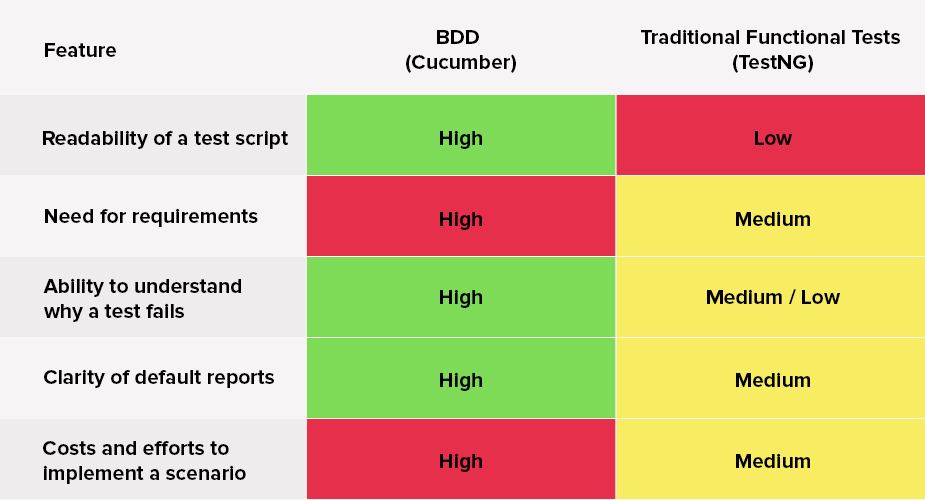
Cucumber BDD framework

BDD – Behavioural Driven Development Framework

We will define the behaviour of the use cases/testcases/framework



Behaviour:

Normal plain English

Using Gherkin keywords (Given, when, then, and, as, \*)

Alternate – Jbehave (complex, not user friendly)

Cucumber can be integrated with Ruby/JAVA

Not only for selenium, but also WebService automations, API automation

**Test Runner**

Junit/testNG

1. Run feature
2. Run StepDefinition
3. Generate output

Feature

+

JAVA

+

Annotations

**Feature:** feature name to be tested

**Scenario:** scenario sentence

**Given**

**When**

**Then**

**As**

**\***

**Scenario Outline** with **Example**

**Step Definition file**

**Feature file** [**Login.feature**]

Steps:

1. Open Maven project
2. Add cucumber dependencies
3. Go to eclipse market place 🡪 natural (eclipse plugin cucumber feature)
4. Create feature file **abcd.feature**

**Example:**

**Feature:** Award Normalization Feature

**Scenario:** Cheking Login Functionality

Given User Logging in with username and password

Then User Validates Home Page Title

And User Validates Home Page

1. Create setpdefinition file

**Example:**

@Given(“^User Logging in with username and password$”)

public void User\_Logging\_in\_with\_username\_and\_password()

{

//codes

}

1. Create runner file with below details

@RunWith(Cucumber.class)

@CucumberOptions(

features = {"C:\\Users\\Administrator\\eclipse-workspace\\AwardNormalization1\\src\\main\\java\\AwardNorm.feature"}

, glue={"com.cmacgm.stepDefinitions"}

, plugin = {"com.cucumber.listener.ExtentCucumberFormatter:output/report.html", "pretty", "html:target/cucumber-reports"}

, tags = { "@SmokeTest1"}

, monochrome=true

, dryRun=false

)

public class TestRunner extends TestBase{

public TestRunner() throws IOException {

super();

}

# Cucumber options (in Runner file)

**Example:**

**@RunWith(Cucumber.class)  
@CucumberOptions(plugin = {“pretty”}, strict = false)**

1. **dryRun** – in **@CucumberOptions**
   1. dryRun = true – Check all the steps in feature file have developed in StepDefinition file (just mapping bw both files)
2. **features**
   1. Location of feature file
      1. features = “Feature“ or
      2. features = “src/test/features“
3. **Glue**
   1. Location of your stepdefinitions
      1. glue = “stepDefinition“ or
      2. glue = “src/test/stepDeinition“
4. Tags
   1. Cucumber run scenarios with a particular tag.
      1. tags ={“@Smoke“} – to include
      2. tags ={“~@Sanity“} – to exclude
      3. more than one tag by separating comma
         1. Logical AND - tags ={“@Smoke”,”@Sanity”}
         2. Logical OR - tags ={“@Smoke, @Sanity”}
5. monochrome
   1. monochrome = true – Console output much more readable
   2. monochrome = false – Console output is not as readable as it should be
6. format/Plugin
   1. different types of reporting (html, json, xml reports)
   2. plugin = {“pretty”  , “html:Folder\_Name”, “json:Folder\_Name/cucumber.json”, “junit:Folder\_Name/cucumber.xml” }
7. strict
   1. strict = true - Cucumber will mark that particular un-defined step def as FAILURE, also show entire scenario failed.
   2. strict = false - Cucumber will not mark that particular un-defined step def as FAILURE
8. Snippets
   1. Cucumber will generate snippets for undefined steps, usually with underscore.
   2. We can customize that.
      1. SnippetType.CAMELCASE
      2. SnippetType.UNDERSCORE

