Acceptance Test-driven Development (ATDD) with Cucumber-jvm (and friends)

Christopher Bartling

What is Acceptance Test-driven Development?

- Elisabeth Hendrickson

- A practice in which the whole team collaboratively discusses acceptance criteria, with examples, and then distills the criteria and examples into a set of concrete acceptance tests before development begins.

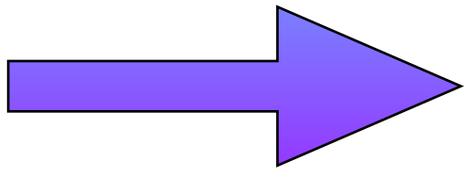
- Builds a shared understanding of what we're building and when to consider it done.

Acceptance test-driven development cadence

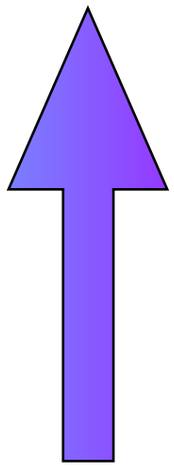
- **Discuss** the requirements and acceptance criteria for the feature, creating examples.
- **Distill** these examples into executable test format.
  - Cucumber: Create features and scenarios.
- **Develop** the feature and hook up the acceptance tests.
  - Cucumber: Create step definitions and page objects.
- **Demo** the feature and successful acceptance tests.

The acceptance test-driven cadence

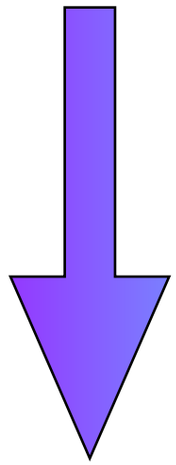
Discuss requirements, create examples



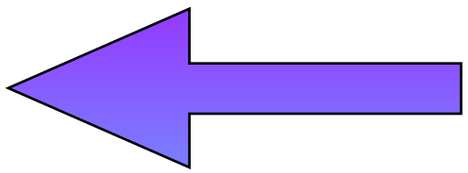
Distill examples into acceptance tests



Test-driven development



Demo features and acceptance tests



Develop features, hook up tests

Specification by example

- Gojko Adzic
- “Living documentation”
  - Documentation from executable specifications.
  - Promotes collaboration.
  - Efficient validation of features.
  - Implement features more effectively and with higher quality.

Benefits of automated acceptance testing

- Forces a thorough analysis of a feature.
- Build concrete agreement about the exact behavior the feature should exhibit.
- Living, executable and trustworthy documentation.
- Automation allows more manual, exploratory testing.
- Scalable regression testing.

Cucumber

- Tool for running automated acceptance tests written in a behavior-driven development (BDD) style.
  - *Given - When - Then*
- Allows the execution of feature documentation written in business-facing text.

Cucumber-jvm

- Java implementation of Cucumber.
- Supports popular JVM programming languages.
  - Java, Groovy, Clojure, Scala, JRuby.
- Integrates with popular dependency injection containers.
  - Spring Framework, Guice, PicoContainer.
- Support several runners: command-line, JUnit, Android.

Groovy

- Dynamic language for the Java Virtual Machine.
- Supports Domain-Specific Languages (DSLs) and other compact syntax so your code becomes easy to read and maintain.
- Seamlessly integrates with existing Java classes, libraries, and frameworks.
- Compiles directly to standard JVM bytecode.

Selenium WebDriver

- Automated browsing. Capabilities include:
  - Navigating to web pages
  - Handling user input
  - Finding and interrogating the DOM
  - JavaScript execution
- **org.openqa.selenium.WebDriver** is the the key Java interface against which tests should be written.

Geb

- Browser automation abstraction over WebDriver.
- Provides an elegant jQuery-like content selection engine.
- Built with Groovy, providing an easy to use DSL.
- Useful for scripting, scraping and general automation of web browsers.
- Integrates nicely with Cucumber-jvm.

Chromedriver

- Standalone server which implements WebDriver's wire protocol for Chromium.
- Consists of three separate pieces:
  - The browser itself (“chrome")
  - Language bindings provided by the Selenium project ("the driver”)
  - Bridge server between "chrome" and the “driver" (the **chromedriver** binary, referred to as the “server").
- Developed by members of the Chromium and WebDriver teams.

Gradle

- Build automation tool.
- Combines the power and flexibility of Ant with the dependency management and conventions of Maven into a more effective way to build.
- Provides a declarative way to describe all kinds of builds through sensible defaults.
- Powered by a Groovy DSL.

Cucumber features

- Plain-text functional scenarios.
- Simple Given/When/Then syntax (Gherkin).
- Feature files are typically written before anything else and verified by business analysts, domain experts, etc. non technical stakeholders.
- Production code is then written *outside-in*, until the stor(ies) successfully pass.

Cucumber feature example

```
Feature: Navigate through the NCAA basketball tournament
site

 Background:
    Given I navigate to the website

 Scenario: Navigate to the first round of the tournament
    When I click the "First round" navigation pill link
    Then I should see the first round brackets

 Scenario: Navigate to the second round of the tournament
    When I click the "Second round" navigation pill link
    Then I should see the second round brackets
```

Cucumber step definitions

- The feature file “code-behind”.
    - The executable part of Cucumber.
- Uses regular expressions to bind to feature file phrase.

```
When(~'^I click the "([^"]*)" navigation pill link$')
{ String navPillText ->
    page.$('a', text: navPillText).click()
}
```

Cucumber Support

- Cucumber provides a number of hooks which allow us to run blocks at various points in the Cucumber test cycle.
- Place hook implementations in file under the support directory.
- Use tagged hooks if you want more fine grained control.
- All defined hooks are run whenever the relevant event occurs.

Page object pattern

- Allows modeling web content in a reusable and maintainable way.

- Reduces the amount of duplicated code.

- If the UI changes, the fix need only be applied in one place.

- Geb provides support for the Page Objects pattern through the **geb.Page** and **geb.Module** abstractions.

Demonstrations

Reflection

- When you plan features, are you enumerating acceptance criteria and coming to shared understanding of Done?

- Are you converting those acceptance criteria into executable examples in an automated testing framework?

- What is preventing you from using acceptance test-driven development?

GitHub repository

- https://github.com/cebartling/ncaa-basketball-tournament
    - **web-client**: Backbone.js-based application
    - **acceptance-tests**: Cucumber-jvm project, Gradle-based build

Resources

- http://testobsessed.com/2008/12/acceptance-test-driven-development-atdd-an-overview/
- http://testobsessed.com/wp-content/uploads/2011/04/atddexample.pdf
- http://specificationbyexample.com/

Thank you!

- Christopher Bartling
  - @cbartling
  - chris@pintailconsultingllc.com