```
@SmokeTest <---- (tags)
Feature : Verify Login Functionality
@Reg
Scenario ; User login attempt with Valid Credentials
Given User is on chrome browser
When User is on application URL
AND User enters user and password
Then User clicks on Login
Then User is on homePage.
```

feature file can have a scenario with Examples.

```
@Reg_Sanity
Scenario outline : User tries to login with multiple users
Given User is on chrome browser
When User is on application URL
And User enters <user> and <password>
Then User clicks on Login
Then User is on hompage

Examples :
|user | password |
|Ritesh | 12345 |
|Raghav | 12345|
```

(This test will Run two times).

TestNG → Cucumber.

@Test → Scenario

DataProvider → dataTable using Examples:

Group → @Tag in Cucumber

~~Suite~~

dependencies required
① Cucumber Java
② Junit dependency
Cucumber Junit io.cucumber

all feature files are kept in src/test/resources folder.

→ Cucumber Eclipse plugin can be installed for better readability of feature files.

① Feature Files

→ Feature : Test Login Feature.

Scenario : Verify user is able to Login to Application
Given User is on Login Page
When User enters UserName and Password
And User Clicks on Login Button
Then User is Navigated to Homepage.

② StepDefinations :- These are glue code which defined what each step in a feature file should perform.

Remember you can directly run your feature file using Run as. Cucumber Feature

③ Runner Class.
This is the place where we define the cucumber Configurations like where our feature files and where our stepdefinations are.

It has two annotation

```
@RunWith (Cucumber.class)
@CucumberOptions ( features = "src/test/resources/features" ,
                   glue = {"stepdefinations"} )
```

more cucumberoptions
monochrome = true , pretty = {}
plugin = {"pretty" , "html: target/HtmlReports"}

①

Various different plugin options availabe are

plugin = {"pretty", "html:target/reports/report.html"}
plugin = {"pretty", "json:target/JSONReports/jsonreport.json"}
plugin = {"pretty", "junit:target/JunitReports/report.xml"}
** at a time only one plugin can be used or give it by,

Next Cucumber Option is tags

tags = "@smoketest"

-x-

Parameterization in Cucumber

Adding variables to your feature file scenarios and looping through them using which we run same Scenario multiple times. (TDD testing).
Data Driven testing

To achieve this Scenario Outline is used Instead of Scenario.

Eg Feature: Test login feature

Scenario Outline: Use is able to login with valid creds

Given * browser is open
And * User is on Login Page
when * user enters <username> and <password>
And user clicks login
Then user is navigated to homepage

Examples:

username	password
Raghav	12345
Ritesh	12345

In this cage entire scenario executes twice for two rows → same like dataProvider in TestNG But

Now in relevant stepDefination we need to capture these parameters

e.g. @When ("user enters (.*) and (.*) ")
public void enter_username_pass(String username, String password) {

driver.findElement(By.id(name").sendkeys(username);
driver.findElement(By.id("password").sendkeys(password);

}

Point to be noted here if we donot use Scenario Outline which means we want to run the scenario with only one data instance, we need to use " double quotes for data we want to pass from feature file

e.g When user enters "123_Ritesh" and "@pass"

—x—

Page Object Model
in Cucumber

Design Pattern to Create Object Repository.

Pagefactory method is a pattern to implement PageObject Model.

@FindBy (id = "id_name")
WebElement name_tx ;

@FindBy (name = "name_1")
WebElement name_tb ;

to find more than one element

@FindBy (partialLinkText = "ritesh")
List<WebElement> links ;

in PageFactory patter we need to ~~do~~ call initElements()
in class constructor.

```
public class LoginPage {


    LoginPage() {
    PageFactory.initElements(driver, this);
                                ↑       ↑ class instance
                   (WebDriver instance)
    }

}
```

Also we have one more feature @CacheLookup

e.g.
```
@FindBy (id = "my-id")
@CacheLookup
WebElement id_Element;
```

In this case we are informing PageFactory to Cache this Element. So that in future if same Element is needed it won't go to DOM to find Element

→ Apply this only if we know element is static i.e it's address won't change on DOM due to any click actions.

→ Avoid using @CacheLookup of Ajax Websites.

Tags in Cucumber

@ annotations applied to scenarios to group them together for execution purpose.

e.g same way we use groups in TestNG.

@SmokeTest @regression
Scenario: User Login with valid credentials

A feature or scenario can have tag (one or more as well). We need to then supply which tag to run in CucumberRunner class

e.g. tags = {@smoke or @regression} = either condition

tags = {@smoke and @regression} = AND condition.

Plese Note, you cannot Run More than 1 tags without mentioning and OR condition (Now depriciated)

tags = {@smoke, @regression} → XNot allowed

Using tags we can also Ignore or skip test scenarios We need to arrange/select tags in RunnaClass accordingly.

We can put tags above any of below gherkin Keywords

Feature
Scenario
Scenario Outline
Examples.

@tags cannot be applied above Background or steps Keywords like Given, When, then, AND., But

tags and inherited by its child for eg is @Smoke tag is applied above feature, it is automatically inherited by scenario, scenarios outline and Examples.

tag applied on Scenario outline is automatically inheritted by Examples.

Multiple testRunner classes can be created so that we donnot need to change tags again and again.

→ Same concept like we do in testNg.xml file

Cucumber Hooks

(Nothing but @Before and @After etc.)

To perform common steps which are required each scenario run before and After.

Scenario Hook → Runs before and after each scenario
Step Hook → Runs before and after each Step
Conditiond hook → associate hooks with tags and do conditional execution

- @Before = Runs before each scenario (@BeforeTest) in testNG
- @After = Runs after each scenario (@AfterTest) in testNG
- @BeforeStep = Runs Before each step
- @AfterStep = Runs After each step.
- @BeforeAll = Runs before all scenarios @BeforeSuite in TestNG
- @AfterAll = Runs after all scenarios. @AfterSuite in testNG

These are all Important Cucumber Hooks. (io.cucumber.java). Not JUNIT

In Cucumber I believe they call it hooks since they want to differentiate between @tags and @hooks.

If you have multiple same hooks use (order=0) and priority. 0 → highest

@Before (Order=0)
@Before (Order=1)

In case of @After (order=0)
@After (order=1)

} 1 executes first then 0 (order works in reverse in after)

Scanned with CamScanner

Conditional hook

you can have conditional hook by mentioning @tags in hook annotation.

@Before ("@smoketest", order=0).

sample where both order and hook is used

Background in Cucumber feature

Background should be used if we want to show in feature what common steps have to be executed before each scenario.

Background in feature file are common steps which should be executed before each scenario in that file

e.g. Feature: Veey Longin Functionity

Background: user is logged in
Given, user is on login page
When user enters usecname and passcoorel
And clicks on login button
Then user is navigated to homepage

Scenario: check logout link
When user clicks on Welcome link
Then logout link is displayed

Scenario: verify quick lauch toolbar is prcont
when user clicks on dashboard link
Then quick launch toolbar is displayed.

Note: you can have only one Background in a feature file.

How to do command Line Execution in Cucumber

→ Required for CI/CD integration (Jenkins)

→ No dependency on IDE.

→ For batch / schedules.

mvn test runs all the files which has Test in the name all java files.

That is the beauty of maven.

mvn test -Dcucumber.options = "--h" gives help.

Basically the options you given in runner class can be supplied from here directly.

Cucumberselenium.git → This is your github Repository of all programs done in pratice