Cucumber

BDD FRAMEWORK

Behavior Driven Development

- Approach to development where we use “ubiquitous” language to bridge the gap between business and development teams
- Developers implement features through the point of view of non-technical stakeholders
- BDD is a superset of TDD (meaning TDD is a part of BDD)
  - BDD takes it further by encouraging effective communication between all parties such that everyone has a strong shared understanding of an application’s behavior not only the development and QA (testing) teams

Steps of developing features

1. Business Analysts (BAs) and stakeholders collaborate with developers and testers to document expected behaviors in plain language (English-like) syntax (“Ubiquitous language”)
2. Developers/testers write tests to validate the behaviors described
3. Developers then write code to pass the test (TDD)

BDD benefits

- Encourages documentation that is understandable to both technical and non-technical parties. BDD is commonly referred to as “living documentation” because it evolves with the software itself
- Emphasize effective communication between non-technical and technical members of a team
- BDD allows us to easily generate test cases from provided scenarios

Implementing BDD w/ Cucumber

1. Developers/BAs first write feature files in Gherkin. Feature files define the various scenarios and steps that define a system's behavior at a high level
2. Once a developer has drafted the feature file, they should run the feature file to generate the "glue-code". Glue-code refers to the potential test/automation methods that are associated with a scenario's steps. A developer can then choose whether or not the implement those methods.
3. Once a developer has generated glue code and written tests, they should choose a test runner to run the steps

Some Cucumber Terminologies

- Feature file: A file with a .feature extension that contains Gherkin syntax in the form of scenarios
- Glue code / Step definition file: Actual Java implementations of our features
- Test Runner: a class that contains configuration regarding where to find feature files and step definition files and will actually run these steps as tests

Gherkin Example

```
Scenario: Logging in Successfully
    Given User is on the login page
    And The username input is present
    And The password input is present
    And The submit button is present
    When User enters a valid username
    And User enters a valid password
    And User presses the submit button
    Then The landing page should be displayed
    But There should not be any display of 'undefined'
```

Cucumber Parameterization

It is possible to define parameters that you want to pass into your glue code methods themselves from the feature files. This allows us to define parameters that are visible from the perspective of the Gherkin itself and couple it with our tests

1. The ability to pass parameters “inline”
2. The ability to pass parameters in a table (similar to the below example, but for a single step) “x rows of data, run that single step x times”
3. The ability to define a “scenario outline” that will run multiple times for each “row” of data
   - If we have x rows of data, run that scenario x times