What is BDD Framework?

- BDD stands for Behavior Driven Development Framework.
- BDD (Behavior Driven development) framework is extension of TDD (Test Driven Development) framework.
- BDD always focus on Scenario not test cases.
- BDD focus on what to test not a how to test?
- Is uses plain English language to explain what is exact requirements
- We use gherkin keyword to write the feature file.

- **There are different tools available in BDD Framework**
  1. Cucumber
  2. JBehave
  3. SpecFlow

What are the different languages supported by Cucumber?

- Java
- Java Script
- PHP
- Python
- Perl
- Ruby
- Dot net

BDD Framework using Cucumber Tool

- Cucumber using Java
- Cucumber is an open-source tool and does not require licensing.
- Cucumber tools use plain English to write the feature files, and these features easily read by any non-technical members in the team.
- Cucumber tool is used to test the web-based application only.
- We can configure cucumber with different languages such as java, python, c#, java script, groovy script, PHP, ruby and Perl.
- Cucumber tool easily configure with different IDE as like Eclipse , IntelliJ idea etc.
- We can easily configure cucumber with different build tool also as like Maven, ant, Gradle etc.
- We Can configure cucumber java with different testing framework as like Junit or TestNG.
- In Cucumber tool we easily can understand the what is requirements, and it act as like bridge between developer, tester, business analyst and Stake holders.

There are 3 main files in cucumber

There 3 main files are present in cucumber

1) Feature file

2) Step Definition

3) Test Runner class

Feature Files

- We create Feature file with dot feature extension.
- Feature file is entry point for cucumber tests.
- We can create many features as per the project requirements.
- Inside the feature file we create multiple scenarios.
- We write feature file by using Gherkin keyword and using plain text language.
- There are different Gherkin keyword present inside the Feature files.
    1. Feature:
    2. Scenario:
    3. Scenario Outline:
    4. Given
    5. When
    6. Then
    7. And
    8. But
    9. *
    10. Background:
    11. Examples:

- Above each keyword used to some purpose.

- Feature keyword is to provide a high-level description of a software feature functionality, and to group related scenarios.
- In feature file first keyword must be the Feature followed by colon symbol then short description of main functionality.

- Scenario keyword is used to write what is exact requirements we have to test. instead of using Scenario keyword we can use Example keyword also.

- To define different steps inside the Scenario we use different Gherkin keywords as like’

- Given keyword is used to mentions preconditions.
- When keyword is used to describe an event, or an *action*
- Then keyword is used to describe an *expected* outcome, or result
- And keyword is used to combine the 2 different steps.
- But keyword is used declare multiple conditions.
- * Keyword is used to mention any above conditions, and it is also called as global keyword.

- Scenario Outline keyword is used to run the same Scenario multiple times, with different combinations of test Data or we can say it is used to achieve the data driven testing in cucumber. Instead of using Scenario Outline keyword we can use Scenario Template keyword also.

- Examples keyword is used to write test data for Scenario Outline. And We can use one more Examples keyword inside the Same feature file. Scenario Outline it runs as per the Set of rows mentioned inside the Examples keywords.

- Background keyword is used to mention the pre-condition for each and every scenario in feature file and We use background keyword after the Feature keyword.

- In feature file we group the scenario by using @ keyword followed by tagName.
- Can we provide the multiple tagName for one scenario as per the requirement wise.

- In feature if we mention any String value within the double quote then automatically cucumber will consider it is String Parameters in snippets.

- In feature file, if we write same steps multiple times, then cucumber is generating the snippets only one time.

- In cucumber if we want to mention test data for single steps then we use Data table concept and we separate the test data by using Pipe symbol.

Step Definition

- Inside the Step Definition package, we maintain all snippets.
- Inside the Snippets we write selenium code as per the steps wise.
- Inside th Step Definitions package we use Hooks.
- These Hooks are used to mention the preconditions as like post conditions.
- By using Hooks we can declare pre condition for feature level , Scenario Level as well as step level.
- There are 2 types of Hooks in cucumber
    1. Pre-condition Hooks
    2. Post Condition Hooks

Pre-condition Hooks

There are 3 types of pre conditions hooks

1. @BeforeAll
2. @Before
3. @BeforeStep

Post Condition Hooks

There are 3 types of Post Conditions hooks

1. @AfterStep
2. @After
3. @AfterAll

@BeforeAll It is pre-condition for all scenario in feature file And this @BeforeAll annotation available in Cucumber 6 and above version.
@AfterAll -It is post-condition for all scenario in feature file And this @AfterAll annotation available in Cucumber 6 and above version.
@Before It is pre-condition for each and every scenario in feature file
@After It is post-condition for each and every scenario in feature file
@BeforeStep It is pre-condition for each and every step-in feature file
@AfterStep It is post-condition for each and every step-in feature file

Test Runner class

- We can configure cucumber test runner class with Junit as well as with TestNG.

1. If we configure cucumber with Junit then we use 2 different annotation class level as
    - @RunWith() annotation and @CucumberOptions() annotation.
    - @RunWith() annotation tells run class using Cucumber Tests.
        - and Inside the @CucumberOptions we mention different configuration as like below.

2. If we configure cucumber Test Runer class with TestNG then we use @CucumberOptions() annotation in class level and we extends the AbstractTestNGCucumberTests class.
        - And inside the @CucumberOptions() annotation we do different configuration as like below.

- **@CucumberOptions() annotation configuration**
    - Feature location by using features keyword by passing feature file location.
    - Step definition file location by using glue or extraGlue keyword by passing Step Definition package name.
    - We check feature files steps vs implements snippets inside the step definition file is correct or not.
    - We display output console in proper readable format by using monochrome equal to true keyword.
    - We generate different reports by using plugin keyword and mention report name and its extension.
    - We run the specific scenario groups or skip the specific scenario groups by using tags keyword by passing tagName.

