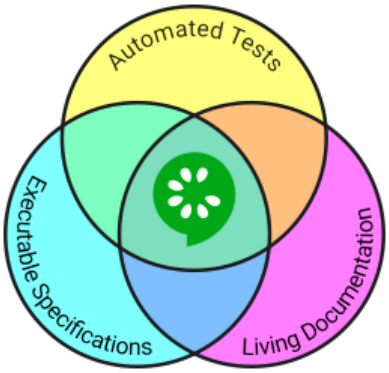
cucumber

BDD & Cucumber Training Outline

About Behaviour-Driven Development



BDD - Behaviour-Driven Development - is an approach to software development that not only focuses on fast and maintainable automated tests, but also on more precise requirements specifications using collaboration techniques for the whole team.

These techniques help non-technical stakeholders communicate better with developers and testers so they can discover misunderstandings before the software is written. BDD encourages testers to shift their focus from finding bugs to preventing bugs.
With BDD and Cucumber, the whole team collaborates on executable specifications in plain text that serve as both automated tests and living documentation of the implemented system. In other words, a single source of truth for the software, shared and owned by the whole team.

BDD Fundamentals

Course outline

A 2-day course, without computers, for the whole team. The course is exclusively hands-on, interactive, and experiential. There are no slides and no lecture element lasts for more than 10 minutes.

Covers the full BDD process and the following BDD practices:

- Discovery
- Formulation

Day 1: Discovery and process

Topics covered:

- BDD Principles
    - Deliberate discovery - “learning is the constraint”
    - Rules & examples - early testing of requirements and understanding
    - Living documentation - specifications that tell you when they are incorrect
- The 3 Practices of BDD - Discovery, Formulation, Automation
- Deliberate discovery with examples - an introduction to examples
    - Essential vs. incidental
    - Concrete vs. abstract
    - The importance of context
- Discovery workshops
    - Guidelines for a productive workshop
    - Example mapping - structuring the conversation
    - Alternatives to index cards
- “Story-sized” user stories
    - User story splitting techniques
    - Pitfalls of estimation
    - Importance of slack
- Introduction to Gherkin and Cucumber/Specflow

- Ideal BDD process
    - How to fit BDD into your process
    - Who does what
    - Variations and pitfalls

Day 2: Formulation and benefits of “driving” development

- Good Gherkin
    - Basic Gherkin
    - Critiquing scenarios
    - Gherkin tips and best practices
    - Writing Gherkin - basic and advanced
- Evolving the ubiquitous language
    - Readability, ambiguity, and bounded contexts
    - Glossaries and alternatives
- Tip of the iceberg
    - Workflow scenarios - for the big picture and smoke tests
- Organising Gherkin
    - Publishing Gherkin
    - Remote collaboration
    - Structuring large bodies of living documentation
- Driving development with automation
    - Automate-first development
    - Principles of TDD (for the **whole** team)
    - The continuing need for testers
    - Automation as a development skill
- Basics of Cucumber/Specflow (for the **whole** team)

BDD with Cucumber

Course outline

A 1-day technical course, with computers, for the developers, testers and the tech-curious team members. The course is exclusively hands-on, interactive, and experiential. There are no slides and no lecture element lasts for more than 10 minutes.

The course can either be delivered using your in-house development environment or using our web-based training environment.

Covers the essential functionality Cucumber and the following BDD practice:

- Automation

Automation and Cucumber/Specflow

Topics covered:

- Structure of a Cucumber project - files, folders, and configuration
- Interaction of BDD and TDD - interpretation and response to failing tests
- Step definitions
    - Regex and/or Cucumber expressions
    - Common failure modes
    - Parameterising your stepdefs
    - Keeping your stepdefs short
- Automating a new scenario
    - Undefined steps
    - Snippet generation
    - Common pitfalls and failures
- Cucumber life-cycle
    - What is Cucumber actually doing
    - Ambiguous steps - and what to do about them
    - Error signalling - and how not to fix them
    - Isolation - what it is, why it's important, and how Cucumber helps
- Importance of code quality in your “test” code

    - Keeping your tests “truthful”
    - Refactoring techniques
    - Extracting complex setup into “support” code
    - Sharing “support” code with programmer tests
    - Preventing test concerns from leaking into production code
- Gherkin features - how to use them, when to use them, and when not
    - Backgrounds
    - Scenario outlines
    - Tags
    - Hooks
    - Data tables
- Growing your specification
    - How to split your stepdefs among multiple files
    - How to share state between multiple step definition files
- The pyramid and the iceberg
    - Keeping scenarios focused on business behaviour
    - Decoupling documentation from automation decisions