# Data Driven Testing Using Cucumber

1. Simple data driven – without example [use regular expressions]

Example:

**Feature:** Given User Logging in with "**abc@ymail.com**" and "**abcdef**"

**Stepdef:** @Given(“^User Logging in with **\"(.\*)\"** and **\"(.\*)\"**$”)

public void User\_Logging\_in\_with\_username\_and\_password(**String un, String pw**) {...}

1. With example + Scenario outline **[will run multiple times with multiple set of data]**

Example:

**Feature:** Given User Logging in with "**<userName>**" and "**<passWord>**"

Examples:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| | | username | | | | passWord | | |
| | | *Kalai* | | | | *abc@123* | | |
| | | *Selvam* | | | | *abc@123* | | |

**Stepdef:** @Given(“^User Logging in with **\"(.\*)\"** and **\"(.\*)\"**$”)

public void User\_Logging\_in\_with\_username\_and\_password(**String un, String pw**) {...}

1. DataTable using List **[won’t run multiple times, so given only single set of data]**

Example:

**Feature:** Given User Logging in with credentials

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| | | *Kalai* | | | | *abc@123* | | |

**Stepdef:** @Given(“User Logging in with credentials”)

public void User\_Logging\_in\_with\_username\_and\_password(**DataTable dt**)

{

**List<List<String>>** l = dt.row();

userNameLink.sendKeys(l.get(0).get(**0**));

PasswordLink.sendKeys(l.get(0).get(**1**));

}

1. DataTable using Map

**Feature:** Given User Logging in with credentials

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| | | ***useName*** | | | | ***password*** | | |
| | | *abc@com* | | | | *Abcd* | | |

Then User enters deal details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| | | ***Title*** | | | | ***amount*** | | |
| | | *Aaa* | | | | *1000* | | |
| | | *Bbb* | | | | 2000 | | |

**Stepdef:** @Then(“User Logging in with credentials”)

public void User\_Logging\_in\_with\_credentials (**DataTable dt**)

{

**For(Map<String, String>** l : dt.asMaps(String.class, String.class))

{

userNameLink.sendKeys(l.get(“**userName**”));

PasswordLink.sendKeys(l.get(“**password**”));

}

}

@Then(“User enters deal details”)

public void User\_enters\_deal\_details (**DataTable dt**)

{

**For(Map<String, String>** l : dt.asMaps(String.class, String.class))

{

titleLink.sendKeys(l.get(“**title**”));

amountLink.sendKeys(l.get(“**amount**”));

}

}

# Hooks

1. @Before()/@After()

Will run before and after of each and every Scenario

@Before/@After

public void setup/teatDown()

{...}

Note: we can give multiple hooks with order attribute

@Before(order=1)

@Before(order=0)

@After(order=0)

@After(order=1)

1. Tagged Hooks (Hooks associated with Tags) [not available in TestNG]

Run only before tagged Scenarios

@Before(“smokeTest”)

public void setup ()

{...}

**Order:**

**GlobalBefore**

**TaggedBefore**

**TaggedScenario**

**GlobalAfter**

**TaggedAfter**

# Sample feature file

**Feature:** Test Background Feature

**Description:** The purpose of this feature is to test the Background keyword

**Background:** User is Logged In

**Given** I navigate to the login page

**When** I submit username and password

**Then** I should be logged in

**Scenario:** Search a product and add the first product to the User basket

**Given** User search for Lenovo Laptop

**When** Add the first laptop that appears in the search result to the basket

**Then** User basket should display with added item

**Scenario:** Navigate to a product and add the same to the User basket

**Given** User navigate for Lenovo Laptop

**When** Add the laptop to the basket

**Then** User basket should display with added item

Note: two ways to make a code run before scenario

1. Background in feature file [for feature level dependency]
2. Hook/tagged hook in hooks file [for scenario level dependency]

Both will run before scenario, among them hooks will run before background