Diff between Background and Hooks?

- We use background keyword inside the feature file after the Feature keyword.
- Background keyword is used to mention pre condition.
- If we have multiple precondition steps for each and every scenario in feature then we declare inside the background keyword.
- Background keyword all steps it will execute running each and every scenario in feature file.

- We use Hooks annotation inside the step definition package
- By using hooks annotations, we can declare pre conditions as well as post conditions.
- If we want to declare precondition as well as postcondition for each and every scenario then can use @Before And @After hooks annotation for cucumber library.
- @Before precondition annotation it will execute before each and every scenario and @After postcondition annotation it will execute after each and every scenario.

If we mention background and hooks precondition annotation which one will execute first?

- We can declare precondition by using Background keyword as well as @Before precondition annotation.
- First priority goes to the @Before precondition annotation and then it will execute the Background keyword.

How to skip specific scenario?

- If want to skip scenario in cucumber, first we declare the group for scenario in cucumber feature by using @TagName and then we use same tagName inside the Test Runner,
- In Test Runner class we use tags keyword inside the @CucumberOptions annotation and inside the tags attribute we use Not keyword then we tagName.

What to Run feature file Parallel using Junit?

- we can run feature file parallel by using maven surefire plugin or failsafe plugin.
- First, we declare browser name parameter in feature file.
- And we use same browser name parameter inside the snippets.
- and write selenium script as per the scenarios
- then inside the pom.xml file add build tag name
- and inside the build tag name add plugins tag.
- inside the plugins tag add plugin tag.
- then inside the plugin tag we add maven surefire plugin or fail safe plugin.
- After adding the maven sure fire plugin or fail safe plugin we add configuration plugin and then we add includes tags and then we include tag and pass the test Runner class location, then close include and includes tag.
- Then we add parallel tag and pass methods as value and close parallel tag.
- and then add threadCount tag and pass the thread count number.
- We done all configuration we run the test cases by using maven.

```
<build>
	<plugins>
		<plugin>
			<groupId>org.apache.maven.plugins</groupId>
			<artifactId>maven-surefire-plugin</artifactId>
			<version>3.0.0-M3</version>
			<configuration>
				<includes>

<include>../TestRunner.RunnerTest.java</include>
				</includes>

				<parallel>methods</parallel>
				<threadCount>4</threadCount>
			</configuration>
		</plugin>

	</plugins>
</build>
```

What to Run feature file Parallel using TestNG?

- we can run parallel feature in cucumber with TestNG by using maven sure fire plugin or maven fail safe plugins.
- If we have to run feature file parallel, then we have create separate Runner classes for each and every feature files and then we create xml file for both the Test runner classes and we use parallel=methods in suite level annotations in testng.xml file.
- And we run the test cases from Xml files.
- Otherwise we can do more configuration, if want to run test cases by using maven.
- then inside the pom.xml file
- we add build tag name and inside the build tag name add plugins tag.
- inside the plugins tag add plugin tag, then inside the plugin tag we add maven surefire plugin or fail safe plugin.
- then inside plugin tag we add configuration tag and inside the configuration tag we add suiteXmlFiles tag and then inside the suiteXmlFiles tag we add suiteXml file tag by passing test.xml file name.
- then we add parallel tag and pass methods as keyword and close parallel tag.
- and then add threadCount tag and pass the thread count number.
- Once we declare above configuration we can run the test cases from maven.

```
<build>
	<plugins>
		<plugin>
			<groupId>org.apache.maven.plugins</groupId>
			<artifactId>maven-surefire-plugin</artifactId>
			<version>3.0.0-M3</version>
			<configuration>
				<suiteXmlFiles>
					<suiteXmlFile> testng.xml </suiteXmlFile>
				<suiteXmlFiles>
				<parallel>methods</parallel>
				<threadCount>4</threadCount>
			</configuration>
		</plugin>

	</plugins>
</build>
```

How to run scenario parallel in cucumber?

- we run scenario parallel by overriding the scenarios() method in test runner class. Once we override the scenario method in Test Runner class we use @DataProvider() annotation by passing parallel=true attribute.

How to read data from Excel sheet in Cucumber BDD Framework?

- We read data from Excel sheet by adding the Apache POI Dependency in pom.xml file.
- We have written reusable script code in Utility Layer package to read the data from Excel sheet, once we read all values from Excel sheet we convert these values to the List<Map<String, String>>.
- First we read the all file content by creating object of FileInputStream class then we load all sheets by creating object of XSSFWorkbook by passing FileInputStream instance, basically there are 2 types of Excel sheet 1st is .xlsx and 2nd .xls, in my current project we use .xlsx excel sheet.
- Once we load all the sheets then we focus on specific sheet using getSheet() method by passing sheet name and .getSheet() method will focus on specific sheet, and once focus on specific sheet then we count how many rows in sheet using getLastRowNum() method.
- Then we have create object of List<Map<String, String> class, then we iterate all the rows using for loop and inside the loop we count how many columns inside the rows by using getLastCellNum() method.
- Then we have created one more object of LinkedHashMap<String, String> then we iterate all columns by using loop.
- Inside the inner loop we capture the sheet column name by using getStringCellValue() method, as well as we capture cells values using getStringCellValue() method.
- Once we capture column heading and cells value then we put these values inside the LinkedHashMap object by using put() method by passing column heading name and cells name.
- Then outside inner for loop we add LinkedHashMap object into the List<Map<String,String>> object using add() method by passing LinkedHashMap object name.
- Then outside outer for loop we return the List<Map<String,String>> object.
- And we call these function inside the step Definition. Before calling these function we mention sheet name and row number inside the Feature file.
- We can use these sheet name and rows inside the Datatable also.