COURSEWARE

- Professional Skills
- Agile Fundamentals
- Jira
- Git
- Databases Introduction
- Java Beginner
- Maven
- Testing (Foundation)
- Java Intermediate
- HTML
- CSS
- Javascript
- Spring Boot
- Selenium
- Sonarqube
- Advanced Testing (Theory)
- Cucumber
  - ○ Intro to Cucumber
  - ○ Setting up Cucumber
  - ○ Feature Files
- MongoDB
- Express
- NodeJS
- React
- Express-Testing
- Networking
- Security
- Cloud Fundamentals
- AWS Foundations
- AWS Intermediate

Feature Files

Contents

Overview

Here we'll review the purpose and function of a feature file and the various keywords we can utilise.

Gherkin & Feature Files

Gherkin is the plain language format; in particular, the `.feature` format which acts as a template for Behaviour-Driven Development to express our user criteria.

One of the major benefits of this is that Gherkin is language agnostic, its purpose is to only specify what the steps should be as part of BDD but it doesn't matter which spoken language it is written in.

> You can view other language localisations on Cucumbers Official Documentation

```
Another benefit of this language-centric format is that it's understood by
everyone involved, so even non-technical users could create a Gherkin file to
outline their requirements!
```

Feature files will be handled by Cucumber and are translated into tangible Junit test steps that are driven by Cucumber instead!

> Important: Junit is still used, but will refer to cucumber to drive tests, this means our Junit assertions will still work!

Primary Keywords

Feature files are typically crafted using keywords derived from BDD, as such there are some core keywords we should focus on to craft our first feature file.

Keyword	Description
Feature	Outlines the main purpose of the feature file; a topic/feature under investigation.

Linux
DevOps
Jenkins Introduction
Jenkins Pipeline
Markdown
IDE Cheatsheet

Keyword	Description
Scenario	A user story that describes the current goal of interaction.
Given	A `prerequisite`, a circumstance required before actions can take place.
When	An `action`, some form of interaction intended to navigate or cause an effect.
Then	An `observable result` or change, usually assertable.

Additional Keywords

We can use additional keywords to help reinforce good expression and expand our capability, defining multiple steps with unique keywords.

Keyword	Description
And	Used to repeat the previous Keyword, outline an addition to the `action`, `prerequisite` or `result` required.
*	Like the `And` Keyword, a generic expression/step that acts well to outline lists within certain scenarios.
But	Rather than the inclusive nature of the `And` keyword, provides an exclusion instead, functionally identical to the `And` keyword.
Background	Acts like a `Scenario`, Backgrounds are shared steps that all Scenarios perform first, useful for outlining shared `Given` steps.

Tutorial

Below are explanations for each keyword and how you might use them in your Cucumber project.

Keyword - Feature

Each feature focuses around a `topic` under test, it is usually briefly described after using the feature keyword in conjunction with the `colon`/`:` key.

```
Feature: Animal Search Functionality
```

Additionally, we can further expand on a feature and provide a commented breakdown.

```
Feature: Animal Search Functionality

    Focusing on determining if we can successfully find
    websites associated with specific animals located on
    http://www.google.com
```

> - Return to Keyword Table

Keyword - Scenario

Scenarios are specific user criteria, they might outline a particular area of testing by addressing a user requirement.

Whereas a feature might outline that we are testing the google search engine, a scenario would be a circumstance a user story would use that feature to achieve a goal.

```
Feature: Google Search Functionality

Scenario: find kittens images on google
    Given  - text here -
    When  - text here -
    Then  - text here -
```

A feature file might contain many Scenarios, be aware that the scenario keyword is followed by a colon, but outlined steps such as `Given`, `When`, and `Then` are indented to show they belong to that scenario.

> - Return to Keyword Table

Keyword - Given

`Given` is our initial BDD state.
It is best thought of as "Given a requirement" or a prerequisite, it allows us to check the prior or existing state of our initial starting point.

```
Feature: Google Search Functionality

Scenario: find kittens images on google
    Given we can open google
    When  - text here -
    Then  - text here -
```

> - Return to Keyword Table

Keyword - When

`When` is our action phase.

This usually is a step that describes navigation, interaction, and the general flow/lifecycle of user interaction to achieve a goal.

```
Feature: Google Search Functionality

Scenario: find kittens images on google
    Given we can open google
    When we search for kittens
    Then  - text here -
```

> - Return to Keyword Table

Keyword - Then

`Then` is our observable state.

It's the end reward for successfully interacting with your targeted website/ or presentation layer.

Perhaps you have just submitted an order, you can now state that "Then I should be able to see the order confirmation". It is our usual assertable goal for the test.

It is important to note that we can and arguably should assert for each step, failure on a step with an assertion can help Junit inform us as to what that test step failed on. But more importantly what returned instead.

```
Feature: Google Search Functionality

Scenario: find kittens images on google
    Given we can open google
    When we search for kittens
    Then google will return us images of kittens
```

- [Return to Keyword Table]()

Additional Keyword - And

`And` is a generic step that syntactically ties to the previous step.
If an `And` step exists after a `Given`, it is as though it is simply described as another Given; however, the flow of the conversion becomes a notion of inclusion to requirements.

Likewise, an `And` underneath a `When` or `Then` would reflect that syntax as part of our Gherkin BDD.

```
Feature: E-commerce Site

    Scenario: log into the account
        Given I have a Username
        And I have a Password
        When I click the login section of the landing page
        And I enter my credentials into the form
        Then I should receive a login success message
        And I should be navigated to my account.
```

- [Return to Keyword Table]()

Additional Keyword - *

The `*` keyword is a reflection of and, it's appropriate to use this as a though it were a bullet point list, but is a general expression of the previously mentioned syntax.

The `*` keyword can generally act as a generic step definition but is better when used as a list of requirements or outline a list.

```
Feature: E-commerce Site

    Scenario: log into the account
        Given I can access the site & have credentials such as
        * A username
        * A password
        When I click the login section of the landing page
        And I enter my credentials into the form
        Then I should receive a login success message
        * See my username
        * See my balance
        * See the number of items in my basket
```

- [Return to Keyword Table]()

Additional Keyword - But

The `But` keyword simply acts like our `And`; referring to the prior step, however, it allows us to take advantage of `inclusive vs exclusive` nature.

With `And`s we are setting a precedence for inclusion; they infer that you need additional factors to be true. `But` is used like the boolean `not`/`!` operator, implying that something shouldn't be the case at that stage (given, when, or then).

```
Feature: E-commerce Site

    Scenario: log into the account
        Given I have a Username
        And I have a Password
        But I'm not logged in
        When I click the login section of the landing page
        And I enter my credentials into the form
        Then I should receive a login success message
        And I should be navigated to my account.
```

- Return to Keyword Table

Additional Keyword - Background

`Background` is provided when a feature file may have many scenarios that all share the same prior steps. The Background keyword can be used to describe a set of steps common to all scenarios, especially the `Given` steps.

```
Feature: Shopping Functionality

    Background:
        Given I'm at the supermarket
        And I have a list of items
        And I have my card to pay for the gathered items

    Scenario: 1
        When I do something
        Then a result returns

    Scenario: 2
        When I collect a fruit item
        And weigh the item
        Then I can determine the price of the item
```

- Return to Keyword Table

Exercises

- Try writing another scenario in your feature file, try to search for another animal!