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

# Intro to Cucumber

## Contents

- Overview
- What is Cucumber?
- The basics of Cucumber
  - Gherkin
  - The BDD Loop
- Tutorial
- Exercises

## Overview

Cucumber is a tool used to automated the running of executable specifications containing acceptance criteria in a *domain specific language*.

## What is Cucumber?

Cucumber is a **specification tool** commonly used for behaviour-driven development (BDD), utilises the Gherkin **domain specific language** (DSL) for describing system behaviours (features) in a naturally cohesive and widely understood form. It is not by itself a testing framework, for this we must include other dependencies like JUnit and Mockito for example.

- Cucumber is available for multiple programming languages including JVM Languages, Android Java, Ruby, OCaml, and more...
- Cucumber can read specifications written in Gherkin and generate *test cases* based on the documented system behaviours, these then become **executable specifications** which can be used to automate the running of tests.

## The basics of Cucumber

Cucumber is a large framework that offers lots of functionality for BDD and producing executable specifications, some basic principles are introduced now.

### Gherkin

Cucumber uses the Gherkin domain specific language (DSL) to write executable specifications, its syntax is simple and can be used in English or other languages. A basic syntax table for writing acceptance criteria follows:

| Gherkin keyword | Description | Example |
|---|---|---|
| Given | Used for preconditions in a scenario. | Given the number 2 |
| When | Used for listing some action/behaviour in a scenario. | When I add 5 |
| Then | Used for listing some action/result that should occur because of the prior steps in a scenario. | Then the answer should be 7 |

Linux
DevOps
Jenkins Introduction
Jenkins Pipeline
Markdown
IDE Cheatsheet

| Gherkin keyword | Description | Example |
|---|---|---|
| And | Used to repeat the prior keyword. | Given the number 2 And the number 5 |
| * | Used to repeat the prior keyword. | Given the number 2 * the number 5 |

▶ Example for withdrawing cash from a bank machine

The other important parts of Cucumber are *feature files* and *step definition files*:

- **Feature file**: A file describing a specific system behaviour and the different scenarios it should be tested under
- **Step definition file**: A code file, Java in our case, that implements the scenarios of a feature

Feature files describe a specific system behaviour and some scenarios to test that behaviour, a feature file has the following outline:

```
Feature: [The feature to be tested]

    Optional feature description here...

    Scenario: [Scenario to be tested]

      Given [some preconditions]
      When [some action occurs]
      Then [this should occur]
```

These will be explored further in the following modules.

## The BDD Loop

BDD has a simple 3 step process that can be followed, defined by SmartBear Software (creators of Cucumber).

SmartBear recommends the following 3 steps as part of the BDD iterative workflow to accurately and rapidly document, develop and test new software features:

1. **Discovery**: Discover the scope of the systems behaviour
   - Explore, discover, question and agree upon concrete examples of user stories and acceptance criteria using the client specification
   - Include the lead tester, lead developer, and product owner in this phase
2. **Formulation**: Create the specification in a business readable language
   - Document the examples in a way that can be automated
   - Check for agreement upon the documented examples
3. **Automation**: Verify the systems behaviour via automation
   - Write code to implement behaviour and tests

This process is seen to be iterative, when one iteration finishes another begins. In *discovery*, we formulate concrete examples of user stories and acceptance criteria, i.e. *As a user, when..., Given ..., When ..., Then ...*). In *formulation*, the concrete user stories and acceptance criteria are documented in an **executable specifications** which can then be *automated* once the behaviour and tests have been implemented.

## Tutorial

There is no tutorial for this module.

## Exercises

1. Create a set of steps, using Gherkin, for depositing cash into a bank machine that could be used as a basis for testing.
2. Create feature files using the above structure for testing the subtraction, division and multiplication features of a calculator. The addition feature is supplied below as an example:

▶ Addition steps in Gherkin